International Database Engineered Applications Symposium

IDEAS 2026

30th International Database Engineered Applications Symposium

Concordia University
May 20 - 21, 2026,
Montreal, Canada

IDEAS 2026

30th International Database Engineered Applications Symposium

Concordia University, Montreal, Canada
2026-05-20– 2026-05-21

Proceedings Editor
Bipin C. Desai, Concordia University, Canada

Program Chairs
Bipin C. Desai, Concordia University, Canada,
Rui Chen, Samsung, United States

ISBN: 978-1-988392-18-9

Editorial production by: ELECTRONIC PUBLISHING BYTEPRESS.COM INC
Cover Artwork by: Logo Copyright Lettie

Table of Content

Full Paper

Full Paper(Continued)

Full Paper(Continued)

Preface

These are the proceedings of the 30th annual event of IDEAS. When the first of this series was planned, I had no idea that it would successfully continue for all these years. Some members of the original and current steering committee, including myself, are stepping down after this meeting. This is also going to be the last, of over 40, meetings of C^3S^2E and IDEAS I would manage as I am soon to retire! It was a honour to have met hundreds of colleagues, many just starting their career. I am glad that IDEAS could have helped them in their early development.

I will continue writing and supervising students but I hand over this task of continuing with IDEAS to my good colleagues Jorge Bernardino, Carson Leung and Peter Revesz. To keep up with the current academic focus of database application, I suggest a new avatar for IDEAS series with the full name as International Database Engineering and Artificial-intelligence Symposium.

We continue to struggle with the challenge of holding highly selective meetings such as IDEAS with an increasing number of conferences which use the all invited papers or speakers approach with papers guaranteed to be accepted. We had some interesting papers and selected some 65% of the submissions with a shorter meeting than originally planned. We are honoured to have had as keynote speaker: Dr. Jiawei Han, Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign.

I would like to take this opportunity to thank the general, program, local and publicity chairs and the program committee for their help in the review process. All the submitted papers were assigned to four reviewers and we got back over 3.5 reviews per paper.

Bipin C. DESAI
Editor, IDEAS 2026
Montreal, Canada

Reviewers
from the Program Committee

* Toshiyuki Amagasa(Tsukuba University, Japan)
* Samuel Appleby(Newcastle University, United Kingdom)
* Masayoshi Aritsugi(Kumamoto University, Japan)
* Orlando Belo(Universidade do Minho, Portugal)
* Giacomo Bergami(Newcastle University, United Kingdom)
* Jorge Bernardino(Instituto Politecnico de Coimbra, Portugal)
* Flavio Bertini(University of Parma, Italy)
* Minal Bhise(Dhirubhai Ambani Institute for Information and Communication Technology, India)
* Alina Campan(Northern Kentucky University, United States)
* Rui Chen(Samsung, United States)
* Marcos Aurelio Domingues(Universidade Estadual de Maringa, Brazil)
* Nuno Escudeiro(Instituto Politecnico do Porto, Portugal)
* Alberto Freitas(Universidade do Porto, Portugal)
* Sven Groppe(Medizinische Universitat Lubeck, Germany)
* Irena Holubova(Charles University Prague, Czech Republic)
* Min Huang(Nakisa Inc, Canada)
* Sergio Ilarri(Universidad de Zaragoza, Spain)
* Dimitrios Katsaros(University of Thessaly, Greece)
* Mohamed Kechar(Universite d'Oran Es-Senia, Algeria)
* Pavel Koupil(Charles University Prague, Czech Republic)
* Wookey Lee(Inha University, Korea Republic)
* Carson K. Leung(University of Manitoba, Canada)
* Chuan-ming Liu(National Taipei University of Technology, Taiwan)
* Hakim Mellah(Concordia University, Canada)
* Danilo Montesi(University of Bologna, Italy)
* Sudhir P. Mudur(Concordia University, Canada)
* Paulo Jorge Oliveira(Instituto Superior de Engenharia do Porto, Portugal)

Reviewers
from the Program Committee
(Continued)

* Kalpdrum Passi(Laurentian University of Sudbury, Canada)
* Valeria Magalhaes Pequeno(Escola Nautica Infante D. Henrique, Portugal)
* Giuseppe Polese(University of Salerno, Italy)
* Lubos Popelinsky(Masaryk University, Czech Republic)
* Peter Z. Revesz(University of Nebraska - Lincoln, United States)
* Jose Ramon Rios Viqueira(Universidad de Santiago de Compostela, Spain)
* Miguel RodriÃƒâ€šÃ‚-guez Luaces(Universidade da Coruna, Spain)
* Marinette Savonnet(Universite de Bourgogne, France)
* Nematollaah Shiri(Concordia University, Canada)
* Giorgio Terracina(University of Calabria, Italy)
* Krishnamurthy Vidyasankar(Memorial University of Newfoundland, Canada)
* Alicja Wieczorkowska(Polish-Japanese Institute of Information Technology in Warsaw, Poland)
* Roberto Yus(University of Maryland Baltimore County, United States)
* Ester Zumpano(University of Calabria, Italy)

External Reviewers

* Will Knight(BytePress.org, Canada)

* Gerry Laval(ConfSys.org, Canada)

* James Rant(BytePress.org, Canada)

IDEAS 2026

Program Chairs
Bipin C. Desai, Concordia University, Canada,
Rui Chen, Samsung, United States

Local Chair
Joey Paquette, Concordia University, Canada,

Steering Committee
Bernardino, Jorge (Co-Chair) Polytechnic of Coimbra
Desai, B. C.(Chair, retiring) Concordia University
Chen, Rui Samsung Gruenwald,
Le Univ. of Oklahoma Holubova,
Irena Charles University Kawashima,
Hideyuki Keio University Kuijpers,
Bart Hasselt University Leung,
Carson K. (Co-Chair) University of Manitoba
Manolopoulos, Yannis Univ. of York, Europe Campus
McClatchey, Richard(retiring) Univ. of the West of England
Munir, Kamran UWE Bristol Ng,
Wilfred(retiring) HKUST
Pokorny, Jaroslav(retiring) Charles University
Revesz, Peter Z. (Co-Chair) Univ. of Nebraska-Lincoln
Toyama, Motomichi Keio University Ullman,
Jeffrey(retiring) Stanford University

LLM-Based Database Knob Tuning: A Taxonomy, Experimental Study, and Research Agenda

Jamshed Karimnazarov[0009−0000−0559−888X] and Le Gruenwald[0000−0002−5245−4747]

University of Oklahoma, Norman OK 73019, USA
{jamshed.k,ggruenwald}@ou.edu

Abstract. Tuning database configuration knobs is a difficult task that can significantly affect database performance. Recent work has explored the use of Large Language Models (LLMs) to automate this process, but existing methods are often not compared directly, are evaluated under different settings, and survey literature does not cover LLM-based techniques. To address these limitations, in this paper, we present a taxonomy of LLM-based database knob tuning techniques and conduct a unified experimental comparison of six existing methods under consistent DBMS and benchmark settings. We further identify research opportunities in this area.

Keywords: Database management · Large Language Models · Database knob configuration tuning.

1 Introduction

Modern database systems have hundreds of configuration knobs, and tuning these knobs can significantly improve database performance. PostgreSQL, for example, provides over 300 configuration parameters [13] which control different aspects of the database behavior. For instance, *shared_buffers* is a resource-consumption parameter that determines how much memory is allocated for caching, while *max_connections* controls the maximum number of concurrent connections. Traditionally, the task of tuning these knobs is carried out by experienced database administrators (DBAs). However, this process is extremely time-consuming and may require years of experience. Finding optimal knob settings has therefore remained a long-standing challenge in the database community, largely due to the size and complexity of the tuning space [32].

Prior database knob tuning research is commonly categorized into four types: *(1) heuristic, (2) Bayesian optimization (BO)-based, (3) deep learning-based, and (4) reinforcement learning (RL)-based tuning methods* [33]. Heuristic methods include PGTune [1] and OpenTuner [2], while BO-based methods include OtterTune [27] and ResTune [30]. Although limited in research, deep learning techniques such as Deep Neural Networks (DNN) [25] and iBTune [20] have been explored. CBDTune [28] and HUNTER [4], on the other hand, rely on RL.

More recently, with advances in Large Language Models (LLMs) such as GPT-4 [15], there has also been a shift toward leveraging LLMs in the database knob tuning process. Several proposals have incorporated LLMs into the tuning workflow, demonstrating the ability of LLMs to fully or partially automate the tuning process [23,13,10,11,7,18]. However, despite the growing number of LLM-based knob tuning methods, the current body of work has several limitations. The existing LLM-based knob tuning studies are evaluated under different experimental settings, workloads, benchmarking tools, and evaluation metrics, making direct comparisons difficult. Additionally, LLM-based tuning introduces additional cost dimensions, such as token usage, fine-tuning cost, and model preparation time, which are not always considered. While some studies include limited comparisons between LLM-based techniques, the recency of this research area means that not all approaches have been evaluated against each other.

Prior studies have surveyed database knob tuning techniques and provided broader experimental comparisons. However, these works mostly focus on traditional machine learning (ML)-based techniques and predate LLMs [33,29]. To address these limitations, this paper makes the following contributions:

- We present the first taxonomy of LLM-based database knob tuning techniques and organize existing methods along key design dimensions
- We benchmark six LLM-based knob tuning methods using five metrics and analyze their performance and cost across database and workload sizes
- We identify broader research gaps and future directions in LLM-based database knob tuning

The structure of the paper is as follows. Section 2 provides background information. Section 3 presents a taxonomy of LLM-based knob tuning techniques. Section 4 describes the selected tuning algorithms. Section 5 outlines our experimental methodology. Section 6 presents the results, Section 7 discusses research opportunities, and Section 8 concludes the paper.

2 Background

The following sections provide background on the key concepts and terminology used in this paper, especially LLMs and BO.

2.1 Large Language Models (LLMs)

LLMs are a subset of language models within the Natural Language Processing (NLP) domain. Trained on massive text corpora, LLMs learn to predict the likelihood of future tokens conditioned on prior context [32]. The introduction of the transformer architecture, and in particular the self-attention mechanism, made it possible to train models with billions of parameters efficiently and in parallel [26]. GPT-3 [3] demonstrated the capabilities of LLMs as few-shot learners, while InstructGPT [16] introduced RL from human feedback,

enabling LLMs to better follow user intent. Within database research, LLMs have been used in text-to-SQL translation [8], query rewrite [19], index advising [14], database diagnosis [34], and predicting data correlation from column names [24]. More recently, LLMs have been applied to database knob configuration tuning [23,13,10,18,7,11]. A detailed discussion of the LLM-based knob tuning algorithms is provided in Section 4.

2.2 Bayesian Optimization (BO)

BO is a method for optimizing an expensive black-box function when the objective is costly to evaluate [33,9]. It begins by evaluating the function at a small number of initial points. The initial points are sampled using random sampling or Latin Hypercube sampling. A probabilistic surrogate model, commonly a Gaussian Process, is then fitted to approximate the objective function. The surrogate provides both a prediction of the value at unseen points and an uncertainty estimate. An acquisition function then chooses the next point by balancing exploitation of promising regions and exploration of unknown regions. After BO selects the next input to evaluate, the objective function is evaluated at that point, and the resulting output is observed. The new observation is added to the dataset, the surrogate model is updated, and the process repeats.

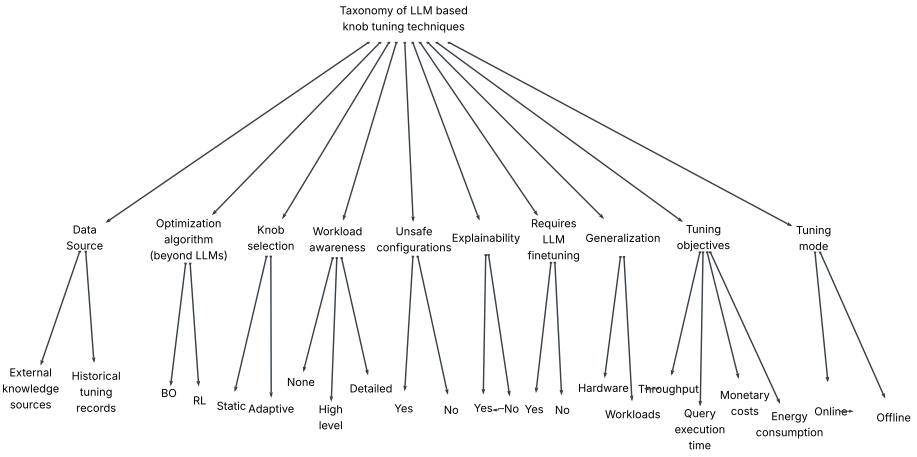


Fig. 1: Taxonomy of LLM-based knob tuning techniques

3 Taxonomy of LLM-based Knob Tuning Techniques

As discussed earlier, in recent years, several techniques have been proposed to use LLMs for database knob tuning. Figure 1 presents a taxonomy of LLM-based knob tuning techniques across several design dimensions. In this section,

we describe these dimensions, explain what they capture, and discuss why they are important for analyzing and comparing existing methods.

3.1 Data Source

Data source refers to the information a tuning method uses to select knobs, generate configurations, or guide the search. This dimension is important because while pre-LLM tuning methods often relied on historical tuning data [27], LLM-based techniques may use external knowledge sources, historical tuning data, or a combination of the two [13,18]. External knowledge can include DBMS manuals, documentation, blogs, and discussion forums. Historical tuning records consist of previously observed tuning run results with configurations and observed performance metrics. The absence of an external data source would imply reliance on the LLM's internal pre-trained knowledge.

3.2 Optimization Algorithm (Beyond LLMs)

An optimization algorithm refers to the mechanism that searches for an effective configuration beyond the LLM itself. This dimension is important because in many systems, the LLM does not always replace the traditional optimizers entirely [23,13]. The most common methods we have identified are BO and RL. However, this set is not exhaustive, and future methods may incorporate other optimization techniques. Some approaches may also primarily rely on the LLM to directly generate candidate configurations without a separate optimization mechanism [10].

3.3 Knob Selection

Knob selection determines whether the set of knobs to tune is fixed or dynamically adjusted for a given tuning task. The importance of this dimension comes from the fact that tuning all available knobs is practically impossible, and performance can depend significantly on the subset of selected knobs [33]. Static approaches rely on a predefined list of knobs identified through prior knowledge or expert selection. Dynamic approaches can select knobs based on workload types, features, system information, or other characteristics. This introduces a new decision layer into the tuning pipeline.

3.4 Workload Awareness

Workload awareness refers to whether the tuning method explicitly incorporates workload characteristics when generating configurations. Providing the entire workload SQL to the LLM is often expensive [10], so existing techniques typically provide summarized workload characteristics. Based on the level of workload information included in the prompt, we categorize workload awareness into three categories: none, high-level, and detailed. Methods in the none category do not

use any workload characteristics [23]. High-level workload characteristics often include the tuning objective, workload type, or hardware profile [7]. Detailed workload awareness refers to the use of query plans, execution metrics, and table statistics in addition to the high-level information [13].

3.5 Possible Unsafe Configurations

Unsafe configurations refer to configuration settings that may lead to harmful outcomes during tuning. Within the knob tuning literature, unsafe configurations are defined as configurations that degrade database performance [31,33]. However, with LLMs, this definition needs to be broadened to include cases of invalid settings, resource exhaustion, database crashes, and failed restarts. This highlights safe configuration generation as an important design dimension.

3.6 Requiring LLM Fine-tuning

Another factor to consider is whether the approach requires fine-tuning the LLM. This is important because fine-tuning can make the LLM more specialized for a given tuning task since the pre-trained LLM knowledge can have limitations [32]. However, it can often require expensive hardware resources (GPUs), large amounts of training data, and significant time, as evidenced by recent fine-tuned LLM-based tuning approaches [11]. This dimension is also treated as a binary in our taxonomy: a method either requires fine-tuning or does not.

3.7 Explainability

Explainability refers to the ability of a tuning method to explain why a given knob is selected or why a specific value is recommended. This dimension is important because explainability can increase user trust and adoption of automated decision-making systems. While interpretability of "black box" models has been a challenge in ML-based knob tuning [33], LLM hallucinations further increase the need for explainability [32]. This is especially relevant in real-world environments where DBAs may want a justification behind a recommendation before applying it to a production system.

3.8 Generalization

Generalization refers to the ability of a tuning system to generalize across different workloads or hardware environments. Pre-LLM knob tuning techniques often need to be retrained on unseen workloads or when the hardware environment changes [33]. This dimension is important because real-world scenarios often differ from experimental conditions. Pre-trained on a large volume of data, LLMs show strong generalization across tasks [3]. Therefore, it is important to test whether these LLM-based knob tuning techniques follow the same pattern when workloads or hardware environments change.

3.9 Tuning Objectives

Tuning objectives refer to the performance or system goals that a tuning method attempts to optimize. This dimension is important because different workloads and deployment settings may prioritize different outcomes. For OLAP workloads, the most common objective is query execution time, while for OLTP workloads, throughput is typically the primary goal [33]. Other objectives, such as reducing monetary cost or energy consumption, often need to be considered.

3.10 Tuning Mode

Tuning mode describes whether a method performs tuning in an offline or online setting. Offline tuning refers to performing tuning in isolated environments on a copy of the database [33]. Online tuning, on the other hand, is done in live production databases, where workloads are dynamic, and restarts can cause high cost to the business [31]. It is therefore important to distinguish between these two tuning modes as each requires different design choices.

4 LLM-Based Tuning Algorithms Selected for Evaluation

In this section, we discuss the six LLM-based tuning algorithms selected for our experimental evaluation: DB-BERT [23], GPTuner [13], LATuner [7], λ-Tune [10], RABBIT [18], and E2ETune [11]. We describe the selected algorithms by incorporating the taxonomy dimensions introduced in Section 3. Table 1 provides the complete classification across all dimensions, while the following descriptions focus on the main design choices.

4.1 DB-BERT

DB-BERT is one of the first LLM-based knob tuning techniques [23]. It utilizes natural-language tuning documents mined from the web as external knowledge sources and RL as the optimization mechanism beyond the LLM. The algorithm first divides the tuning documents into multiple snippets, which include implicit and explicit knob tuning hints. These hints are extracted, and DB-BERT prioritizes the most frequently mentioned hints during the iteration loop, where configurations are constructed and applied to the database. At each iteration, three types of decisions are made. First, it translates the extracted hints into concrete knob settings using a fine-tuned BERT language model. Second, it decides whether to adjust the suggested knob values by applying multiplicative factors, which helps introduce diversity. Third, it assigns a weight to each hint to represent its relative importance. After the hints are processed, they are applied to the database, and the workload is run to measure throughput and query execution time as tuning objectives depending on the workload type. DB-BERT then learns how to improve these three tasks by using performance as a reward signal for deep RL. Since the knobs are extracted from external knowledge sources, DB-BERT follows an adaptive knob selection strategy. It has no workload awareness because workload characteristics are not provided during hint translation.

Table 1: Features of the Evaluated LLM-based Database Knob Tuning Methods based on the Taxonomy Design Dimensions

		DB-BERT (Trummer, 2022)	GPTuner (Lao, 2024)	LATuner (Fan, 2024)	λ-Tune (Trummer, 2025)	Rabbit (Sun, 2025)	E2ETune (Huang, 2025)
Data source	External knowledge sources	✓	✓			✓	
	Historical tuning records					✓	✓
Optimization algorithm (beyond LLM)		RL	BO	BO		BO	
Knob selection	Static						✓
	Adaptive	✓	✓	✓	✓	✓	
Workload awareness	None	✓					
	High level			✓		✓	
	Detailed		✓		✓		✓
Possible unsafe configurations	Yes	✓	✓	✓	✓	✓	✓
	No						
Requires LLM fine-tuning	Yes	✓					✓
	No		✓	✓	✓	✓	
Explainability	Yes						
	No	✓	✓	✓	✓	✓	✓
Generalization	Hardware	✓	✓	✓	✓	✓	
	Workloads	✓	✓	✓	✓	✓	✓
Tuning objective	Throughput	✓	✓	✓		✓	✓
	Query execution time	✓	✓	✓	✓	✓	✓
	Monetary costs				✓		
	Energy consumption						
Tuning mode	Offline	✓	✓	✓	✓	✓	✓
	Online						

4.2 GPTuner

GPTuner is distinguished from DB-BERT by its direct use of the LLM to extract and structure tuning knowledge from external knowledge sources, which is then used to refine and prune the search space [13]. The algorithm first collects this tuning knowledge from DBMS manuals, online resources, and GPT-4-generated suggestions. Since these sources may contain noisy or contradictory information, GPTuner utilizes GPT-4 to determine whether a given hint is consistent with the target knob system view. The retained knowledge is then summarized and transformed into a structured JSON file containing suggested values, minimum values, maximum values, and special values. Next, GPTuner prompts the LLM to select knobs relevant to the tuning task by incorporating detailed workload characteristics such as workload type, hardware information, database type, and select query plans. This places GPTuner in the adaptive knob selection and detailed workload awareness categories of our taxonomy. Since all interactions with the LLM are prompt-based, it does not require fine-tuning. Once the search space is optimized, GPTuner uses BO as an optimization mechanism beyond the LLM through coarse-grained and fine-grained search. During each iteration, BO selects a configuration and runs the workload against the database to measure throughput or query execution time as the primary tuning objective.

4.3 LATuner

LATuner integrates an LLM into multiple stages of the knob tuning process. The algorithm initially prompts the LLM to select a set of relevant knobs for the tuning task, placing it in the adaptive knob selection category of our taxonomy. The prompt includes workload type and hardware information, which gives LATuner

high-level workload awareness. LATuner relies on the LLM's pre-trained knowledge rather than external knowledge sources or historical tuning records, and it does not require fine-tuning. LATuner then generates an initial set of configurations to warm-start BO as the main optimization algorithm beyond the LLM. During the main BO loop, LATuner balances exploration and exploitation through a Hybrid Candidate Configuration Sampler. For exploitation, the LLM recommends promising configurations based on workload type, hardware information, and previously tested configurations. For exploration, LATuner uses prior observations and L-BFGS-based random sampling to generate additional candidates. Finally, LATuner maintains both a GP surrogate and an LLM-based surrogate, with the Multi-Armed Bandit method selecting the best surrogate at each round. The chosen configuration is then applied to the database to measure throughput or query execution time as the primary tuning objective.

4.4 λ-Tune

λ-Tune is a unique case because it utilizes an LLM to generate entire configurations [10] and does not use a separate optimization algorithm like BO or RL. Similar to LATuner, λ-Tune mainly relies on the LLM's pre-trained knowledge and does not require fine-tuning. The algorithm first formulates a prompt using the hardware specifications, target DBMS, and input workload, which places it in the detailed workload awareness category of our taxonomy. To keep prompting cost-efficient, it uses integer linear programming (ILP) to select the most informative subset of the workload. This enables λ-Tune to incorporate token cost as a cost constraint, while query execution time remains the primary objective. λ-Tune then makes multiple LLM calls with different levels of randomness to generate diverse candidate configurations. Since the LLM is tasked with selecting knobs and values, the algorithm performs adaptive knob selection based on workload characteristics. To reduce the tuning time spent on poor configurations, it uses a round-based configuration selection strategy in which each round has a timeout. These timeouts increase geometrically, which helps reduce wasted work by avoiding repeated evaluation of poor configurations. While λ-Tune can also recommend indexes, the discussion here focuses on its knob tuning workflow.

4.5 RABBIT

RABBIT uses Retrieval-Augmented Generation to enhance LLMs' capabilities in knob tuning [18]. Unlike previously discussed techniques, RABBIT builds a knowledge base from both external knowledge sources and historical tuning records. It encodes external knowledge sources in a knowledge graph so that dependencies between knobs can be stored and retrieved more effectively. RABBIT then uses the LLM to generate a few initial configurations from the DBMS, tuning objective, hardware information, workload type, and table names, placing it in the high-level workload awareness category. It then looks for similar past tuning tasks and uses that experience to rank important knobs. After that,

RABBIT extends the knob set using dependency information from the knowledge graph, placing it in the adaptive knob selection category. RABBIT utilizes a multi-agent pruning strategy to reduce each knob's value range into a smaller set of promising candidates. Finally, it uses a two-stage optimization process involving BO as the optimization algorithm beyond the LLM. The first stage explores the pruned space with an LLM-based surrogate model, while the second stage expands the search with traditional BO. Each iteration runs the workload to measure throughput or query execution time, depending on the workload type.

4.6 E2ETune

E2ETune fine-tunes a Mistral-7B model [12] to directly generate knob configurations. The method consists of three stages: training data construction, LLM fine-tuning, and knob tuning. First, E2ETune generates {workload, suitable configuration} pairs, thereby using historical tuning records as a data source. For OLAP workloads, GPT-4-Turbo generates diverse SQL queries, while for OLTP workloads, diversity is introduced by varying transaction weights in OLTPBench [6]. For each workload, HEBO [5] generates a suitable configuration, and a surrogate model accelerates data collection. During fine-tuning, the LLM input consists of three components: workload statistics, query plans, and internal system metrics. This places E2ETune in the detailed workload awareness category. The output is a set of suitable knob configurations. Since these configurations are generated over a fixed set of knobs, E2ETune follows a static knob selection strategy. In total, 2,953 training examples are used to fine-tune the model across 10 benchmarks. In the knob tuning stage, the fine-tuned model generates 8 candidate configurations, which are ranked by the surrogate model. The selected configuration is applied to the database to measure throughput or query execution time. Although BO is utilized in the data collection phase, E2ETune does not rely on a separate optimization algorithm during the knob tuning stage.

5 Experimental Methodology

In this section, we first describe the differences in the evaluation models reported in the publications of the six LLM-based database knob tuning algorithms presented in Section 4. These differences motivate us to develop a uniform evaluation model to evaluate the algorithms fairly. We then present our model consisting of hardware, software, benchmarks, evaluation metrics, and dynamic parameters.

5.1 Differences in Prior Experimental Settings

Table 2 summarizes prior experimental settings, showing that LLM-based knob tuning studies differ significantly in their experimental models. Some studies evaluate their methods on PostgreSQL, while others use MySQL, and DBMS versions also vary. The benchmarks and scaling factors differ across the experiments and are not consistent. Specifically, in TPC-H settings, some studies use

Table 2: Experimental settings used in prior LLM-based knob tuning studies

		DB-BERT (Trummer, 2022)	GPTuner (Lao, 2024)	LATuner (Fan, 2024)	λ-Tune (Trummer, 2025)	Rabbit (Sun, 2025)	E2ETune (Huang, 2025)
DBMS System	DBMS System	PostgreSQL 13.2 MySQL 8.0	PostgreSQL 14.9 MySQL 8.0	MySQL 5.7	PostgreSQL 12.0 , MySQL 8.0	MySQL 8.0.	PostgreSQL 12.2
Knob scope	Candidate knobs	All numerical/Boolean knobs PostgreSQL: 232, MySQL: 266	Initial adjustable knobs PostgreSQL: 60, MySQL: 40	Initial adjustable knobs MySQL: 60	LLM-generated knob-value pairs	Initial adjustable knobs MySQL: 70	Manually knobs set PostgreSQL: 45
Benchmarks	OLTP	TPC-C: SF-20	TPC-C: SF-200	TPC-C: SF-200 YCSB: 20 GB Twitter: 20GB Wikipedia: 20GB SEATS: 20GB		TPC-C: SF-10,100,500 YCSB: SF-20,000 SEATS: SF-5	TPC-C: 5GB Smallbank: 20 GB YCSB: 4 GB Twitter: 0.4 GB Wikipedia: 4.4 GB
	OLAP	TPC-H: SF-1, SF-10	TPC-H: SF-1, SF-10, SF-50		TPC-H: SF-1, SF-10 TPC-DS: SF-1 JOB	TPC-H: SF-5	TPC-H 10 GB, JOB-7GB, SSB-13GB, SSB_flat 12GB, TPC-DS 2 GB
	Custom						StackOverflow, SSAG and AMPS (private Bytedace data)
TPC-H settings	Benchmarking tool	Custom python script	OLTPBench		Custom python script	OLTPBench	Custom python script
	Number of queries	22	22		22	22	Varies
	Number of Indexes	10	23		10	23	Unspecified
TPC-C settings	Benchmarking tool	OLTPBench	OLTPBench	OLTPBench		OLTPBench	OLTPBench
	Number of threads	Unspecified	Unspecified	16		32	Unspecified
Performance metric	OLAP	Total query execution time	95th %-tile Latency		Total query execution time	95th %-tile Latency	Delta of total query execution time
	OLTP	Throughput	Throughput	Throughput		Throughput	Delta of throughput
Compared to	ML based	DDPG++	DDPG++, GP, SMAC	GP, SMAC	UDO, LlamaTune, ParamTree	GP, DDPG, SMAC	SMAC, HEBO, CDBTune
	LLM-based system	Prior prototype by the same author	DB-BERT		DB-BERT, GPTuner	GPTuner	GPTuner, DB-BERT
Cost analysis	Monetary		Yes	No	Yes	Yes	150$ to use GPT4-Turbo to collect labels
	Time	Fine-tuning time				Yes	10 hours to fine-tune Mistral-7B
	Tokens		Yes	No	Yes	Yes	
Hardware platform	Experiments	8 vCPUs, 61 GB of RAM	24-core Intel Xeon E5-2676 110 GB of RAM 931 GB WD SSD	Standard D4lds v5 4 cores, 8GB RAM, 1TB Disk Size	EC2 p3.2xlarge 8 vCPUs, 61 GiB RAM	28-Core Intel Xeon 6330 256 GB RAM 960 GB SSD	Intel Xeon E5 2650 v4 (12 cores, 24 threads), 64GB RAM
	Fine-tuning	Tesla V100 GPU 16 GB VRAM					Intel Xeon Gold 5218 CPU, 256GB RAM, 4 NVIDIA GeForce RTX 3090 GPUs

a custom script to run the workloads, while others rely on benchmarking frameworks such as OLTPBench [6]. For OLAP workloads, some studies report total query execution time while others report latency-based metrics such as 95th percentile latency. For OLTP workloads, parameters such as runtime duration, terminal count, and arrival rate are not always consistent or fully reported. Due to the recency of LLM-based knob tuning research, most studies only compare their algorithm to a small subset of existing approaches.

Because of these differences, direct comparison of existing LLM-based knob tuning techniques is difficult. To address this limitation, we evaluate the selected methods under a unified experimental framework with consistent settings for hardware and software configurations, benchmark setup, evaluation metrics, and dynamic parameters, which are discussed in the following sections.

5.2 Hardware and Software Configuration

Our experiments were conducted on three identical Google Cloud n2d-standard-8 instances, each provisioned with 8 vCPUs, 32 GB RAM, and 300 GB SSD storage, running Ubuntu 22.04. PostgreSQL 14.19 was used as the DBMS for all evaluations, consistent with Table 2, where four out of the six algorithms use PostgreSQL.

5.3 Benchmarks

We evaluated the tuning algorithms using both analytical (OLAP) and transactional (OLTP) workloads. For OLAP evaluation, we used the TPC-H benchmark

as it was the most commonly used in prior work. All TPC-H queries were executed sequentially to measure total query execution time. For OLTP evaluation, we used the TPC-C benchmark as it is also the most frequently used benchmark in Table 2. The workloads were executed using OLTPBench [6], with each run lasting for 2 minutes, using 16 terminals and an unlimited arrival rate.

5.4 Evaluation Metrics

The following metrics were used to compare database performance across different dimensions.

Total query execution time: It refers to the total time in seconds taken to execute all the queries within a workload. Query execution time is a standard performance metric for OLAP workloads [32].

Throughput: Throughput is defined as the number of transactions the database can process per unit of time [32]. We use transactions per second (TPS) as a measure of throughput consistent with Table 2 and prior studies [28,27]. Since λ-Tune targets OLAP workloads, it is excluded from analysis.

Recommendation time: It is defined as the total wall-clock time required for a tuning method to complete the entire tuning process. This captures the practical cost of tuning.

Fine-tuning time: For DB-BERT and E2ETune, which rely on fine-tuned open-source LLMs, we report the time required to fine-tune the model. This metric captures the overall preparation costs.

Token usage: For techniques that rely on API calls to closed-source LLMs, we report token usage statistics. We will only report this metric for GPTuner, RABBIT, LATuner, and λ-Tune as only these algorithms use closed-source LLMs.

5.5 Dynamic Parameters

The evaluation metrics described above were collected under different conditions to evaluate the six algorithms. We varied the database size and workload size to measure how each algorithm behaved.

Impact of Database Size: To study the impact of database size, we increased the database size for both TPC-H [22] and TPC-C [21] while keeping all other system parameters constant. For TPC-H, we used scale factors 1 (1.6 GB), 5 (8 GB), 10 (16 GB), and 13 (21 GB). We executed 22 TPC-H queries for each scale factor to match the experimental settings summarized in the *TPC-H settings* row of Table 2. For TPC-C, we used scale factors 10 (1 GB), 50 (5 GB), 100 (10 GB), and 200 (20 GB).

Impact of Workload Size: To study workload size, we varied the number of TPC-H queries while keeping the database size constant. We executed workloads containing 22, 88, 154, 220, and 440 queries. We used 1.6 GB as the default database size because it was the most common scale factor in prior studies (see Table 2). For TPC-C, the query count is not varied because we configured OLTPBench to execute unlimited transactions within 2 minutes.

5.6 Implementation Details

To experimentally evaluate the selected LLM-based database knob tuning algorithms, we used the source code and prompts provided by the authors. Since not all algorithms used the same benchmarking tool, we made minimal modifications to ensure consistent workload execution across the methods. For TPC-H, we developed a custom Python script that runs the full workload and computes the total query execution time. For TPC-C, we used OLTPBench [6] and made minimal changes to LATuner, E2ETune, and DB-BERT.

Additionally, we followed each method's initial candidate knobs as shown in Table 2. This preserves each algorithm's original implementation, but it also means that performance differences may partly reflect differences in knob coverage. We use the default PostgreSQL configuration as the common baseline because our evaluation focuses on comparing automated LLM-based tuning methods rather than expert-guided tuning.

6 Experimental Results

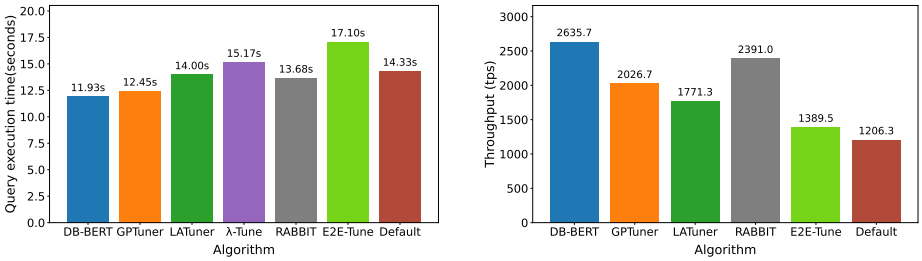
In this section, we present the results of our experimental evaluation. Section 6.1 discusses the overall results under the default experimental settings, while Sections 6.2 and 6.3 analyze the impact of the database size and workload size, respectively, on the algorithms' performance.

6.1 Overall Results

We first present the experimental results under the default dynamic parameter settings. For TPC-H, the database contains 1.6 GB of data, and the workload consists of 22 queries. For TPC-C, the database contains 10 GB of data, and the workload is executed for 2 minutes with an unlimited query arrival rate.

Figure 2 summarizes the overall results of the evaluated algorithms. For the TPC-H workload in Figure 2a, the bar labeled "Default" reports the query execution time under the default PostgreSQL knob configuration. DB-BERT achieves the largest improvement, reducing query execution time by 16.7% on average, followed by GPTuner with a 13.1% improvement. RABBIT and LATuner provide smaller gains of 4.5% and 2.3%, respectively. λ-Tune and E2ETune perform worse than the Default configuration, increasing execution time by 5.9% and 19.3%. These results suggest that LLM-based methods combined with iterative optimization are more robust than direct LLM-based configuration generation, while RABBIT's graph-based RAG provides only limited gains in this setting.

For the TPC-C, Figure 2b shows the highest throughput achieved by each tuning method. All techniques outperform the default PostgreSQL configuration. DB-BERT, RABBIT, GPTuner, and LATuner achieve the most significant throughput increases, with average improvements of 118.5%, 98.2%, 68.0%, and 46.8%, respectively. E2ETune, on the other hand, achieves a small average improvement of 15.2%. DB-BERT achieves the largest throughput, suggesting that



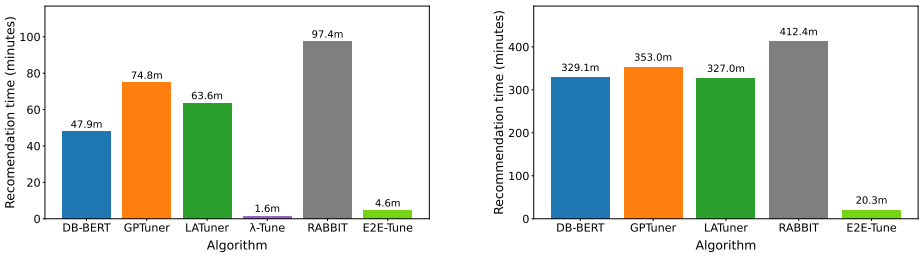
(a) TPC-H best total query execution time found

(b) TPC-C best throughput found

Fig. 2: Overall performance on TPC-H and TPC-C benchmarks

its extracted tuning hints, RL framework, and smaller knob set are likely contributing factors.

The recommendation times for both the TPC-C and TPC-H benchmarks are shown in Figure 3. GPTuner, LATuner, RABBIT, and DB-BERT exhibit significantly higher recommendation times than the other methods. All these algorithms iteratively explore the search space, with each iteration requiring execution of the benchmark workload to evaluate the candidate configuration. In contrast, E2ETune and λ-Tune generate configurations directly through the LLM without iterative benchmarking, resulting in lower recommendation times.



(a) TPC-H recommendation time

(b) TPC-C recommendation time

Fig. 3: Overall recommendation time on TPC-H and TPC-C benchmarks

For our experiments, only DB-BERT and E2ETune require fine-tuning of an open-source LLM. DB-BERT requires approximately 41 minutes to complete the fine-tuning stage, whereas E2ETune requires approximately 11 hours. Notably, E2ETune requires roughly 30 days to generate training data and fine-tune the model using four NVIDIA GeForce RTX 3090 GPUs [11]. Due to these constraints, we used the pre-trained model [17] released by the authors.

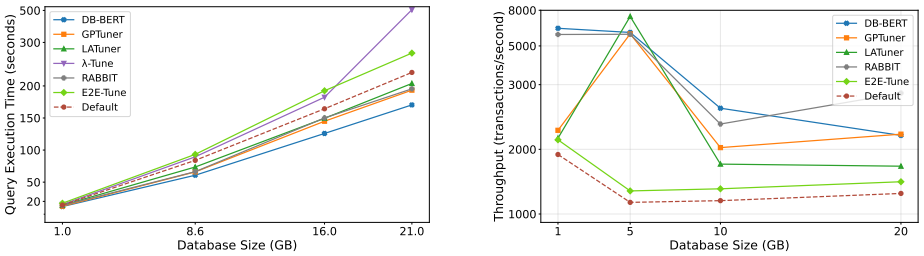
Table 3: Total token usage under default settings (in thousands)

Benchmark	GPTuner	LATuner	RABBIT	λ-Tune
TPC-H	16.2	684.9	5079.4	1.5
TPC-C	16.4	703.5	4895.6	–

Table 3 displays the total token usage. RABBIT incurs the largest token usage for both TPC-H and TPC-C benchmarks, with 5.1 million and 4.9 million, respectively, due to repeated API calls during a single iteration of BO. Similarly, LATuner uses LLM as a surrogate and makes API calls during the BO iteration loop, resulting in significant token usage. GPTuner, on the other hand, primarily utilizes the LLM before the BO step by reducing the number of knobs and pruning the range. λ-Tune, by prompting the LLM for full configurations, incurs the fewest tokens.

6.2 Impact of Database Size

As shown in Figure 4a, the total query execution time increases for all methods as the database size grows. The performance improvements are similar to the overall results with DB-BERT, GPTuner, RABBIT, and LATuner reducing the total query execution time on average by 22.7%, 14.7%, 11.4%, and 7.9%, respectively. E2ETune and λ-Tune, however, achieve worse performance compared to Default, increasing query execution time on average by 15.5% and 38.0%, respectively. Similar reasoning to the analysis in Section 6.1 applies here as well.

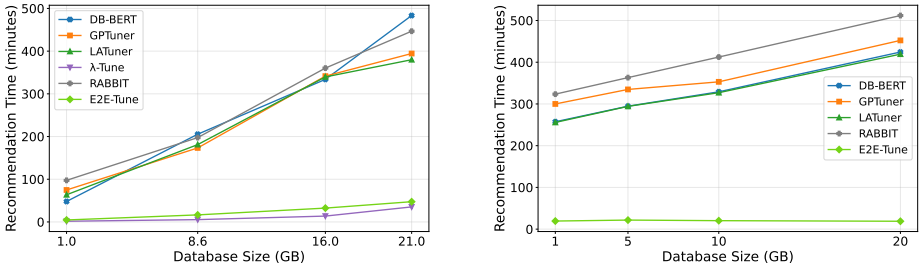


(a) TPC-H total query execution time vs database size

(b) TPC-C throughput vs database size

Fig. 4: Impact of database size on total query execution time and throughput

Figure 4b represents the throughput as we scale the database size. DB-BERT and RABBIT achieve the best average improvement in throughput over Default with 205.4% and 203.2%, followed by LATuner and GPTuner with 154.2% and 138.4%, respectively. E2ETune achieves only an average of 13.9% over Default. LATuner reaches the highest performance at 5 GB (scaling factor 50). However,



(a) TPC-H recommendation time vs database size

(b) TPC-C recommendation time vs database size

Fig. 5: Impact of database size on recommendation time

Table 4: Token usage across database sizes (in thousands)

Algorithm	**TPC-H**					**TPC-C**				
	1 GB	**8.6 GB**	**16 GB**	**21 GB**	**Average**	**1 GB**	**8.6 GB**	**16 GB**	**21 GB**	**Average**
GPTuner	16.2	20.1	15.0	16.2	16.9	22.7	13.4	16.4	19.0	17.9
LATuner	684.9	692.2	668.6	567.5	653.3	722.6	715.8	703.5	720.5	715.6
RABBIT	5079.4	2774.8	2974.2	2829.7	3414.5	5014.2	3499.0	4895.6	1838.3	3811.8
λ-Tune	1.5	1.5	1.7	1.4	1.5	–	–	–	–	–

as the database size increases further, throughput declines for all methods. RABBIT maintains stable performance across larger database sizes, suggesting that its graph-based representation of knob relationships allows it to adapt more effectively as workload complexity increases. DB-BERT's performance at scaling factor 20 diminishes, suggesting that simply translating tuning hints becomes less effective as the database size grows. E2ETune underperforms, suggesting its limitations when deployed on hardware different from its training environment.

Figure 5 shows how the recommendation time changes as the database size increases for both TPC-H and TPC-C benchmarks. For the TPC-H benchmark, GPTuner, LATuner, DB-BERT, and RABBIT exhibit significantly higher recommendation times compared to the other methods. As the database size grows, the recommendation time grows linearly. This is primarily because these methods evaluate many candidate configurations over repeated iterations, and each iteration requires running the benchmark workload on the database. As the database size increases, each benchmark run takes longer due to higher query execution time (Figure 4a), which leads to longer recommendation times. In contrast, λ-Tune and E2ETune generate configurations directly through LLM inference, resulting in substantially lower recommendation times. A similar trend is observed for the TPC-C benchmark.

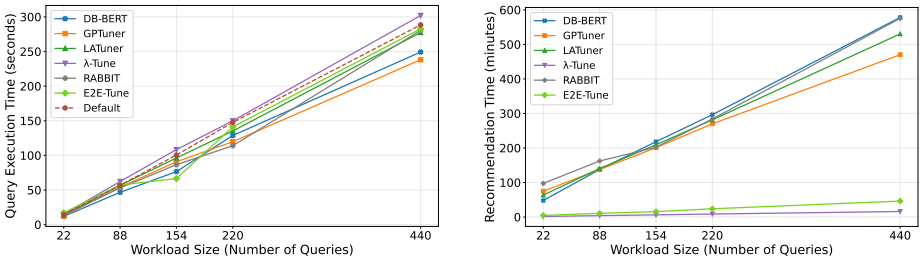
Table 4 displays the token usage for both TPC-H and TPC-C benchmarks. The token usage remains mostly similar as the database size grows. The variance is largely due to the variability in generated responses from the LLM, which may vary in response length.

Table 5: Token usage across different TPC-H workload sizes (in thousands)

Algorithm	**22Q**	**88Q**	**154Q**	**220Q**	**440Q**	**Avg**
GPTuner	16.2	24.6	21.1	21.6	15.7	19.8
LATuner	684.9	707.5	711.7	683.7	704.9	698.5
RABBIT	5079.4	2186.9	3515.9	3256.7	3569.0	3521.6
λ-Tune	1.5	1.5	2.3	1.5	2.0	1.8

6.3 Impact of Workload Size

Figure 6a shows that the total query execution time increases nearly linearly as the workload size grows, which is expected since more queries must be processed. DB-BERT, GPTuner, and RABBIT achieve double-digit reduction in total query execution time across different workload sizes with 16.9%, 12.8%, and 10.1% average improvement over Default, respectively. E2ETune and LATuner achieve a smaller improvement of 4.0% and 3.7% respectively, while λ-Tune performs the worst by increasing the total query execution time by 5.9%. This shows that configurations generated without iterative optimization are less effective as workload size increases.



(a) Impact of workload size on TPC-H total query execution time

(b) Impact of workload size on recommendation time

Fig. 6: Impact of workload size on performance metrics for the TPC-H workload.

Figure 6b shows the recommendation time for the TPC-H workload as the workload size increases. As the number of queries increases, so does the total execution time, which takes the majority of the recommendation time for the algorithms relying on iteration. The recommendation time for E2ETune and λ-Tune remains similar even as the query size increases. Table 5 lists token usage as the query size grows. Similar to other token usage analyses, RABBIT and LATuner consume the largest number of tokens in the range of hundreds of thousands to millions, while GPTuner and λ-Tune are in the range of thousands.

7 Research Opportunities

Based on our experimental results and the feature comparison summarized in Table 1, we identify several areas for future research.

Hallucination Mitigation: A primary concern with the use of LLMs is hallucinations [32]. While the results in sections 6.1- 6.3 show performance gains, this does not reveal whether the recommended configurations are based on grounded tuning knowledge or whether some knob names, values, ranges, dependencies, or rationales were hallucinated by the LLM. Although existing studies partially mitigate hallucinations, these are typically implicit rather than a design objective. Future work needs to identify and categorize different types of hallucinations in LLM-based knob tuning and develop mitigation strategies.

Configuration Safety: During our experiments, we observed that some configurations recommended crashed the database or resulted in failed restarts. In these cases, existing techniques typically recovered the database and applied a performance penalty, so these failures are not always explicitly visible in performance plots. As reflected in Table 1, existing techniques do not provide safety layers to prevent such scenarios. This can be important, especially in online tuning environments where unsafe configurations can affect production systems.

Online Tuning: As reported in Table 1, current techniques mostly focus on offline tuning. Some traditional ML-based techniques have been explored in the context of online tuning [31]. Extending LLM-based methods to this setting represents a promising direction for future research. Applications such as e-commerce systems during flash sales and real-time platforms such as ride-sharing or payment systems often experience rapidly changing workload patterns that limit the effectiveness of static offline tuning.

Multi-objective Tuning: Most existing techniques discussed in this paper optimize for query execution time or throughput, as shown in Table 1. However, RABBIT and LATuner show that performance can often come with a cost of substantially more tokens (see Table 3). Real-world deployments can often introduce additional objectives such as cloud resource cost, energy consumption, or CPU utilization in shared environments. Therefore, multi-objective configuration tuning presents an interesting research opportunity.

Fine-tuned LLMs: Fine-tuned models, such as E2ETune, offer a promising direction because they can significantly reduce recommendation time once training is complete. This is supported by our experiments where E2ETune showed lower recommendation times (see Figure 3). However, its weaker performance (see Figure 2) may indicate limitations when deployed on hardware different

from its training environment. Future work should explore methods for improving hardware-aware generalization while reducing the cost of generating training data.

Explainability: As shown in Table 1, none of the evaluated algorithms currently provide explanations as to why specific knobs or values were selected. Meaningful explainability would increase trust in automated tuning systems. Future work should explore methods for generating justifications for knob selection and configuration recommendations.

8 Conclusion

In this paper, we presented a taxonomy, a unified experimental comparison of six existing LLM-based database knob tuning techniques, and discussed research opportunities in this area. The experimental studies conducted on PostgreSQL using the TPC-C and TPC-H benchmarks showed that the methods that combine LLMs with iterative optimization, particularly BO and RL-based approaches, achieved the strongest and most consistent performance improvements. However, these gains came with substantially higher recommendation times, while direct LLM-based generation methods were faster but generally less robust under scaling. Our results also show that LLM-based tuning should be evaluated not only by throughput and query execution time, but also by additional dimensions such as recommendation time, fine-tuning cost, and token usage.

Our work mainly focuses on overall algorithm performance and does not perform a detailed knob-level ablation study. In future work, we aim to analyze whether omitted knobs or missing knob categories contribute to weaker performance. Since most algorithms perform adaptive knob selection, they may target different knobs for the same tuning task. Future work could also compare LLM-generated configurations with DBA recommendations and analyze prompt sensitivity.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Pgtune. `https://pgtune.leopard.in.ua/`
2. Ansel, J., Kamil, S., Veeramachaneni, K., et al.: Opentuner: An extensible framework for program autotuning. In: Proceedings of the 23rd international conference on Parallel architectures and compilation. pp. 303–316 (2014)
3. Brown, T.B., et al.: Language models are few-shot learners. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20 (2020)
4. Cai, B., Liu, Y., Zhang, C., et al.: Hunter: An online cloud database hybrid tuning system for personalized requirements. In: Proceedings of the 2022 International Conference on Management of Data. p. 646–659. SIGMOD '22 (2022)

5. Cowen-Rivers, A.I., Lyu, W., et al.: Hebo: Pushing the limits of sample-efficient hyper-parameter optimisation. J. Artif. Int. Res. (2022)
6. Difallah, D.E., Pavlo, A., Curino, C., Cudré-Mauroux, P.: Oltp-bench: An extensible testbed for benchmarking relational databases. PVLDB pp. 277–288 (2013)
7. Fan, C., Pan, Z., Sun, W., Yang, C., Chen, W.N.: Latuner: An llm-enhanced database tuning system based on adaptive surrogate model. In: Machine Learning and Knowledge Discovery in Databases. p. 372–388 (2024)
8. Fan, J., Gu, Z., Zhang, S., et al.: Combining small language models and large language models for zero-shot nl2sql. Proc. VLDB Endow. p. 2750–2763 (2024)
9. Frazier, P.I.: A tutorial on bayesian optimization. arXiv preprint (2018), https://arxiv.org/abs/1807.02811
10. Giannakouris, V., Trummer, I.: -tune: Harnessing large language models for automated database system tuning. Proc. ACM Manag. Data (2025)
11. Huang, X., Li, H., Zhang, J., Zhao, X., Yao, Z., Li, Y., Zhang, T., Chen, J., Chen, H., Li, C.: E2etune: End-to-end knob tuning via fine-tuned generative language model. Proc. VLDB Endow. p. 5540–5554 (2026)
12. Jiang, A.Q., Sablayrolles, A., Mensch, A., et al.: Mistral 7b. arXiv preprint (2023), https://arxiv.org/abs/2310.06825
13. Lao, J., Wang, Y., Li, Y., Wang, J., Zhang, Y., Cheng, Z., Chen, W., Tang, M., Wang, J.: Gptuner: A manual-reading database tuning system via gpt-guided bayesian optimization. Proc. VLDB Endow. p. 1939–1952 (2024)
14. Lyu, X., Zhang, J., Zheng, Y., Li, G., Lin, C.: Llm4ia: Index advising via large language models. In: Proceedings of the 34th ACM International Conference on Information and Knowledge Management. CIKM '25 (2025)
15. OpenAI, Achiam, J., Adler, S., et al.: Gpt-4 technical report (2024), https://arxiv.org/abs/2303.08774
16. Ouyang, L., Wu, J., Jiang, X., et al.: Training language models to follow instructions with human feedback. NIPS '22 (2022)
17. springhxm: E2etune. https://huggingface.co/springhxm/E2ETune (2024), fine-tuned version of mistralai/Mistral-7B-Instruct-v0.2
18. Sun, W., Pan, Z., Hu, Z., Liu, Y., Yang, C., Zhang, R., Zhou, X.: Rabbit: Retrieval-augmented generation enables better automatic database knob tuning. In: 2025 IEEE 41st International Conference on Data Engineering. pp. 3807–3820 (2025)
19. Sun, Z., Zhou, X., Li, G., Yu, X., Feng, J., Zhang, Y.: R-bot: An llm-based query rewrite system. Proc. VLDB Endow. p. 5031–5044 (2025)
20. Tan, J., Zhang, T., Li, F., et al.: ibtune: individualized buffer tuning for large-scale cloud databases. Proc. VLDB Endow. p. 1221–1234 (2019)
21. Transaction Processing Performance Council: TPC-C Homepage. https://www.tpc.org/tpcc/default5.asp
22. Transaction Processing Performance Council: TPC-H Homepage. https://www.tpc.org/tpch/
23. Trummer, I.: Db-bert: A database tuning tool that "reads the manual". In: Proceedings of the 2022 International Conference on Management of Data. p. 190–203. SIGMOD '22 (2022)
24. Trummer, I.: Can large language models predict data correlations from column names? Proc. VLDB Endow. p. 4310–4323 (2023)
25. Van Aken, D., Yang, D., Brillard, S., et al.: An inquiry into machine learning-based automatic configuration tuning services on real-world database management systems. Proc. VLDB Endow. p. 1241–1253 (2021)

26. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 6000–6010 (2017)
27. Zhang, B., Van Aken, D., Wang, J., et al.: A demonstration of the ottertune automatic database management system tuning service. Proc. VLDB Endow. p. 1910–1913 (2018)
28. Zhang, J., Liu, Y., Zhou, K., et al.: An end-to-end automatic cloud database tuning system using deep reinforcement learning. In: Proceedings of the 2019 International Conference on Management of Data. p. 415–432. SIGMOD '19 (2019)
29. Zhang, X., Chang, Z., Li, Y., Wu, H., Tan, J., Li, F., Cui, B.: Facilitating database tuning with hyper-parameter optimization: a comprehensive experimental evaluation. Proc. VLDB Endow. p. 1808–1821 (2022)
30. Zhang, X., Wu, H., Chang, Z., et al.: Restune: Resource oriented tuning boosted by meta-learning for cloud databases. In: Proceedings of the 2021 International Conference on Management of Data. p. 2102–2114 (2021)
31. Zhang, X., Wu, H., Li, Y., Tan, J., Li, F., Cui, B.: Towards dynamic and safe configuration tuning for cloud databases. In: Proceedings of the 2022 International Conference on Management of Data. p. 631–645. SIGMOD '22 (2022)
32. Zhao, W.X., Zhou, K., Li, J., et al.: A survey of large language models (2025), https://arxiv.org/abs/2303.18223
33. Zhao, X., Zhou, X., Li, G.: Automatic database knob tuning: A survey. IEEE Transactions on Knowledge and Data Engineering pp. 12470–12490 (2023)
34. Zhou, X., Li, G., Sun, Z., et al.: D-bot: Database diagnosis system using large language models. Proc. VLDB Endow. p. 2514–2527 (Jun 2024)

An Overview of NewSQL Databases: CockroachDB, TiDB, OceanBase, YugabyteDB and VoltDB

Daniela Neves[1] [0009-0001-1362-1020] and Jorge Bernardino[1] [0000-0001-9660-2011]

[1] Polytechnic University of Coimbra, Rua da Misericórdia, Lagar dos Cortiços, São Martinho do Bispo, 3045-093 Coimbra, Portuga
a21220120@isec.pt, jorge@isec.pt

Abstract. As NewSQL databases evolve to offer the scalability of NoSQL and the transaction guarantees of relational systems, this paper explores five leading implementations: CockroachDB, TiDB, OceanBase, YugabyteDB, and VoltDB. These systems were selected based on their high ranking in the 2026 DB-Engines Index and their technical diversity. The research methodology includes a detailed comparative analysis of their architectural styles, replication methods, partitioning strategies, and transaction models. The findings reveal that, although all five databases support horizontal scaling and ACID compliance, they use different consensus mechanisms, such as Raft, Paxos, and K-safety, to manage data integrity. Each database has unique strengths and weaknesses. For example, TiDB supports hybrid transactional/analytical processing (HTAP), while VoltDB focuses on low-latency in-memory transactions. We conclude that the different architectural designs of NewSQL databases require careful alignment with specific workload needs.

Keywords: NewSQL databases, database comparison, database management systems.

1 Introduction

The evolution of database management systems has shifted from traditional, monolithic, relational systems to horizontally scalable, NoSQL solutions. These solutions often compromise ACID (atomicity, consistency, isolation, durability) guarantees to improve performance. NewSQL databases have emerged to address this issue, offering the horizontal scalability of NoSQL alongside the strict transactional integrity of traditional SQL systems [1]. The growing demand for modern, globally distributed applications that require high availability without sacrificing data consistency highlights the significance of this development [2].

Researchers are taking a variety of approaches to evaluate these systems. Some researchers have provided extensive NewSQL feature classifications, while others have evaluated the

performance of specific implementations, including TiDB, YugabyteDB, VoltDB, and CockroachDB. A notable point of divergence in the field concerns the adequacy of current benchmarking tools. Some researchers argue that traditional online transactional processing (OLTP) benchmarks do not adequately capture the complexities and nuances of modern distributed transactional environments [3] [4].

This work aims to provide a comprehensive architectural overview and comparative analysis of the five most prominent NewSQL databases: CockroachDB, TiDB, OceanBase, YugabyteDB, and VoltDB. These databases were selected based on their projected industry popularity and technical diversity in 2026 [5]. By examining their distinct partitioning strategies, transaction models, and consensus mechanisms (e.g., Raft, Paxos, and K-safety), the study aims to establish a clear framework for their application. The main conclusion is that although these databases share the same core objectives of scalability and ACID compliance, their diverse architectural designs result in significant performance trade-offs. These designs range from hybrid transactional/analytical processing (HTAP) to low-latency in-memory execution and require careful alignment with specific workload requirements.

The remainder of this work is structured as follows. Section 2 presents related work on NewSQL database evaluation. Section 3 provides an analysis of the five NewSQL databases, highlighting their primary advantages and limitations. Section 4 compares the main characteristics of the five databases. Finally, Section 5 presents the conclusions and future work.

2 Related Work

This section presents some related work that evaluates NewSQL databases perfor-mance.

Almassabi et al. [2], presents a classification of leading NewSQL databases and compares their main features. The work is useful to understanding the diversity of systems in the NewSQL landscape, although its focus is more on feature classification than on experimental benchmarking.

Huang et al. [3], evaluated TiDB using CH-benCHmark, whose OLTP part corre-sponds to the TPC-C benchmark. The paper reports experimental details such as the number of servers, memory, CPU resources and benchmarking metrics including performance and latency. The results indicate that TiDB can support both OLTP and OLAP workloads with limited interfaces between them. However, the paper does not refer to any limitations which mainly appear in the evaluation section, such as limited comparisons with the other systems and higher replications delay when the workload becomes heavier.

Watanabe et al. [6], experiments on YugabyteDB performed with the TPC-C benchmark. The paper also describes the experimental setup, including a 10-node cluster, hardware resources, 10 warehouses, a dataset of 19.8 GB, and performance measurement in TPM-C. The results suggest that the proxy-based approach can improve transaction performance, mainly when the required data is already in cache. However, the paper also indicates that

performance can be lower without cached data, and the proxy may become a single point of failure.

Schumacher [7], evaluated VoltDB using PyTPCC, a modified version of the TPC-C benchmark. The work reports the experimental setup, including AWS-based server instances, CPU and memory resources, 300 simulated clients, and TPMC as the main performance metric. It also analyses the impact of horizontal scaling, vertical scaling, memory size, replication factor, partitioning strategy, and number of clients. The results show that VoltDB performs better when the transactions access only a small portion of the data, which suggests that good portioning is important for a good achieving better performance. The study also shows that VoltDB remained consistent in all experiments. However, the paper does not clearly discuss its limitations.

Håkansson et al. [8], evaluated CockroachDB in a cloud-native environment deployed on Kubernetes. The study analyzed performance, throughput, horizontal and vertical scaling, version hot-swapping, and node disruptions, and used the TPC-C benchmark for performance measurement. The results show that CockroachDB scales easily in both horizontal and vertical directions, maintains stable performance under manageable workloads and remains robust during version changes and node failures. However, the paper also mentions some limitations related to the shared test environment, limited access to the setup and the possibility that the deployment was not fully optimized.

According to Qu et al. [9], the current benchmarks are adequate for evaluating distributed transactional databases. The work shows that benchmark selection should consider the characteristics of modern distributed systems and not only traditional OLTP evaluation, since no single benchmark fully captures all relevant distributed system aspects.

In Difallah et al. [10], present OLTP-Bench, an extensible benchmarking testbed for relational databases. The framework was designed to improve repeatability and comparability across DBMS experiments and supports multiple workloads for transactional benchmarking. It also provides mechanisms for workload configuration and performance measurements across different database systems. Its current continuation is BenchBase [11] which is described by the official project as the modernized version of OLTP-Bench.

Unlike previous studies, which often focus on a single database implementation or general feature classification, this paper provides a comprehensive, comparative architectural analysis of five distinct NewSQL systems: CockroachDB, TiDB, OceanBase, YugabyteDB, and VoltDB. These systems were selected based on their top-tier popularity in the industry, according to the 2026 rankings [5]. While previous research often treats these systems in isolation, our work fills this gap by directly comparing their partitioning strategies, consensus mechanisms (Raft, Paxos, and K-safety), and transaction models within a unified framework. Furthermore, this study offers a more holistic view of the performance trade-offs inherent in different NewSQL architectural designs.

3 New SQL Databases Overview

This section describes the five NewSQL databases: CockroachDB, TiDB, OceanBase, YugabyteDB, and VoltDB. These databases were selected as the five most popular NewSQL databases according to the DB-Engines Ranking 2026 [5].

3.1 CockroachDB

CockroachDB is a cloud-native, distributed SQL database created in 2014 that is fully PostgreSQL-compatible. It was designed to be fault-tolerant, providing high availability and resilience across a variety of deployment environments, including physical servers, virtual machines, and cloud platforms. The system supports horizontal scalability, by adding new nodes to the cluster, as well as vertical scalability, by increasing resources such as memory on existing nodes. CockroachDB ensures ACID-compliant transactions across distributed nodes, features automatic data replication and does not require manual intervention after node failures. The consistency model is maintained through the Raft consensus algorithm, enabling reliable coordination and data integrity across the cluster.
CockroachDB's architecture is designed in layers to make it easier to manage **Fig. 1** shows a diagram of its architecture [12].

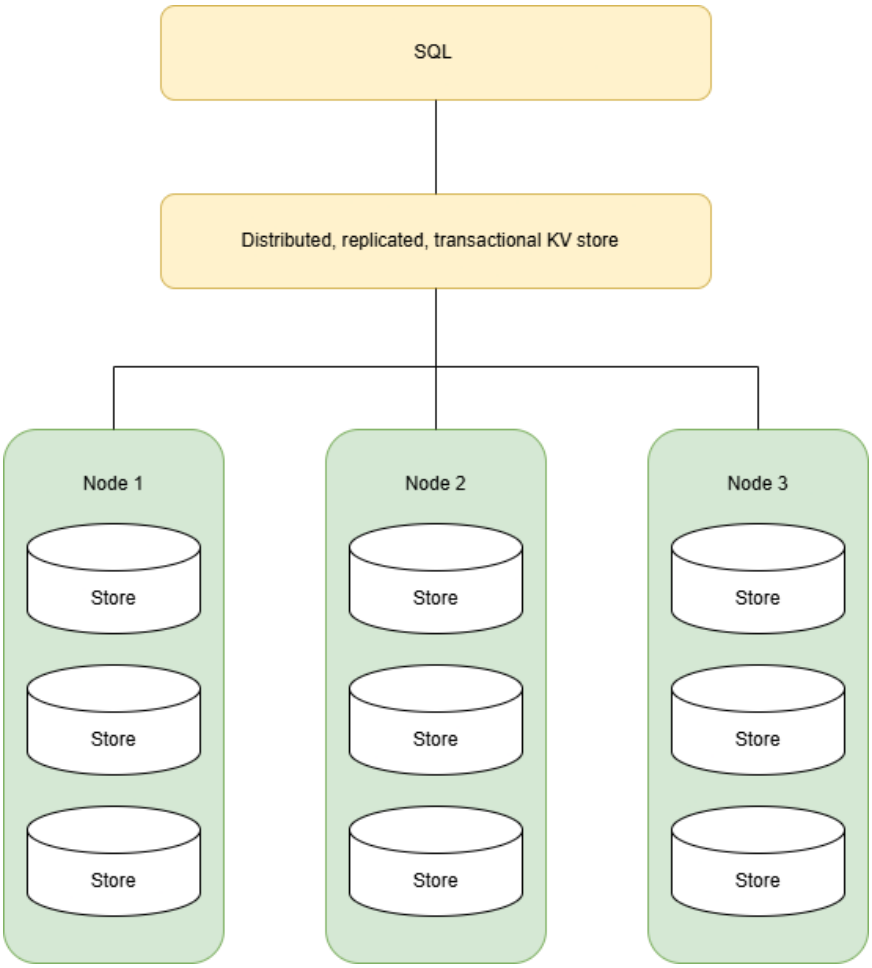


Fig. 1. CockroachDB architecture diagram.

CoackroachDB is divided into layers, and each one has the following functionalities:

- **SQL:** This is the highest layer of architecture and is the main interface between client and the database. It receives SQL queries, parses them and translates them into lower-level operations that can be executed by the underlying key-value system;
- **Distributed, replicated, transactional KV store:** This block represents the core of the CockroachDB. At this level, data is handled as key-value pairs rather than as relational

tables. It combines transaction management, data distribution and replication across the cluster, allowing the system to provide ACID transactions, strong consistency and fault tolerance in a distributed environment;
- **Nodes:** These blocks represent the nodes of the CockroachDB cluster;
- **Stores:** Each node contains one or more stores, which are the local storage units where the database reads and writes data on disk.

The main advantages of CockroachDB are the following:

- High availability/fault tolerance: data is replicated on multiple nodes, and the system can continue after node failures with little/no manual intervention;
- Strong consistency and ACID transactions in a distributed setup (the transaction layer is made for ACID semantics);
- Uses Raft for replication/consensus, with leader election for ranges (partitions), which helps reliability;
- Horizontal scaling: we can add nodes and the database rebalances data automatically.

The main limitations are the following:

- Distributed SQL has latency trade-offs: if data or queries need to touch many nodes (or regions), latency can increase (network hops + consensus);
- Some schema changes have limitations compared to “classic” single-node databases (for example, CockroachDB does not support schema changes in-side explicit transactions with full atomicity guarantees);
- Licensing is not OSI open-source (source-available, with multiple licenses). This can be a disadvantage if your project requires strict open-source licensing.

3.2 TiDB

TiDB is a distributed SQL database created in 2015 and is compatible with MySQL. It was designed to run in cloud environments and to support horizontal scaling. The architecture is divided into a processing layer and a storage layer. This separation allows the cluster capacity to be increased by adding more nodes to the storage layer or the processing layer. The replication and strong consistency in the transactional layer are guaranteed by the Raft algorithm, where each data partition has multiple replicas and a leader responsible for coordinating writes to the database. If this leader fails, a new one is automatically elected, which ensures high availability and automatic recovery of the service. TiDB supports OLTP workloads and, with TiFlash, the analytical component supports HTAP, i.e., the combination of transactions and analytics in (near) real time in the same system, which reduces the need for heavy ETL processes. TiDB supports ACID transactions for OLTP workloads. TiDB architecture [13] is represented by three main components: the TiDB cluster, the PD clusters and the storage clusters as demonstrated in **Fig. 2**.

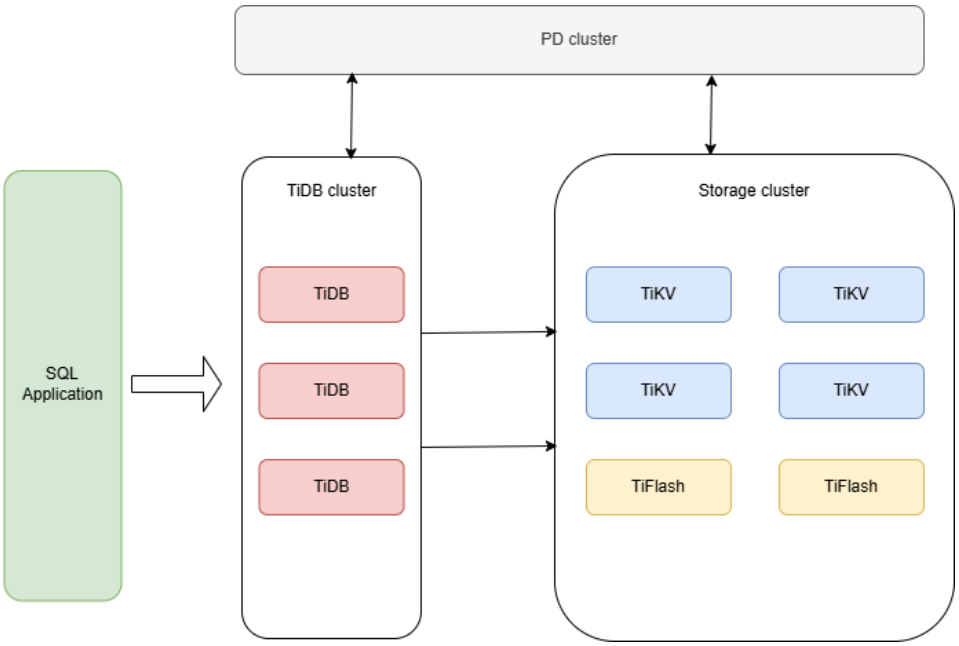


Fig. 2. TiDB architecture diagram

TiDB has the following components:

- **SQL Application:** This is the entry point of the system. The application communicates with TiDB through the MySQL protocol, which makes the system compatible with MySQL tools and applications;
- **TiDB cluster:** This block represents the TiDB servers, which form the SQL and computing layer of the system. They receive and process SQL queries, optimize execution plans, and send read and write requests to the storage layer. TiDB servers are stateless, which makes horizontal scaling easier;
- **PD cluster**: This block represents the Placement Driver (PD), which manages cluster metadata and scheduling. It is responsible for tasks such as metadata management and global timestamp allocation, acting as the coordination component of the cluster.
- **Storage cluster:** This block represents the distributed storage layer of TiDB. It contains the storage engines that keep and serve the data used by the system;
- **TiKV:** is the transactional storage engine of TiDB. It stores data in a distributed key-value format, provides ACID-compliant transactional APIs, and uses Raft replication to ensure consistency and high availability;

- **TiFlash:** is the analytical storage engine of TiDB. It stores data in columnar format and is designed to accelerate analytical queries, which allows TiDB to support HTAP workloads.

The main advantages of TiDB are the following:

- Good for HTAP: OLTP plus analytics with TiKV (row) and TiFlash (column);
- Strong consistency plus replication using the Raft in the storage layer;
- Horizontal scaling and high availability by design (distributed components).

The main limitations are the following:

- More operational complexity than a single-node SQL database (cluster components, etc.);
- Strong consistency with Raft can add overhead/latency compared to weaker consistency systems.

3.3 OceanBase

OceanBase is a distributed SQL database and a high-availability RDBMS, created in 2010. It was designed for business and cloud scenarios, and it can scale horizontally by adding new nodes to the cluster. It supports MySQL compatibility and an Oracle-compatible mode (configured per tenant), which helps when migrating existing applications.
For data reliability, OceanBase replicates each partition across multiple replicas and keeps strong consistency using a Paxos-based consensus protocol (Multi-Paxos). Each replication group has a leader that coordinates write, if the leader fails, a new leader is elected automatically, which allows failover and recovery without manual intervention. In practice, OceanBase is mainly used for OLTP workloads, and it can also support mixed workloads depending on the deployment and configuration. OceanBase supports ACID trans-actions for OLTP workloads.
The OceanBase architecture [14] is represented as distributed cluster of OBServer nodes, where data is divided into partitions and replicated across nodes using a Paxos mechanism to ensure high availability. In **Fig. 3** it demonstrates how OceanBase works.

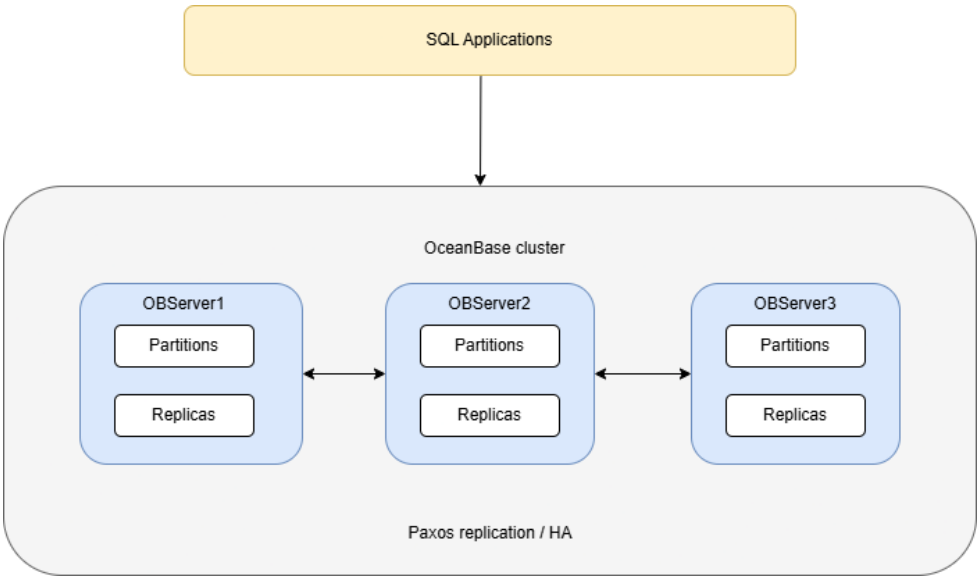


Fig. 3. OceanBase architecture diagram

OceanBase has the following functionalities:

- **SQL Applications:** This block represents the client applications that send SQL requests to the OceanBase cluster. In practice, OceanBase is designed to receive database requests from external applications and process them in a distributed environment;
- **OceanBase cluster:** This block represents the main database cluster. Ocean-Base adopts a shared-nothing architecture, in which multiple nodes cooperate to process queries, manage transactions, and store data in a distributed way;
- **OBServer's:** These blocks represent the OceanBase server nodes. Each server runs the observer process and is responsible for both computing and storage functions within the cluster. This means that each node can participate in SQL execution, transaction processing, and local data storage;
- **Partitions:** This block represents the logical division of data inside the cluster. Ocean-Base stores user data in partitions, which allows the system to distribute data and workload across multiple nodes and improve scalability;
- **Replicas:** This block represents the multiple copies of each partition stored on different nodes. OceanBase keeps several replicas of each partition to support fault tolerance and maintain data availability if a node fails;
- **Paxos** replication: This part of the diagram represents the replication mechanism used by OceanBase. The replicas of each partition form an independent Paxos replication group, in which one replica acts as the leader and the others act as followers. This

mechanism provides high availability, synchronization between replicas, and fault tolerance at the partition level.

The main advantages of OceanBase are the following:

- High availability at partition level with a leader/follower's model using Paxos;
- Strong consistency with Paxos replication groups;
- MySQL mode and Oracle mode, helpful for migration.

The main limitations are the following:

- Distributed design means more planning (partitions, zones, resources per tenant);
- Paxos/strong consistency has cost/overhead (trade-off of synchronous replication).

3.4 YugabyteDB

YugabyteDB is a high-performance distributed SQL database and was created in 2017. It was developed for cloud environments, and it can scale horizontally by adding new nodes to the cluster. It provides a SQL API called YSQL (PostgreSQL-compatible) and YCQL.
Replication and consistency are handled with the Raft consensus algorithm: each data partition has several replicas and one leader that coordinates the writes. If the leader fails, a new leader is elected automatically, which helps the system to keep running. YugabyteDB transactions are ACID compliant (for example in YSQL).
Yugabyte architecture [15] is shown as a distributed cluster with master nodes and tablets server, where data is divided into tablets and replicated across nodes using Raft.
The way in which YugabyteDB functions is illustrated in **Fig. 4**.

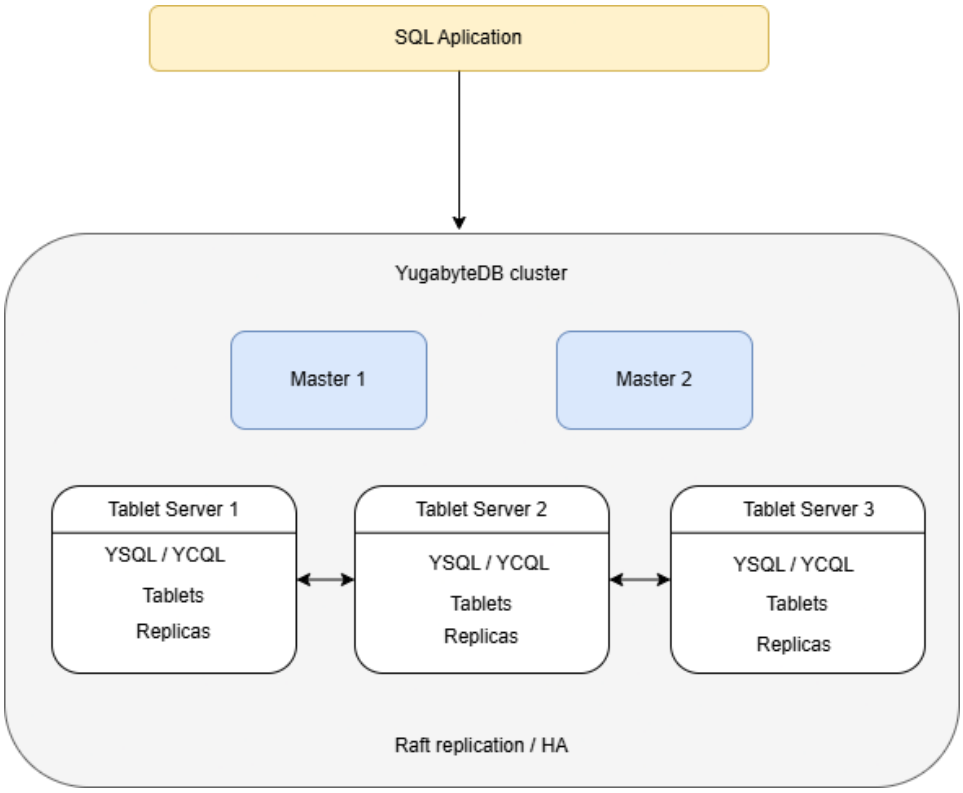


Fig. 4. YugabyteDB architecture diagram

YugabyteDB is composed of the following components:

- **SQL Applications:** This part represents the client applications that send requests to the YugabyteDB cluster;
- **YugabyteDB cluster:** This block represents the whole database cluster, where different nodes work together to process queries and store data;
- **Master nodes:** These nodes manage the cluster metadata and help organize the placement of data in the system;
- **Tablet Servers:** These nodes handle reading and writing requests and store the distributed data;
- **YSQL / YCQL:** These are the main interfaces supported by YugabyteDB; YSQL is PostgreSQL-compatible, while YCQL follows the Cassandra query model;
- **Tablets:** These are the data partitions used to distribute data across the cluster;

- **Replicas:** These are copies of the tablets stored on different nodes to improve availability and fault tolerance;
- **Raft replication / HA:** This part represents the Raft consensus mechanism used to keep replicas consistent and to provide high availability.

The main advantages of YugabyteDB are the following:

- ACID transactions (especially in YSQL) and support distributed transactions.
- Raft replication with leader election for high availability.
- Horizontal scaling with automatic sharding across nodes.

The main limitations are the following:

- In distributed SQL, performance depends more on data distribution and query design (can require more tuning than a single PostgresSQL);
- Strong consistency plus replication can add write overhead (again, typical trade-off).

3.5 VoltDB (Community Edition)

VoltDB Community Edition is a distributed in-memory NewSQL RDBMS, with initial release in 2010, and the Community Edition is open-source under AGPL.
It focuses on OLTP workloads with low latency. It can scale horizontally by partitioning data and distributing the work across nodes, so the cluster can grow by adding more machines.
For availability, VoltDB uses replication (K-safety) so the system can continue if a node fails and it can recover by restoring/rejoining replicas. VoltDB is a fully AC-ID-compliant transactional database (CE includes the core transactional features).
The VoltDB architecture [16] is displayed as a distributed cluster where the data is di-vided into partitions and replicated across the nodes using K-Safety. In the **Fig. 5.** VoltDB architecture diagram demonstrates how VoltDB works:

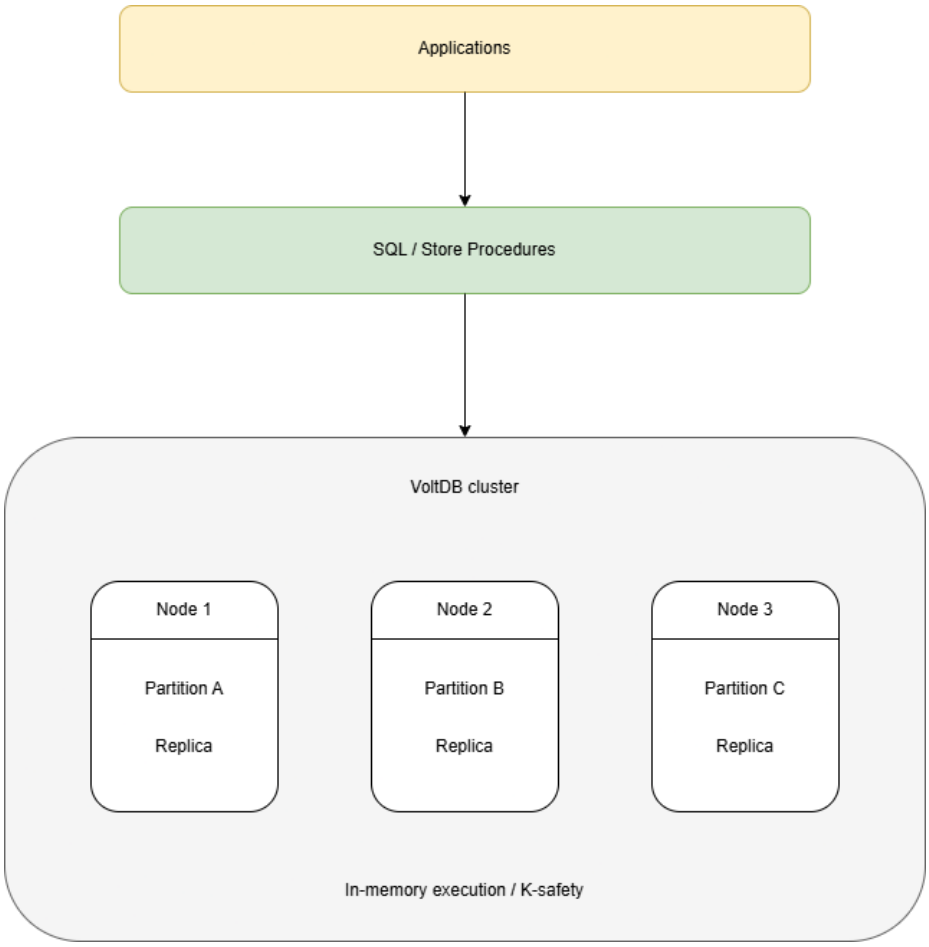


Fig. 5. VoltDB architecture diagram

VoltDB is composed of the following components:

- **Applications:** This block represents the client applications that send requests to the VoltDB cluster;
- **SQL / Stored Procedures:** This layer represents the main way applications interact with VoltDB. The system supports SQL, but it is especially designed to execute stored procedures efficiently for transactional workloads;
- **VoltDB cluster:** This block represents the distributed database cluster. It is composed of multiple nodes that work together to process transactions and store data;

- **Nodes:** These blocks represent the nodes of the VoltDB cluster. Each node stores part of the data and participates in transaction execution;
- **Partitions:** These blocks represent the data partitions. VoltDB divides the database into partitions so that transactions can be processed in parallel across different nodes;
- **Replica:** This block represents a copy of a partition stored on another node. Replicas improve fault tolerance and help the system remain available if a node fails;
- **In-memory execution / K-safety:** This part represents two important characteristics of VoltDB. The system is optimized for in-memory transaction processing, and it uses K-safety to replicate partitions across nodes and provide high availability.

The main advantages of VoltDB are the following:

- Very strong for low-latency OLTP, often used as an in-memory transactional system;
- High availability using K-safety (duplicate partitions so the cluster can survive node loss).

The main limitations are the following:

- Best performance often needs good partitioning and using its *"recommended style"* (procedures/partition-aware design);
- Higher K-safety (more replicas) can reduce performance (HA trade-off).

4 NewSQL Databases Comparison

This section presents a comparison of the main characteristics of TiDB, OceanBase, YugabyteDB, VoltDB(CE) and CockroachDB.
Table 1 presents an analysis of the following characteristics for all databases: archi-tecture style, foreign keys, isolation, partitioning strategy, replication method, scalability model, secondary indexes, and transaction model.

Table 1. Summary of the characteristics for each NewSQL Databases

Characteristics	CockroachDB	TiDB	OceanBase	YugabyteDB	VoltDB(CE)
Architecture style	Distributed SQL over transactional KV store	HTAP with separated SQL, row-store, and column-store components	Distributed relational database	Distributed SQL over sharded tablets	Shared-nothing in-memory
Foreign keys	Yes	Yes	Yes	Yes	No

Isolation	Strong. / Serializable	Immediate	Immediate	Strong writes, tunable reads	N/A
Partitioning strategy	Automatic key-range distribution	Range-based regions	Partition-based distribution	Hash and range sharding	Partition-key based
Replication method	Raft	Raft	Paxos	Raft	Source-replica / K-safety
Scalability model	Horizontal	Horizontal	Horizontal	Horizontal	Horizontal
Secondary indexes	Yes	Yes	Yes	Yes	Yes
Transaction model	ACID	ACID	ACID	Distributed ACID	In-memory OLTP

5 Conclusions and Future Work

This paper provides a comprehensive overview and comparison of five leading NewSQL databases: CockroachDB, TiDB, OceanBase, YugabyteDB, and VoltDB. These databases were selected based on their prominent position in the 2026 industry rankings. The analysis shows that, although these systems have the same goal of providing horizontal scalability without sacrificing ACID compliance, they use different architectural strategies and consensus mechanisms to reach it. TiDB, for example, distinguishes itself through its Hybrid Transactional/Analytical Processing (HTAP) capabilities, while VoltDB focuses on low-latency OLTP through a shared-nothing, in-memory execution model. Additionally, the study revealed that different replication methods, such as Raft, Multi-Paxos, and K-safety, introduce unique trade-offs regarding consistency, latency, and operational complexity. Ultimately, the choice of a NewSQL implementation must carefully align with the workload's particular needs, such as PostgreSQL or Oracle compatibility or specific partitioning requirements.

As future work, we intend to conduct a rigorous experimental evaluation of these databases to quantify the performance differences identified in this architectural review. The evaluation will use the TPC-C benchmark, an established standard for measuring transactional performance and scalability in OLTP systems. Experiments will be executed on the CloudLab platform using BenchBase to ensure a modernized, repeatable benchmarking process across all selected systems. This empirical phase will provide deeper insights into how these NewSQL architectures handle heavy workloads and distributed transaction complexities in practice.

References

1. Pavlo, A., Aslett, M.: What's Really New with NewSQL? ACM SIGMOD Record 45(2), 45–55 (2016)
2. Almassabi, A., Bawazeer, O., Adam, S.: Top NewSQL Databases and Features Classification. International Journal of Database Management Systems 10(2), 11–31 (2018).
3. Huang, D., Liu, Q., Cui, Q., Fang, Z., Ma, X., Xu, F., Shen, L., Tang, L., Zhou, Y., Huang, M., Wei, W., Liu, C., Zhang, J., Li, J., Wu, X., Song, L., Sun, R., Yu, S., Zhao, L., Cameron, N., Pei, L., Tang, X.: TiDB: A Raft-based HTAP Database. Proceedings of the VLDB Endowment 13(12), 3072–3084 (2020)
4. Santos, R.J.; Bernardino, J.; Vieira, M. A survey on data security in data warehousing: Issues, challenges and opportunities. In Proceedings of the 2011 IEEE EUROCON—International Conference on Computer as a Tool, Lisbon, Portugal, 1–4 (2011)
5. DB-Engines: DB-Engines Ranking, https://db-engines.com/en/ranking, last accessed 2026/03/15
6. Watanabe, Y., Kawashima, R., Matsuo, H.: Proxy-Based Transaction Acceleration for NewSQL. In: Proceedings of the 2023 Eleventh International Symposium on Computing and Networking Workshops (CANDARW 2023), Matsue, Japan, 28 November–1 December 2023, pp. 343–348 (2023)
7. Schumacher, K.: Benchmarking NewSQL Database VoltDB. Master's Project, San José State University, San Jose, CA, USA (2022), https://scholarworks.sjsu.edu/etd_projects/1101/, last accessed 2026/03/15
8. Håkansson, K., Rosenqvist, A.: Evaluation of CockroachDB in a Cloud-Native Environment. Bachelor's Thesis, Blekinge Institute of Technology, Karlskrona, Sweden (2021), http://urn.kb.se/resolve?urn=urn:nbn:se:bth-21671, last accessed 2026/03/15
9. Qu, L., Wang, Q., Chen, T., Li, K., Zhang, R., Zhou, X., Xu, Q., Yang, Z., Yang, C.: Are Current Benchmarks Adequate to Evaluate Distributed Transactional Databases? BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2(1) (2022).
10. Difallah, D.E., Pavlo, A., Curino, C., Cudré-Mauroux, P.: OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases. Proceedings of the VLDB Endowment 7(4), 277–288 (2014)
11. CMU Database Group: BenchBase: Multi-DBMS SQL Benchmarking Framework via JDBC. GitHub Repository, https://github.com/cmu-db/benchbase, last accessed 2026/03/17
12. Cockroach Labs: Architecture Overview, https://www.cockroachlabs.com/docs/stable/architecture/overview, last accessed 2026/03/07
13. PingCAP: TiDB Architecture. TiDB Documentation, https://docs.pingcap.com/tidb/stable/tidb-architecture/, last accessed 2026/03/08
14. OceanBase: Architecture. OceanBase Documentation, https://en.oceanbase.com/docs/common-oceanbase-database-10000000001029040, last accessed 2026/03/08
15. YugabyteDB: Architecture. YugabyteDB Documentation, https://docs.yugabyte.com/stable/architecture/, last accessed 2026/03/08
16. VoltDB: How VoltDB Works. VoltDB Documentation, https://docs.voltdb.com/UsingVoltDB/IntroHowVoltDBWorks.php, last accessed 2026/03/08

Ride-Sharing Ant Colony Optimization for Cost-Aware Organ Air Transportation

Natsuki Shimomura[1][0009−0005−9580−9869], Pedro Henrique González[2][0000−0003−0057−7670], Masayoshi Aritsugi[1][0000−0003−0861−849X], and Israel Mendonça[1][0000−0001−6819−4305]

[1] Kumamoto University, Japan
[2] Federal University of Rio de Janeiro, Brazil

Abstract. This study addresses the organ transplant air transportation problem, in which rapid and reliable route selection must balance destination priorities, time constraints, and operational costs. We propose a ride-sharing Ant Colony Optimization (ACO) method that jointly optimizes routes across multiple organ transportation cases, enabling cost reduction through shared flights while respecting cold ischemia time limits. Experiments on real-world instances from the Brazilian National Transplant System and synthetic benchmarks show that the proposed method improves destination priority by 34% on average compared to a shortest-path baseline, while reducing transportation costs by up to 25% when ride-sharing opportunities are available. Although the method may select slightly longer routes, the additional travel time remains clinically negligible relative to organ preservation limits. These results demonstrate that coordinated multi-case optimization can yield practically significant improvements in both recipient prioritization and resource efficiency for organ air transportation, highlighting its potential applicability in real-world decision support systems.

Keywords: Ant colony optimization · Medical Logistics· Organ transportation · Resource Sharing · Ride-sharing.

1 Introduction

Organ transplantation is the best, and frequently the only available treatment for end-stage organ failure [6]. However, the demand for transplantable organs far outweighs the available supply and this imbalance has important consequences for health.

According to the Health Resources and Services Administration (HRSA) [7], each organ has a limited period during which it can remain viable outside the body, known as the cold ischemic time (CIT). In organ transportation, shorter arrival times generally result in reduced CIT and are therefore associated with higher transplantation success rates [9].

In continent-sized countries such as Brazil, air transportation often represents the only feasible option for meeting the maximum preservation time constraints

of organs. For this reason, in 2001, the Brazilian Ministry of Health established cooperation agreements with airports. Under these agreements, the Brazilian Air Force and commercial airlines voluntarily provided free transportation of tissues, organs, and medical teams for transplantation purposes using military or commercial aircraft [4].

Furthermore, in Brazil, the National Transplantation Central (Central Nacional de Transplantes, CNT) is responsible for the procurement and distribution of organs and tissues for transplantation across states, as well as for the management and coordination of transplant waiting lists [4]. Although the planning of organ transportation is performed manually by CNT technicians, selecting an appropriate trade-off between speed and reliability from countless feasible routes—each constructed by combining subsets of hundreds of possible flights—constitutes an extremely challenging and inherently unfair task. Accordingly, Balster et al. [2] proposed a labeling algorithm that does not rely on commercial solvers and is capable of computing optimal solutions even under the stringent time-window constraints imposed by organ preservation limits.

However, air transportation entails high operational costs and complex airport procedures [3]. Although donors are not required to directly bear these operational costs, reducing the costs incurred by public institutions would allow resources to be reallocated to other medical needs. Therefore, the trade-off between maintaining organ viability and reducing operational costs underscores the need for transportation strategies that can achieve both objectives simultaneously.

Therefore, this study proposes an Ant Colony Optimization (ACO) algorithm that enables the sharing of air transportation resources across multiple organ transportation cases, with the objective of improving graft survival while reducing operational costs.

2 Background

2.1 Mathematical Formulation for Organ Transportation

Carrara et al. [5] proposed a Mixed-Integer Linear Programming (MILP) formulation to optimally address the problem of organ transportation for transplantation purposes. Their model can be regarded as an extension of the classical shortest path problem, augmented with additional constraints that reflect the specific operational requirements of organ logistics.

Graph Representation The transportation network is modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ represents the set of nodes corresponding to airports, and $\mathcal{A}$ denotes the set of directed arcs corresponding to available flights.The origin airport is denoted by s, while the destination airport is denoted by t. The set of intermediate airports that may be used as transshipment points is denoted by $\mathcal{V}_\Gamma$. Accordingly, the complete set of nodes is defined as

$$\mathcal{V} = \mathcal{V}_\Gamma \cup \{s, t\}. \tag{1}$$

For any pair of distinct nodes $i, j \in \mathcal{V}$, let $\mathcal{A}_{ij}$ denote the set of arcs representing all flights available from airport i to airport j.

Time Constraints and Operational Parameters Each arc $a \in \mathcal{A}_{ij}$ is associated with a scheduled departure time h_{ij}^a and a flight duration d_{ij}^a. Given the organ to be transplanted, a maximum allowable transportation time $D_{\max}$ is defined, reflecting the maximum preservation time during which the organ remains viable. The organ becomes available for transportation at time $H_{\text{available}}$ and must arrive at the destination airport t no later than a prescribed deadline $H_t^{\max}$. When the transportation path involves one or more intermediate airports from the set $\mathcal{V}_\Gamma$, sufficient time must be allocated for handling operations such as aircraft changes. This handling time is represented by the parameter τ_{ij}, which Carrara et al. [5] assume to be equal to 30 minutes for all airport pairs.

Although the formulation explicitly accounts for handling times, minimizing the number of intermediate stops is still desirable, as additional transfers increase both organ manipulation and exposure to potential disruptions.To balance early arrival at the destination with a reduced number of flights, a penalty parameter P, chosen by the modeler, is incorporated into the objective function.

Decision Variables Regarding the decision variables, let $T_i \geq 0$ denote the time at which the organ departs from node $i \in \mathcal{V} \setminus \{t\}$. In particular, for the origin airport s, the departure time must satisfy

$$T_s \geq H_{\text{available}}. \tag{2}$$

Furthermore, for each arc $a \in \mathcal{A}_{ij}$, a binary decision variable x_{ij}^a is introduced, which takes the value 1 if flight a is selected as part of the transportation path from node i to node j, and 0 otherwise.

2.2 A Dynamic Programming Labeling Algorithm

The organ transportation problem has been formulated as a resource-constrained shortest path problem (RCSP) [5], where the preservation time of the organ imposes strict time constraints on feasible routes.

Balster et al. [2] modeled the air transportation of organs as a shortest path problem with multiple flights between airports and proposed a labeling algorithm to efficiently solve its resource-constrained variant.

Their approach associates each partial path with a label that stores the accumulated arrival time and the number of flights, and applies dominance rules to discard inefficient labels during the search process. By exploiting dominance relations and preprocessing techniques, the algorithm is able to compute optimal solutions without relying on commercial solvers, even under tight time window constraints imposed by organ preservation limits.

However, this approach focuses on optimizing a single transportation path between a donor and a recipient, and does not address cost-sharing or coordination among multiple organ transports. In contrast, our study extends this

line of research by considering ride-sharing among multiple organ transports and aims to reduce the overall transportation cost while respecting preservation-time constraints.

2.3 Ant Colony Optimization

A wide variety of methods have been proposed for routing and path planning problems. One representative example is the Data Mule Routing Problem with Limited Autonomy (DMRP-wLA) [8], which addresses efficient route design for autonomous agents subject to limited autonomy due to the need for periodic recharging of communication devices.

To tackle this problem, previous studies have proposed hybrid approaches that combine Ant Colony Optimization (ACO), Random Variable Neighborhood Descent, and reinforcement learning [1], where the order of neighborhood exploration is adaptively learned and optimized.

Ant Colony Optimization is a metaheuristic inspired by the foraging behavior of ants, in which indirect information sharing is achieved through pheromone deposition and evaporation. By reinforcing high-quality routes discovered during the search process, ACO has been shown to be effective for large-scale routing problems. In particular, its effectiveness has been demonstrated in reducing both the total travel distance and operational time in UAV routing applications.

Although these problems differ in their application domains, they share a common structural characteristic with the organ air transportation problem: the need to efficiently explore and select routes under strict time and resource constraints. Motivated by this similarity, this study adopts an ACO framework that enables the sharing of route information among multiple agents through pheromone trails, and proposes a transportation planning method that explicitly considers ride-sharing by transporting multiple organs on the same flight in time-constrained organ air transportation.

3 Proposed Method

This section provides a comprehensive overview of the proposed method.

In this study, we propose a ride-sharing ant colony optimization (ACO) approach for organ air transportation that simultaneously considers arrival time and cost reduction achieved through shared transportation. In the proposed method, instead of optimizing each transportation case independently, route information from multiple cases is shared in the form of pheromone trails. This mechanism naturally encourages the selection of flight routes that can be shared across cases. As a result, the proposed approach effectively guides the optimization process toward routes that are jointly usable by multiple transportation cases.

The proposed ride-sharing ACO can be simplified into two main steps: solution construction and pheromone updating. The solution construction process is described in Section 3.1, while the pheromone updating mechanism is explained in Section 3.2.

3.1 The solution construction process

In the first step, which involves solution construction, the probability that ant k selects flight l from airport i to airport j is defined by Eq.3. Here, $\mathcal{F}(i)$ denotes the set of reachable (h, m) pairs from node i, where $h \in V$ is a destination airport and $m \in L$ is a flight index. The set $L = \{1, 2, ..., L_{ij}\}$ indexes all available flights on a given arc (i, j) ,where L_{ij} denotes the number of flights from airport i to airport j.

$\tau_{i,j,l}$ and $\eta_{i,j,l}$ denote the pheromone level and the heuristic information on arc (i, j, l), respectively, where $\eta_{i,j,l} = 1/d_{i,j,l}$. The parameters α and β control the relative influence of the pheromone trail and the heuristic information, respectively.

$$p_{i,j,l} = \frac{(\tau_{i,j,l})^{\alpha} (\eta_{i,j,l})^{\beta}}{\sum\limits_{(h,m)\in\mathcal{F}(i)} (\tau_{i,h,m})^{\alpha} (\eta_{i,h,m})^{\beta}} \tag{3}$$

The flight selection process is defined in Eq.4, where q denotes the probability of greedy selection. During the first iteration ($iteration = 1$) or whenever all feasible options are false - i.e., no path reaches a goal node - subsequent iterations ($iteration > 1$) employ sampling based selection.

$$(j*, l*) = \begin{cases} \arg \max\limits_{(h,m)\in\mathcal{F}_i} p(i, h, m) & \text{with probability } q \\ \text{sample according to } p_{i,j,l} & \text{with probability } 1 - q \end{cases} \tag{4}$$

Subsequently, the feasibility of air transportation constraints is evaluated. If the constraints are satisfied, the variable $feasible$ is set to true otherwise, it is set to false. When $feasible$ is true and node j is an end node, the corresponding path is recorded as feasible path. If $feasible$ is true but node j is not an end node, the algorithm proceeds to select the next node. If $feasible$ is false, the path is recorded as an infeasible path.

3.2 The pheromone updating mechanism

The pheromone update step consists of two components: evaporation, which prevents unlimited pheromone accumulation while maintaining exploration capability, and deposition, which reflects the search results of all ants that traverse arc (i, j, l). The pheromone level on arc (i, j, l) is defined in Eq.5, where ρ denotes the pheromone evaporation rate.

$$\tau_{i,j,l} \leftarrow (1 - \rho)\, \tau_{i,j,l} \tag{5}$$

Here, the amount of pheromone $\tau^k_{i,j,l}$ deposited by ant k on arc (i, j, l) is defined in Eq.6. $\Delta\tau^k_{i,j,l}$ represents the amount of pheromone deposited, and $\mathcal{K}_{i,j,l}$ denotes the set of ants that traverse arc (i, j, l).

$$\tau_{i,j,l} \leftarrow \tau_{i,j,l} + \sum_{k \in \mathcal{K}_{i,j,l}} \Delta\tau^k_{i,j,l} \tag{6}$$

As shown in Eq.6, pheromone is deposited without being separated by individual cases. This shared deposition mechanism promotes cooperation by reinforcing routes that can be repeatedly used across multiple transportation cases.

The amount of pheromone added for path evaluation is defined as Eq.7. Let c_k denote the total travel time of the path constructed by ant k, measured as the sum of flight durations and handling times along the path. As defined in Eq.8, $C_k(i)$ represents the set of feasible nodes that ant k can visit from node i.

In addition, Q denotes the amount of pheromone released by an ant in each iteration. Among the paths that satisfy the air transportation constraints, the path that reaches the destination in the shortest time is reinforced by depositing twice the amount of pheromone. Furthermore, a small amount of pheromone is also added to paths that do not satisfy the air transportation constraints, so that useful information can still be propagated even when no feasible path exists.

$$\Delta\tau^k_{i,j,l} = \begin{cases} \dfrac{2Q}{c_k} & \text{if } k \text{ is feasible and best} \\ \dfrac{Q}{c_k} & \text{if } k \text{ is feasible} \\ \dfrac{0.02Q}{c_k} & \text{if } k \text{ is non-feasible and best} \\ \dfrac{0.01Q}{c_k} & \text{if } k \text{ is non-feasible} \end{cases} \tag{7}$$

$$C_k(i) = \{(j,l) | j \in \text{unvisited}_k, l \in L, \tau_{i,j,l} > 0\} \tag{8}$$

Note that $C_k(i)$ is an ant-specific subset of $F(i)$: while $F(i)$ contains all reachable next-flight tuples from node i, $C_k(i)$ further restricts the set to airports not yet visited by ant k and to arcs with positive pheronome levels.

Moreover, the amount of additional pheromone for ride-sharing paths is defined in Eq.9. Here, $s_{i,j,l}$ denotes the number of transportation cases that share arc (i,j,l), and $\gamma > 0$ is aride-sharing reward coefficient that controls the srenght of the bonus pheromone for shared arcs. By reinforcing shared paths with additional pheromone, the proposed method is designed to encourage the discovery of ride-sharing opportunities.

$$\Delta\tau^k_{i,j,l} \leftarrow \Delta\tau^k_{i,j,l} \left(1 + \gamma(s_{i,j,l} - 1)\right) \tag{9}$$

4 EXPERIMENTAL SETUP

The following four subsections describe the experimental setup of this study. Section 4.1 presents the experimental environment, Section 4.2 introduces the instances, Section 4.3 explains parameter settings, and Section 4.4 describes the experimental procedure.

4.1 Experimental Environment

All experiments were conducted on a MacBook Air equipped with an Apple M4 processor (10 cores: 4 performance and 6 efficiency) and 24 GB of RAM, running macOS Tahoe 26.1. The proposed method was implemented in Python 3.9.12. All the experiments were executed on this single consumer-grade laptop without parallelization, demonstrating that the approach does not require high-performance computing infrastructure.

4.2 Instances

Each experimental instance consists of the following elements: a case identifier, the organ type, the time at which the organ becomes available for transportation ($H_{\text{available}}$), the origin airport, and a ranked list of candidate destination airports in descending order of priority.

Figure 1 shows the Brazilian domestic airport network used in this study, comprising 32 airports including all state capitals and major domestic airline hubs. Flight data from three commercial airlines (Azul, Gol, and Latam) are used to construct feasible routes.

Fig. 1. The airport network in Brazil [2]

Next, we describe the instances used in this study, which consist of a real-world instance and two types of artificial instances.

Real-world instance The first type of instance corresponds to real-world cases reported by Carrara et al. [5] within the CNT framework, as summarized in Table 1. In practice, destination selection has traditionally been performed manually by CNT technicians based on information from airline websites. The actually selected airports are highlighted in bold red in Table 1. Notably, in 12 of the 25 cases, the highest-priority recipient was not selected, highlighting the potential benefit of algorithmic decision support.

Table 1. REAL-WORLD INSTANCE and destinations chosen by CNT among a ranked list of receivers

				Priority order of destinations								
Case	Organ	$H_{\text{available}}$	Origin	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
C01	Kidney	02:42	SBSV	SBRF	SBFZ	SBRJ	**SBVT**	SBBH				
C02	Kidneys	06:50	SBRJ	SBVT	SBBH	SBRF	SBCT	**SBPA**				
C03	Kidneys	09:34	SBFZ	**SBRF**	SBRJ	SBVT	SBBR					
C04	Kidney	13:35	SBSV	SBRF	SBFZ	**SBPA**	SBBR	SBBH				
C05	Liver	18:45	SBBH	**SBVT**	SBRJ	SBCT	SBRF	SBPA				
C06	Kidneys	03:32	SBBH	SBRJ	SBVT	**SBPA**						
C07	Liver	18:45	SBCG	SBRF	**SBBR**	SBRJ	SBVT	SBBH	SBPA			
C08	Heart	18:35	SBCG	**SBSP**	SBBR							
C09	Kidneys	19:25	SBCG	**SBSP**	SBEG	SBBE	SBBR	SBGO	SBRF			
C10	Liver	02:35	SBSV	**SBRF**	SBFZ	SBRJ	SBCT	SBVT	SBBR	SBBH		
C11	Kidney	01:30	SBRB	**SBRF**	SBBR	SBBE	SBGO					
C12	Kidney	01:30	SBRB	SBRF	**SBGR**	SBBR	SBBE	SBGO				
C13	Liver	00:50	SBRB	**SBBR**	SBRJ	SBVT	SBRF	SBFZ				
C14	Liver	04:45	SBNT	SBSV	SBRF	**SBFZ**	SBRJ	SBBR	SBCT	SBVT		
C15	Kidney	16:20	SBFL	**SBSP**	SBPA	SBCT	SBRJ	SBRF	SBBH	SBBE	SBVT	
C16	Kidney	16:20	SBFL	SBPA	**SBCT**	SBRJ	SBRF	SBBH	SBBE	SBVT		
C17	Kidney	14:05	SBFL	SBRF	**SBSP**	SBPA						
C18	Liver	11:48	SBNT	SBRF	**SBFZ**	SBRJ	SBBR	SBVT	SBCT			
C19	Liver	00:53	SBNT	**SBGR**	SBRF	SBFZ	SBCT	SBRJ	SBBR	SBVT		
C20	Liver	12:55	SBSV	**SBFZ**	SBRF	SBRJ	SBBR					
C21	Kidney	16:50	SBFZ	SBSV	SBMO	**SBPA**	SBRJ	SBBE	SBTE	SBJP	SBRF	SBSL
C22	Kidneys	01:50	SBFZ	SBSV	SBMO	SBJP	SBRF	SBNT	**SBTE**			
C23	Heart	04:36	SBGO	**SBBR**								
C24	Lung	04:36	SBGO	**SBBR**								
C25	Liver	04:36	SBGO	**SBBR**	SBRJ	SBVT	SBPE	SBCE	SBPR			

Artificial instances To evaluate the effectiveness of the proposed method, we conduct experiments not only on instances derived from real-world operations but also on artificially constructed instances.

In real-world organ transportation, key factors such as the origin airport, the priority order of recipients, and the available transportation time vary on a daily basis. As a result, operational data are inherently non-stationary and

difficult to reproduce. Consequently, evaluating the proposed method solely on publicly available real-world instances is insufficient to assess its applicability and generality.

To address this issue and to systematically examine a wide range of situations, we additionally perform experiments using artificial instances constructed under controlled conditions.

Transfer-intensive instance The transfer-intensive instance is intentionally designed to include scenarios in which routes involving multiple transfers are likely to occur. The details of this instance are summarized in Table 2. In this instance, a large number of origin–destination pairs are designed to become feasible or advantageous only when intermediate airports are used, rather than relying on direct flights. This structure allows us to explicitly evaluate whether the proposed method can effectively exploit ride-sharing opportunities along transfer-based routes.

In this instance, a large number of origin–destination pairs are designed to become feasible or advantageous only when intermediate airports are used, rather than relying on direct flights. This structure allows us to explicitly evaluate whether the proposed method can effectively exploit ride-sharing opportunities and transfer-based routes.

In total, we consider five variants of this instance: four single-organ cases (heart-only, lung-only, liver-only, and kidney-only) and one multi-organ case. Due to space limitations, we report only the results for the multi-organ instance, as the other cases exhibit similar trends. By eliminating differences in time constraints arising from organ types, this variant enables a clearer evaluation of the pure effect of transporting multiple organs on the same flight, i.e., ride-sharing.

Table 2. Overview of TRANSFER-INTENSIVE INSTANCE (multi-organ)

				Priority order of destinations							
Case	Organ	$H_{\text{available}}$	Origin	1st	2nd	3rd	4th	5th	6th	7th	8th
C01	Kidney	03:10	SBSV	SBRF	SBFZ	SBRJ	SBCT	SBBH			
C02	Heart	05:30	SBCG	SBRF	SBBR	SBRJ	SBVT	SBBH	SBPA		
C03	Liver	01:20	SBGO	SBBR	SBGR	SBRJ	SBVT				
C04	Kidney	08:00	SBFZ	SBSV	SBMO	SBPA	SBRJ	SBBE	SBTE	SBJP	SBRF
C05	Liver	12:40	SBSV	SBFZ	SBRF	SBRJ	SBBR	SBBH			
C06	Kidney	06:15	SBRB	SBBR	SBRJ	SBVT	SBFZ	SBRF			
C07	Lung	04:00	SBGO	SBBR	SBRJ	SBVT					
C08	Kidney	10:30	SBFZ	SBSV	SBMO	SBRJ	SBPA	SBBE	SBTE	SBJP	SBRF

Randomized instance The randomized instance is constructed by randomly generating the origin airport, candidate destinations, organ types, and available transportation times. This instance evaluates the computational performance of the proposed method under general conditions without intentionally imposed

structural characteristics. The number of cases is expanded to 20; details are shown in Table 3.

Each case consists of a case identifier, organ type, available transportation time $H_{\text{available}}$, origin airport, and a prioritized list of candidate destination airports. Organ types are selected according to a weighted probability distribution reflecting their relative frequencies in practice. The available transportation time is generated uniformly at random within a 24-hour range. The origin airport is selected randomly from the airport set, and up to nine distinct destination airports are randomly chosen from the remaining airports and ordered to represent priority. To ensure reproducibility, a fixed random seed is used during instance generation.

Table 3. RANDOMIZED INSTANCE

				Priority order of destinations								
Case	Organ	$H_{\text{available}}$	Origin	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
C01	Kidney	20:07	SBKP	SBPV	SBSP	SBGR	SBCT	SBRP				
C02	Liver	21:28	SBCT	SBBE								
C03	Lung	05:35	SBGR	SBMO	SBSP	SBJP	SBRP					
C04	Liver	19:10	SBJP	SBSP	SBPA	SBKP	SBBV	SBBE				
C05	Heart	23:14	SBPA	SBVT	SBKP	SBMQ	SBBE	SBRB	SBPV	SBEG		
C06	Kidney	16:41	SBEG	SBRF	SBRP	SBSL	SBMO	SBBR	SBCG	SBRB	SBFZ	SBGR
C07	Kidney	14:58	SBGO	SBKP	SBAR	SBBR	SBBE	SBPJ	SBCT			
C08	Heart	19:30	SBBE	SBPA	SBBR							
C09	Lung	18:19	SBSP	SBEG	SBRF	SBAR	SBVT					
C10	Liver	13:20	SBAR	SBJP	SBMQ	SBTE	SBPV	SBCY	SBCT	SBGR		
C11	Liver	06:21	SBCT	SBSV	SBGO	SBBV	SBJP	SBPV	SBMO	SBCG	SBRF	
C12	Kidney	00:20	SBCG	SBPA	SBKP							
C13	Liver	03:57	SBCG	SBRP	SBRJ							
C14	Kidney	18:50	SBRB	SBGO	SBCY	SBRP	SBMQ	SBGR	SBBV	SBRF	SBJP	SBTE
C15	Liver	08:05	SBCY	SBEG								
C16	Kidney	02:23	SBCT	SBVT	SBBV	SBCG						
C17	Liver	00:04	SBGR	SBSP	SBRB	SBCT	SBBV	SBRF	SBAR	SBMO	SBCY	SBRP
C18	Heart	22:42	SBCT	SBRF	SBSP	SBKP	SBRJ	SBBE	SBSL	SBJP	SBVT	SBMQ
C19	Liver	22:58	SBEG	SBRJ	SBGR	SBFZ						
C20	Liver	04:09	SBRF	SBMQ	SBJP	SBSV	SBFZ					

4.3 Parameter Settings

To model organ transportation, we define the time-related parameters associated with the transplantation process. Following Carrara et al. [5], the corresponding values are summarized in Table4. Let D_{cg} denote the organ-dependent time required to perform the procurement surgery. The time required to transport the organ from the donor hospital to the origin airport s is denoted by D_{hs}, while D_{th} represents the transportation time from the destination airport t to the recipient hospital. Let D_{st} be the elapsed time from the moment the organ becomes available for transport at the origin airport s ($H_{\text{available}}$) until its arrival at the destination airport t (T_t). The maximum allowable transportation time between airports s and t is denoted by $D_{\max}$, such that the following con-

straint holds:$D_{st} \leq D_{\max}$. Finally, the maximum cold ischemia time, denoted by $\mathrm{CIT}_{\max}$, is defined as the upper bound on the organ preservation time.

Table 4. TIME PARAMETERS used in the transplantation process

Organ	$\mathrm{CIT}_{\max}$	D_{cg}	D_{hs}	D_{th}	$D_{\max}$
Heart	04:00	00:30	00:30	00:30	02:30
Lung	06:00	00:30	00:30	00:30	04:30
Liver	12:00	00:40	00:30	00:30	10:20
Kidney	36:00	01:20	00:30	00:30	33:40

In all experiments, the number of iterations was fixed to 100. The number of ants was set equal to the number of airport nodes, allowing the search effort to scale with the size of the transportation network. Moreover, three colonies were employed to enhance solution diversity while maintaining a reasonable computational cost. The parameters $\alpha = 1.0$, $\beta = 5.0$ and $\rho = 0.34$, $Q = 1.0$, following the same values used in the ACO proposed by Alves et al. [1]. The ride-sharing reward coefficient was set to $\gamma = 0.5$, as this parameter was the one that yield the best performance on preliminary experiments (We tested parameters from the range of 0.4 until 0.6). No additional parameter tuning was performed, as the focus of this study is on the impact of ride-sharing rather than on parameter optimization.

4.4 Experimental Procedure

The proposed ride-sharing ACO method is evaluated by comparison with the baseline approach proposed by Carrara et al. [2], which was described in Section 2.2.

For each instance, the proposed ACO-based method is executed for the fixed number of iterations. Since the baseline method is deterministic and always returns the same solution, it is executed only once per instance.

The metrics are shown below.

- **Priority**: Average priority rank of the selected destination airport. A lower value indicates that organs are delivered to higher-priority airports.
- **Time**: Average total transportation time from the origin to the destination airport, including flight times and transfer times when applicable.
- **Cost**: Average transportation cost measured as the number of flights used in the transportation path. When ride-sharing occurs, the cost of a shared flight is divided by the number of organs transported on that flight.

5 EXPERIMENTAL RESULTS

The result tables are primarily divided into three blocks: the CNT block, the Baseline block, and the Our Method block. The CNT block reports the transportation routes and total transportation times that were manually determined

by CNT in actual operations. The Baseline block presents the results obtained using the dynamic programming approach described in Section2.2, while the Our Method block shows the results produced by the proposed method. In the CNT block, entries such as "FAB" and "Charter aircraft" indicate cases in which military or charter flights were used instead of regular commercial flights.

Within each block, "Path" represents the computed transportation route. "Time" indicates the total transportation time, "Cost" denotes the normalized transportation cost, and "Prio" represents the priority rank of the selected destination at arrival, where smaller values correspond to higher priority. "Cost" is defined on a per-flight basis. When multiple organs are transported on the same flight segment, the cost of that flight is evenly divided among the number of organs sharing it. Therefore, routes involving shared flights may result in fractional cost values (e.g., 0.5 or 0.33).

For each metric, the best results between the Baseline method and the Ant Colony–based method are highlighted in bold red. "Avg" reports the average performance over all cases for each airline, while "Total Avg" summarizes the overall average across all airlines.

The experimental results are presented as follows. Section 5.1 reports the results for the real-world instance, while Section 5.2 presents the results for the artificial instances. The computational time of each algorithm is discussed in Section 5.3.

5.1 Real-world instance

We compared the baseline method and the proposed approach using real-world cases, and the results are summarized in Table 5. The results indicate that, although the baseline method achieves a shorter average total travel time by 31 minutes, the proposed method selects significantly higher-priority destination airports, with an average priority rank of 1.88 compared to 2.85 for the baseline. As a result of increased path sharing among trips, the average cost is reduced from 1.06 to 1.00, corresponding to a reduction of approximately 5.7%.

This suggests that the ride-sharing ACO successfully identifies paths that allow passengers to share trips efficiently. Moreover, the results show that the shortest route does not necessarily correspond to the highest-priority destination. By searching for paths that both allow ride-sharing and minimize travel time, the proposed method naturally introduces diversity in selected destinations, which increases the likelihood of choosing high-priority airports.

It is worth noting that the increase in average time (approximately 31 minutes) remains clinically negligible relative to the maximum allowable transportation times. Even for the most time-sensitive organ in our dataset (the heart) the additional travel time represents only 21% of the available margin. For kidneys, which constitute the majority of cases in the real-world instance, the 31-minute increase correspond to less than 2% of the 33-hour-40 minutes limit. Furthermore, the proposed method exhibits particularly strong performance for certain airline networks: when restricted to Gol flights, the method achieves a perfect average priority of 1.00 (compared to 3.00 for the baseline) while simultaneously

reducing the transportation costs by approximately 25%. These results suggest that the proposed ride-sharing ACO is especially effective when the underlying flight network offers sufficient route diversity to exploit sharing opportunities.

Table 5. Comparison of the Baseline and the Proposed Method on REAL-WORLD INSTANCE

		CNT				Baseline				Our Method			
Case	Comp	Path	Time	Cost	Prio	Path	Time	Cost	Prio	Path	Time	Cost	Prio
c01	azul	SBSV SBBH SBVT	08:11	2	4	SBSV SBRF	04:33	0.5	1	SBSV SBRF	04:33	0.5	1
c02	azul	SBRJ SBPA	17:42	1	5	SBRJ SBRF	04:05	1	2	SBRJ SBRF	04:05	1	2
c03	azul	SBFZ SBRF	09:51	1	1	SBFZ SBRF	01:51	1	1	SBFZ SBRF	01:51	1	1
c04	azul	SBSV SBGR SBPA	19:50	2	3	SBSV SBRF	02:50	0.5	1	SBSV SBRF	02:50	0.5	1
c07	azul	SBCG SBBR	FAB	1	2	SBCG SBKP SBRF	07:20	1	1	SBCG SBKP SBRF	07:20	1	1
c09	azul	SBCG SBSP	Charter aircraft	1	1	SBCG SBKP SBRF	06:40	1	6	SBCG SBKP SBRF	06:40	1	6
c10	azul	SBSV SBRF	06:21	1	1	SBSV SBRF	04:40	0.5	1	SBSV SBRF	04:40	0.5	1
c20	azul	SBSV SBNT SBFZ	04:07	2	1	SBSV SBRF	03:30	0.5	2	SBSV SBRF	03:30	0.5	2
c21	azul	SBFZ SBGR SBPA	20:05	2	3	SBFZ SBRF	**02:35**	**1**	8	SBFZ SBRF SBMO	04:45	2	**2**
c22	azul	SBFZ SBBR SBTE	21:33	2	6	SBFZ SBRF	05:10	1	4	SBFZ SBRF	05:10	1	4
c25	azul	SBGO SBBR	FAB	1	1	SBGO SBKP SBVT	**04:39**	2	3	SBGO SBKP SBRJ	05:19	2	**2**
Avg			13:27	1.46	2.55		**04:21**	**0.91**	2.73		04:36	1.00	**2.09**
c01	gol	SBSV SBBH SBVT	08:11	2	4	SBSV SBRJ	**05:38**	1	3	SBSV SBRF	07:18	**0.5**	**1**
c02	gol	SBRJ SBPA	17:42	1	5	SBRJ SBVT	03:55	1	1	SBRJ SBVT	03:55	1	1
c03	gol	SBFZ SBRF	09:51	1	1	SBFZ SBSV SBVT	**06:36**	2	3	SBFZ SBSV SBRF	06:56	**1**	**1**
c04	gol	SBSV SBGR SBPA	19:50	2	3	SBSV SBBR	**02:40**	0.5	4	SBSV SBRF	02:55	0.5	**1**
c10	gol	SBSV SBRF	06:21	1	1	SBSV SBBR	**05:20**	1	6	SBSV SBRF	07:25	**0.5**	**1**
c20	gol	SBSV SBNT SBFZ	04:07	2	1	SBSV SBBR	**03:20**	**0.5**	4	SBSV SBFZ	03:30	1	**1**
c21	gol	SBFZ SBGR SBPA	20:05	2	1	SBFZ SBBR SBRJ	**05:30**	2	4	SBFZ SBBR SBSV	06:00	2	**1**
c22	gol	SBFZ SBBR SBTE	21:33	2	6	SBFZ SBBR SBSV	**08:45**	2	1	SBFZ SBSV	12:15	**0.5**	1
c25	gol	SBGO SBBR	FAB	1	1	SBGO SBGR SBBR	05:54	2	1	SBGO SBGR SBBR	05:54	2	1
Avg			13:27	1.56	2.56		**05:17**	1.33	3.00		06:14	**1.00**	**1.00**
c01	latam	SBSV SBBH SBVT	08:11	2	4	SBSV SBGR SBRJ	**06:08**	2	**3**	SBSV SBGR SBCT	06:18	2	4
c02	latam	SBRJ SBPA	17:42	1	5	SBRJ SBVT	04:15	1	1	SBRJ SBVT	04:15	1	1
c03	latam	SBFZ SBRF	09:51	1	1	SBFZ SBRF	06:01	1	1	SBFZ SBRF	06:01	1	1
c04	latam	SBSV SBGR SBPA	19:50	2	3	SBSV SBBR	04:55	1	4	SBSV SBBR	04:55	1	4
c07	latam	SBCG SBBR	FAB	1	2	SBCG SBGR SBPA	**05:20**	1.5	5	SBCG SBGR SBBR	06:20	**1**	**2**
c09	latam	SBCG SBSP	Charter aircraft	1	1	SBCG SBGR SBBR	05:40	1.5	4	SBCG SBGR SBBR	05:40	**1**	4
c10	latam	SBSV SBRF	06:21	1	1	SBSV SBBR	**04:50**	**1**	6	SBSV SBBR SBRF	08:25	2	**1**
c11	latam	SBRB SBBR SBRF	09:18	2	1	SBRB SBBR	05:10	0.33	2	SBRB SBBR	05:10	0.33	2
c12	latam	SBRB SBBR SBGR	08:30	2	2	SBRB SBBR	05:10	0.33	3	SBRB SBBR	05:10	0.33	3
c13	latam	SBRB SBBR	06:45	1	1	SBRB SBBR	05:50	0.33	1	SBRB SBBR	05:50	0.33	1
c20	latam	SBSV SBNT SBFZ	04:07	2	1	SBSV SBSP SBRJ	**04:35**	2	3	SBSV SBSP SBRJ	05:35	2	3
c22	latam	SBFZ SBBR SBTE	21:33	2	6	SBFZ SBTE	**01:55**	1	5	SBFZ SBRF	02:10	1	**4**
c24	latam	SBGO SBBR	FAB	1	1	SBGO SBBR	03:09	0.5	1	SBGO SBBR	03:09	0.5	1
c25	latam	SBGO SBBR	FAB	1	1	SBGO SBBR	02:59	0.5	1	SBGO SBBR	02:59	0.5	1
Avg			13:27	1.43	2.14		**04:42**	1.00	2.86		05:08	1.00	**2.29**
Total Avg			13:27	1.47	2.38		**04:44**	1.06	2.85		05:15	**1.00**	**1.88**

5.2 Artificial instances

Transfer-intensive instance We further evaluated the baseline method and the proposed approach on synthetic cases. Experiments were conducted using instances containing only heart, only lung, only liver, only kidney, as well as instances combining multi organs. The results for each scenario are summarized in Tables 6 to 10. For instances with a short maximum allowable travel distance ($D_{\max}$), such as heart and lung cases, the number of reachable airports is limited. As a result, direct flights are more likely to be selected, and no difference in transportation cost is observed between the baseline and the proposed method. Across all cases except for heart, the priority of the selected airports remained high. The differences in priority scores ranged from 0.43 to 1.43, indicating that the selected airports maintained higher priority values. Additionally, in the liver and lung cases, a cost reduction of approximately 3.5% was achieved through

ride-sharing, demonstrating the effectiveness of the proposed method under constrained conditions. In instances involving multiple organs, multiple ride-sharing occurrences were observed with the proposed ride-sharing ACO compared to the baseline. However, because the ride-sharing ACO sometimes selects paths redundantly, the overall travel cost ended up being comparable to that of the baseline.

These results highlight that, while ride-sharing opportunities can be effectively exploited, excessive path selection may counterbalance potential cost savings.

Table 6. Comparison of the Baseline and the Proposed Method on the HEART-ONLY TRANSFER-INTENSIVE INSTANCE

		Baseline				Our Method			
Case	Comp	Path	Time	Cost	Prio	Path	Time	Cost	Prio
c02	latam	SBCG SBBR	02:15	1	2	SBCG SBBR	02:15	1	2
Total Avg			02:15	1	2		02:15	1	2

Table 7. Comparison of the Baseline and the Proposed Method on the LUNG-ONLY TRANSFER-INTENSIVE INSTANCE

		Baseline				Our Method			
Case	Comp	Path	Time	Cost	Prio	Path	Time	Cost	Prio
c01	azul	SBSV SBRF	04:05	1	1	SBSV SBRF	04:05	1	1
c04	azul	SBFZ SBBE	03:10	1	5	SBFZ SBBE	03:10	1	5
c05	azul	SBSV SBRF	03:45	1	2	SBSV SBRF	03:45	1	2
Avg			03:40	1.00	2.67		03:40	1.00	2.67
c05	gol	SBSV SBBR	**03:35**	1	4	SBSV SBFZ	03:45	1	**1**
c08	gol	SBFZ SBSV	03:35	1	1	SBFZ SBSV	03:35	1	1
Avg			**03:35**	1.00	2.50		03:40	1.00	**1.00**
c02	latam	SBCG SBBR	02:15	1	2	SBCG SBBR	02:15	1	2
c07	latam	SBGO SBBR	03:45	1	1	SBGO SBBR	03:45	1	1
Avg			03:00	1.00	1.50		03:00	1.00	1.50
Total Avg			**03:27**	1.00	2.29		03:28	1.00	**1.86**

Randomized instance Finally, we tested the methods on randomly generated cases, and the results are presented in Table 11. In these cases, compared with the baseline, the proposed method still selected high-priority airports, achieving a priority score of 1.12 ($= 3.69 - 2.57$). However, due to limited path sharing, the travel cost was approximately 6.1% higher than that of the baseline.

This outcome can be attributed to the limited availability of ride-sharing opportunities in these random instances. These findings suggest that, in scenarios

Table 8. Comparison of the Baseline and the Proposed Method on the LIVER-ONLY TRANSFER-INTENSIVE INSTANCE

		Baseline				Our Method			
Case	Comp	Path	Time	Cost	Prio	Path	Time	Cost	Prio
c01	azul	SBSV SBRF	04:05	1	1	SBSV SBRF	04:05	1	1
c02	azul	SBCG SBKP SBBR	09:15	2	2	SBCG SBKP SBBR	09:15	2	2
c03	azul	SBGO SBKP SBVT	**08:05**	1	4	SBGO SBKP SBBR	08:35	1	**1**
c04	azul	SBFZ SBBE	**03:10**	**1**	5	SBFZ SBRF SBMO	05:40	2	**2**
c05	azul	SBSV SBRF	03:45	1	2	SBSV SBRF	03:45	1	2
c07	azul	SBGO SBKP SBVT	**05:25**	1	3	SBGO SBKP SBBR	05:55	1	**1**
c08	azul	SBFZ SBRF	**05:40**	**1**	8	SBFZ SBRF SBSV	08:20	2	**1**
Avg			**05:37**	**1.14**	3.57		06:30	1.43	**1.43**
c01	gol	SBSV SBRJ	**05:10**	1	3	SBSV SBRF	06:50	1	**1**
c02	gol	SBCG SBGR SBBR	05:10	1.5	2	SBCG SBGR SBBR	05:10	1.5	2
c03	gol	SBGO SBGR	06:40	0.5	2	SBGO SBGR	06:40	0.5	2
c04	gol	SBFZ SBSV	06:05	0.5	1	SBFZ SBSV	06:05	0.5	1
c05	gol	SBSV SBBR	**03:35**	1	4	SBSV SBFZ	03:45	1	**1**
c07	gol	SBGO SBGR SBBR	06:40	1	1	SBGO SBGR SBBR	06:40	1	1
c08	gol	SBFZ SBSV	03:35	0.5	1	SBFZ SBSV	03:35	0.5	1
Avg			**05:16**	0.86	2.00		05:32	0.86	**1.29**
c01	latam	SBSV SBBR SBCT	**06:55**	2	4	SBSV SBBR SBRJ	07:25	2	**3**
c02	latam	SBCG SBBR	02:15	1	2	SBCG SBBR	02:15	1	2
c03	latam	SBGO SBGR	**05:15**	1	2	SBGO SBBR	06:25	**0.5**	**1**
c04	latam	SBFZ SBSP SBRJ	**06:40**	2	4	SBFZ SBSV	07:55	**0.5**	**1**
c05	latam	SBSV SBSP SBRJ	04:50	2	3	SBSV SBSP SBRJ	04:50	2	3
c07	latam	SBGO SBBR	03:45	1	1	SBGO SBBR	03:45	**0.5**	1
c08	latam	SBFZ SBTE	**04:50**	1	6	SBFZ SBSV	05:25	**0.5**	**1**
Avg			**04:55**	1.43	3.14		05:25	**1.00**	**1.71**
Total Avg			**05:16**	1.14	2.90		05:49	**1.10**	**1.47**

Table 9. Comparison of the Baseline and the Proposed Method on the KIDNEY-ONLY TRANSFER-INTENSIVE INSTANCE

		Baseline				Our Method			
Case	Comp	Path	Time	Cost	Prio	Path	Time	Cost	Prio
c01	azul	SBSV SBRF	04:05	1	1	SBSV SBRF	04:05	1	1
c02	azul	SBCG SBKP SBBR	09:15	2	2	SBCG SBKP SBBR	09:15	2	2
c03	azul	SBGO SBKP SBVT	**08:05**	1	4	SBGO SBKP SBRJ	08:45	1	**3**
c04	azul	SBFZ SBBE	**03:10**	**1**	5	SBFZ SBRF SBSV	05:55	2	**1**
c05	azul	SBSV SBRF	03:45	1	2	SBSV SBRF	03:45	1	2
c07	azul	SBGO SBKP SBVT	**05:25**	1	3	SBGO SBKP SBRJ	06:05	1	**2**
c08	azul	SBFZ SBRF	**05:40**	**1**	8	SBFZ SBRF SBSV	08:20	2	**1**
Avg.			**05:37**	**1.14**	3.57		06:35	1.43	**1.71**
c01	gol	SBSV SBRJ	**05:10**	1	3	SBSV SBRF	06:50	1	**1**
c02	gol	SBCG SBGR SBBR	**05:10**	1.5	**2**	SBCG SBGR SBRJ	05:20	1.5	3
c03	gol	SBGO SBGR	06:40	0.5	2	SBGO SBGR	06:40	0.5	2
c04	gol	SBFZ SBSV	06:05	0.5	1	SBFZ SBSV	06:05	0.5	1
c05	gol	SBSV SBBR	**03:35**	1	4	SBSV SBFZ	03:45	1	**1**
c07	gol	SBGO SBGR SBBR	**06:40**	1	**1**	SBGO SBGR SBRJ	06:50	1	2
c08	gol	SBFZ SBSV	03:35	0.5	1	SBFZ SBSV	03:35	0.5	1
Avg.			**05:16**	0.86	2.00		05:35	0.86	**1.57**
c01	latam	SBSV SBBR SBCT	**06:55**	2	4	SBSV SBBR SBRJ	07:25	2	**3**
c02	latam	SBCG SBBR	02:15	1	2	SBCG SBBR	02:15	1	2
c03	latam	SBGO SBGR	**05:15**	1	2	SBGO SBBR	06:25	**0.5**	**1**
c04	latam	SBFZ SBSP SBRJ	**06:40**	2	4	SBFZ SBSV	07:55	**0.5**	**1**
c05	latam	SBSV SBSP SBRJ	**04:50**	2	3	SBSV SBSP SBRJ	05:50	2	3
c07	latam	SBGO SBBR	03:45	1	1	SBGO SBBR	03:45	**0.5**	1
c08	latam	SBFZ SBTE	**04:50**	1	6	SBFZ SBSV	05:25	**0.5**	**1**
Avg.			**04:55**	1.43	3.14		05:34	**1.00**	**1.71**
Total Avg.			**05:16**	1.14	2.90		05:55	**1.10**	**1.67**

Table 10. Comparison of the Baseline and the Proposed Method on the MULTI-ORGAN TRANSFER-INTENSIVE INSTANCE

		Baseline				Our Method			
Case	Comp	Path	Time	Cost	Prio	Path	Time	Cost	Prio
c01	azul	SBSV SBRF	04:05	1	1	SBSV SBRF	04:05	1	1
c02	azul	SBCG SBKP SBBR	09:15	2	2	SBCG SBKP SBBR	09:15	2	2
c03	azul	SBGO SBKP SBVT	**08:05**	**1**	4	SBGO SBKP SBRJ	08:45	1.5	**3**
c04	azul	SBFZ SBBE	**03:10**	**1**	5	SBFZ SBRF SBMO	05:40	2	**2**
c05	azul	SBSV SBRF	03:45	1	2	SBSV SBRF	03:45	1	2
c07	azul	SBGO SBKP SBVT	05:25	**1**	3	SBGO SBKP SBVT	05:25	1.5	3
c08	azul	SBFZ SBRF	**05:40**	**1**	8	SBFZ SBRF SBSV	08:20	2	**1**
Avg			**05:37**	**1.14**	3.57		06:27	1.57	**2.00**
c01	gol	SBSV SBRJ	**05:10**	1	3	SBSV SBRF	06:50	1	**1**
c02	gol	SBCG SBGR SBBR	**05:10**	**1.5**	**2**	SBCG SBGR SBRJ	05:20	2	3
c03	gol	SBGO SBGR	06:40	0.5	2	SBGO SBGR	06:40	0.5	2
c04	gol	SBFZ SBSV	06:05	0.5	1	SBFZ SBSV	06:05	0.5	1
c05	gol	SBSV SBBR	**03:35**	1	4	SBSV SBFZ	03:45	1	**1**
c07	gol	SBGO SBGR SBBR	06:40	**1**	1	SBGO SBGR SBBR	06:40	1.5	1
c08	gol	SBFZ SBSV	03:35	0.5	1	SBFZ SBSV	03:35	0.5	1
Avg			**05:16**	**0.86**	2.00		05:33	1.00	**1.43**
c01	latam	SBSV SBBR SBCT	**06:55**	2	4	SBSV SBRJ	07:45	**1**	**3**
c02	latam	SBCG SBBR	02:15	1	2	SBCG SBBR	02:15	1	2
c03	latam	SBGO SBGR	**05:15**	1	2	SBGO SBBR	06:25	**0.5**	**1**
c04	latam	SBFZ SBSP SBRJ	**06:40**	2	4	SBFZ SBSV	07:55	**0.5**	**1**
c05	latam	SBSV SBSP SBRJ	04:50	2	3	SBSV SBSP SBRJ	04:50	2	3
c07	latam	SBGO SBBR	03:45	1	1	SBGO SBBR	03:45	**0.5**	1
c08	latam	SBFZ SBTE	**04:50**	1	6	SBFZ SBSV	05:25	**0.5**	**1**
Avg			**04:55**	1.43	3.14		05:28	**0.86**	**1.71**
Total Avg			**05:16**	1.14	2.90		05:50	1.14	**1.71**

where the ride-sharing ACO searches for paths across all cases, an increase in path diversity can lead to the selection of higher-cost paths, even as the method continues to prioritize high-priority destinations.

Despite the limited cost advantage observed in the randomized instances, the proposed method consistently improved destination priority across all experimental settings. This robustness indicates that the priority-aware behavior of the ride-sharing ACO is not an artifact of specific instance structures but rather an emergent property of the shared pheromone mechanism, which naturally biases the search towards routes that are reachable under multiple transportation scenarios.

Table 11. Comparison of the Baseline and the Proposed Method on the RANDOMIZED INSTANCE

		Baseline				Our Method			
Case	Comp	Path	Time	Cost	Prio	Path	Time	Cost	Prio
c01	azul	SBKP SBRP	**04:08**	1	5	SBKP SBCT	04:28	1	**4**
c04	azul	SBJP SBRF SBBE	05:55	2	5	SBJP SBRF SBBE	05:55	2	5
c07	azul	SBGO SBKP	01:52	1	1	SBGO SBKP	01:52	1	1
c10	azul	SBAR SBSP SBCT	09:15	2	6	SBAR SBSP SBCT	09:15	2	6
c11	azul	SBCT SBKP SBSV	**04:19**	2	**1**	SBCT SBKP SBGO	08:09	2	2
c12	azul	SBCG SBKP	04:40	1	2	SBCG SBKP	04:40	1	2
c16	azul	SBCT SBKP SBVT	**07:02**	2	1	SBCT SBKP SBVT	12:22	2	1
c17	azul	SBGR SBCT	07:56	1	3	SBGR SBCT	07:56	1	3
c20	azul	SBRF SBFZ	**05:11**	1	4	SBRF SBJP	05:16	1	**2**
Avg			**05:45**	1.50	3.30		06:39	**1.44**	**2.89**
c06	gol	SBEG SBBR	**03:14**	**1**	5	SBEG SBBR SBRF	07:29	2	**1**
c07	gol	SBGO SBBR	04:32	1	3	SBGO SBBR	04:32	1	3
c09	gol	SBSP SBRF	03:41	1	2	SBSP SBRF	03:41	1	2
c10	gol	SBAR SBGR	06:40	1	7	SBAR SBGR	06:40	1	7
c11	gol	SBCT SBSP SBGO	**06:39**	2	2	SBCT SBSP SBSV	07:04	2	**1**
c12	gol	SBCG SBGR SBPA	**10:30**	**1.5**	1	SBCG SBSP SBPA	13:00	2	1
c13	gol	SBCG SBGR SBRJ	06:53	**1.5**	2	SBCG SBGR SBRJ	06:53	2	2
c16	gol	SBCT SBGR SBCG	**08:47**	2	3	SBCT SBSP SBVT	13:27	2	**1**
c20	gol	SBRF SBBR SBSV	06:26	2	3	SBRF SBBR SBSV	06:26	2	3
Avg			**06:22**	**1.44**	3.11		07:41	1.67	**2.33**
c06	latam	SBEG SBGR	**04:09**	1	9	SBEG SBBR	04:14	1	**5**
c07	latam	SBGO SBSP SBBR	**04:27**	2	3	SBGO SBSP SBBR	05:42	2	3
c09	latam	SBSP SBVT	01:26	1	4	SBSP SBVT	01:26	1	4
c10	latam	SBAR SBGR	**05:15**	**1**	7	SBAR SBGR SBJP	09:50	2	**1**
c11	latam	SBCT SBGR SBRF	**05:14**	2	8	SBCT SBRJ SBSV	05:29	2	**1**
c12	latam	SBCG SBBR SBPA	11:00	1.5	1	SBCG SBBR SBPA	11:00	1.5	1
c13	latam	SBCG SBBR SBRJ	06:38	1.5	2	SBCG SBBR SBRJ	06:38	1.5	2
c16	latam	SBCT SBGR SBCG	**07:02**	2	3	SBCT SBSP SBVT	10:22	2	**1**
c17	latam	SBGR SBAR	02:36	1	6	SBGR SBAR	02:36	1	6
c20	latam	SBRF SBBR SBSV	**06:56**	2	3	SBRF SBBR SBMQ	07:56	2	**1**
Avg			**05:28**	**1.50**	4.60		06:31	1.60	**2.50**
Total Avg			**05:50**	**1.48**	3.69		06:56	1.57	**2.57**

5.3 Computational Time

The execution times for the real-world instance are as follows: 0.48 seconds for the baseline method and 727.32 seconds (approximately 12 minutes) for the proposed approach. This difference arises because the baseline processes each case independently within a limited search space, whereas the proposed approach simultaneously considers multiple cases and explores potential ride-sharing opportunities, which requires a substantially larger search space.

Despite this increase, the proposed method remains practically viable for organ transportation planning. All experiments were conducted on a standard consumer-grade laptop (Macbook air), demonstrating that the approach does not require specialized high-performance computing infrastructure. More importantly, the execution time of approximately 12 minutes is well within the operational margins imposed by organ preservation constraints. Even for the most time-critical organ type considered in this study — the heart, with a maximum cold ischemia time of 4 hours and a maximum allowable airport-to-airport transportation time of 2 hours and 30 minutes — the computation time represents less than 10% of the available planning window. For organs with longer preservation times, such as kidneys (up to 36 hours), the computational overhead is negligible relative to the decision-making timeframe.

Furthermore, the current implementation is written entirely in Python, which prioritizes development flexibility over execution speed. Reimplementing the algorithm in a compiled language such as C or C++ could yield substantial reductions in computation time, potentially by an order of magnitude, making the approach even more suitable for time-sensitive scenarios.

Therefore, its execution time is fully compatible with the offline planning context in which organ transportation logistics are coordinated, and it delivers higher-priority destination selections and cost-effective ride-sharing paths that cannot be obtained by simple shortest-path-based approaches.

6 CONCLUSIONS

In this study, we proposed a ride-sharing–aware path planning method based ACO for the organ transplant transportation problem and conducted a comparative evaluation against a baseline approach.

The experimental results demonstrate that, although the proposed method does not always select the path with the shortest travel time, it consistently chooses routes leading to higher-priority destination airports and effectively reduces transportation costs by exploiting ride-sharing opportunities. In particular, for instances with strict travel-distance constraints, the cost reduction achieved through ride-sharing is especially pronounced.

On the other hand, when ride-sharing opportunities are limited or when the diversity of candidate paths becomes excessively high, redundant path selections may occur, which can offset the cost reduction benefits. This observation highlights a trade-off between exploration diversity and cost efficiency in ride-sharing–oriented path search.

While the proposed ride-sharing ACO framework shows promising potential, several limitations remain. For instance, the experimental evaluation is limited to a single deterministic baseline, namely the shortest-path labeling algorithm of Balster et al. [2], which does not explicitly consider ride-sharing or priority. Future work will include a broader set of benchmark methods for a more comprehensive evaluation. Also, the real-world dataset is relatively small (25 cases), which may limit the statistical reliability and generalizability of the results. Expanding the dataset and considering larger and more diverse instances will be addressed in future work. Improving cost efficiency remains an important challenge. In particular, reducing unnecessary path exploration by suppressing routes with low ride-sharing potential is a key direction for future research. Furthermore, the results on randomized instances suggest that the proposed method may not always outperform the baseline in scenarios with limited ride-sharing opportunities. This highlights the need for more robust search strategies that can adapt to such conditions. Finally, we plan to conduct ablation studies to isolate the individual contributions of ride-sharing and shared pheromone mechanisms.

References

1. Alves, C., Mendonça, I., de Almeida Guimarèes, V., González, P.H.: Aco with reinforcement learning applied to rescues operations on urban forests. In: 2024 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE (2024)
2. Balster, I., Caetano, J.A., Ribeiro, G.M., Bahiense, L.: Optimizing the transport of organs for transplantation. Computers & Operations Research **176**, 106934 (2025)
3. Bellotti, H.B., Francoso, M.T.: System for transporting human organs. Case Studies on Transport Policy **9**(2), 431–442 (2021)
4. Brazilian Ministry of Health: National transplant system (sistema nacional de transplantes). Online (2022), `https://www.gov.br/saude/pt-br/composicao/saes/snt`
5. Carrara, B.A., Heinzen, E., Leal Junio, I.C., Ribeiro, G.M.: Optimizing the organ transportation for transplant in brazil by commercial domestic flights. International journal of transport economics: Rivista internazionale di economia dei trasporti: XLV, 2, 2018 pp. 185–210 (2018)
6. European Directorate for the Quality of Medicines & HealthCare: Guide to the Quality and Safety of Organs for Transplantation, 9th Edition. Council of Europe / EDQM, Strasbourg, France (2025), `https://www.edqm.eu/en/guide-quality-and-safety-of-organs-for-transplantation`
7. Health Resources and Services Administration: Matching donors and recipients. Online (2021), `https://www.organdonor.gov/learn/process/matching`
8. Morais, I., de Almeida Guimarães, V., da Silva, E.B., González, P.H.: Prescriptive analytics in smart cities: A combinatorial approach in rescue operations. In: Ibero-American Congress of Smart Cities. pp. 131–145. Springer (2021)
9. Ponticelli, C.E.: The impact of cold ischemia time on renal transplant outcome. Kidney international **87**(2), 272–275 (2015)

An Algorithm for Mining Vertical Quantitative Frequent Patterns from Dense Data

Carson K. Leung, Thanh Trung Jack Nguyen, and Adam G.M. Pazdor

University of Manitoba, Winnipeg, MB, Canada
Carson.Leung@UManitoba.ca

Abstract. In the era of big data, mining frequently co-occurring patterns and association rules has become important. In many real-world database engineered applications, data are associated with quantitative values, requiring the use of quantitative frequent pattern mining techniques. However, mining quantitative frequent patterns from dense datasets is computationally expensive. In this paper, we present an algorithm for mining vertical quantitative frequent patterns from dense data. The algorithm makes good use of vertical representation of dense data, which is crucial for efficient quantitative frequent pattern mining. To reduce memory consumption and thus speed up the mining process, the algorithm stores only the transactions in which singleton items are absent. At each iterative stage, it further maintains only differential sets (diffsets), thereby significantly reducing computational overhead. In addition, the algorithm integrates quantitative attributes directly into the mining process. Evaluation results on high-density benchmark datasets demonstrate that the presented algorithm achieves superior performance in both execution time and scalability compared with related works. Furthermore, the algorithm can be extended to support other data science tasks, such as high-utility pattern mining.

Keywords: Database engineered application, Data mining, Frequent pattern, Quantitative value, Vertical mining, Set representation.

1 Introduction

With technological advancements, huge volumes of data—which can be of different levels of varsity—can be generated or collected at a rapid rate from a wide variety of rich data sources in numerous real-world data engineered applications [1, 2]. Embedded in these big data [3-5] are useful information, insight, and knowledge, which can be discovered by database engineered solutions, via data science [6-10], big data management and analytics [11-14], data mining [15-17], machine learning [18, 19], and/or data visualization [20, 21] techniques. Database engineered solutions help users to gain insight or discover knowledge from the real-world database engineered applications such as bioinfor-

matics [22], health informatics [23-25], transport analytics [26], social media analytics [27-29], social network analysis [30-36], and environmental analysis [37, 38]. The insight or discovered knowledge can then transform into actions and progress, which can help save life and improve our living conditions.

In many applications, it is interesting to find frequently occurring patterns or trends. This leads to frequent pattern mining and/or association rule mining [39, 40]. As a popular big data analytics and knowledge discovery task, association rule mining aims to discover rules that reveal interesting associations among the antecedents and consequents of the rules. Generally, these rules are mined by first discovering frequent patterns and then using these discovered frequent patterns to form the rules. *Frequent pattern mining* aims to discover frequently occurring sets of items (e.g., merchandise items, events) from big data. Given a series of transactions containing a set of items, frequent pattern mining seeks to determine the sets of items, which occur in many transactions. In addition, we wish to discover interesting association rules. Association rules state that whenever a certain set of items occurs in a transaction, another set of items tends to occur in that transaction. The problems of frequent pattern mining and association rule mining form the basis of many real-life applications such as marketing in business, discovering biological patterns, studying human populations, and web log mining.

Frequent patterns can be discovered horizontally by transaction-centric mining algorithms or vertically by item-centric mining algorithms. The Apriori algorithm [41, 42] is an example of horizontal transaction-centric frequent pattern mining algorithms, with which data are represented as a collection of transactions. Each transaction captures the presence or absence of items. For example, consider a transaction-centric representation of a dataset D containing three transactions: t1 = {a, b}, t2 = {a, b, c} and t3 = {b, c}. This representation captures that items a & b are in transaction t1, items a, b & c are in transaction t2, and items b & c are in transaction t3.

Alternatively, frequent patterns can also be discovered vertically. The Eclat (Equivalence CLAss Transformation) algorithm [43] is an example of vertical item-centric frequent pattern mining algorithms, with which data are represented as a collection of equivalence classes according to their prefix item labels. Each domain item is represented by one of these classes, and the corresponding transaction ID set (tidset) for an item captures which transactions contain the specific item. Specifically, the set contains transaction IDs. An advantage of such a set representation is that the size of set is proportional to the density of the data. Sparse data would lead to a small transaction ID set. The algorithm was shown to be efficient as it takes advantage of set operations in the mining process. For example, consider an item-centric representation of the aforementioned dataset D: tidset(a) = {t1, t2}, tidset(b) = {t1, t2, t3} and tidset(c) = {t2, t3}. This representation captures that item a is in transactions t1 & t2, item b is in all three transactions, and item c is in transaction t2 & t3.

While Eclat algorithm efficiently handles sparse data, its set representation grows for dense data as its size of set is proportional to the density of the data. To mitigate this prob-

lem, the dEclat algorithm [44, 45] was proposed. As a variant of the Eclat algorithm, dEclat was designed for handling dense data. It utilizes both a vertical database layout and a data structure called a difference set (i.e., diffset for short). The algorithm significantly reduces runtime and memory usage compared to using traditional transaction ID sets when handling dense data. To elaborate, like Eclat, each domain item is represented by one of the equivalence classes. Unlike Eclat, the corresponding diffset for an item captures which transactions the specific item is absent from. Then, frequent patterns are mined in a level-wise bottom-up fashion. For each subsequent level, the diffset from one level to another captures which addition transactions (if any) are absent when extending a k-itemset to its (k+1)-superset. For example, consider a dataset with three transactions t1, t2 & t3 and three domain items a, b & c. If tidset({b}) = {t1, t2, t3} and tidset({c}) = {t2, t3}, then diffset({b}) = {} and diffset({c}) = {t1} because {b} is not absent from any transaction and {c} is absent from transaction t1. Hence, as tidset({b, c}) = {t2, t3}, the corresponding diffset({b, c}) = {t1}. This implies that transaction t_1 is absent when itemset {b} is extended to become {b, c}.

Regardless of their mining direction (horizontal or vertical), traditional frequent pattern mining captures only the presence or absence of items in sets of frequently occurring items. Similarly, traditional association rule mining captures only the presence or absence of items in antecedents of consequences of interesting association rules. In other words, an item is contained in a transaction 0 or 1 times. For this reason, we can also refer to traditional frequent pattern mining as *Boolean* frequent pattern mining.

However, in many real-world applications, data not only represent the presence or absence of items (e.g., merchandise items, events, or services) but also come with quantities associated with them (e.g., multiple orders of the same merchandise items, multiple tickets for the same events, multiple orders of the same services). This calls for algorithms beyond the traditional frequent pattern mining and/or association rule mining. To address this shortcoming, the notion of quantitative association rule mining or *quantitative* frequent pattern mining [46, 47] was studied. Quantitative frequent pattern mining is essentially an extension of frequent pattern mining to allow transactions to contain an item more than once. Rather than just trying to find items (which commonly occur in transactions), there is a demand for discovering commonly occurring quantities of items. For example, in Boolean frequent itemset mining, we may discover that bananas are a frequently purchased item. In quantitative frequent pattern mining, we may discover that customers frequently purchase at least five bananas at a time. As another example, the quantity of items may also affect profits of selling the items within the discovered patterns.

By discovering quantitative frequent patterns and quantitative association rules, we can obtain more interesting results than we would if Boolean association rule mining were used instead. In addition to receiving information about which items commonly occur together in transactions, we also obtain information regarding how many of each of those

items tend to occur in transactions. MQA-M algorithm [48] extends the Apriori algorithm to mine *quantitative frequent patterns* (aka *itemexpsets*) horizontally.

In this paper, we integrate (a) vertical mining with diffset and (b) quantitative mining for handling dense data. In other words, we incorporate benefits of both dEclat and MQA-M into a single vertical quantitative frequent pattern mining algorithm. Key differences among these algorithms are shown in Table 1. Our *key contributions* to this paper include:

- our design and implementation of the Q-dEclat algorithm,
- our enhanced pruning rules (to avoid recursive execution of pruning rules used in MQA-M algorithm), and
- our further enhancement of direct generation of relevant itemexpsets (to avoid generating and then pruning redundant itemexpsets).

Table 1. Table captions should be placed above the tables.

	MQA-M [48]	Eclat [43]	dEclat [44, 45]	Q-Eclat [46]	Our Q-dEclat
Horizontal mining	✓				
Vertical mining		✓	✓	✓	✓
Can mine sparse data	✓	✓	✓	✓	✓
Efficiently mine dense data			✓		✓
Can mine Boolean frequent patterns	✓	✓	✓	✓	✓
Can mine quantitative freq. patterns	✓			✓	✓

The remainder of this paper is organized as follows. The next section provides background and related works. Section 3 describes our Q-dEclat algorithm. Section 4 reports our evaluation results, and Section 5 draws conclusions.

2 Background and Related Works

2.1 Boolean Vertical Frequent Pattern Mining with the dEclat Algorithm

The dEclat algorithm [44] is an example of vertical item-centric frequent pattern mining algorithms, with which data are represented as a collection of difference sets (i.e., diffsets). Each diffset for an item captures which transactions do not contain the specific item. The presence of the transaction ID in the diffset(X) indicates the item X is absent from the corresponding transaction. Let us discuss the difference between the horizontal transaction database and the vertical transaction database. Horizontal transaction databases refer to the usual representation of transactions, where a set of items is associated with each transaction. The dEclat algorithm represents the transaction database in a "vertical" format. A diffset for an item can represent a transaction database in a vertical format

by adding a transaction ID to the diffset for indicating the absence of the item in the corresponding transaction.

Example 1. Let items a, b & c be three domain items. Consider three transactions t1 = {a, b}, t2 = {a, b, c} and t3 = {b, c} in a horizontal transaction dataset. Then, the corresponding vertical representation of the transaction dataset is diffset({a}) = {t3}, diffset({b}) = {} and diffset({c}) = {t1}. This indicates that item a is absent from transaction t3. Item b is not absent from any of the three transactions, which implies item b is present in all three transactions. Item c is absent from transaction t1. The dEclat algorithm makes use of the vertical representation to mine frequent patterns. Let C_k and L_k be the sets containing candidate and frequent k-itemsets, respectively:

1. First, the dEclat algorithm discovers which itemsets are in L_1. It then computes the support of any candidate 1-itemset by subtracting the number of transaction IDs in its corresponding diffset from the number of transactions. Mathematically, for a singleton $\{x\}$, sup($\{x\}$) = |transactions| – |diffset($\{x\}$)|. For example, sup({a}) = 3 – 1 = 2 and sup({b}) = 3 – 0 = 3.
2. After computing the support for every item occurring in the transaction database, L_1 contains singletons with a support ≥ minsup.
3. After discovering L_1, the main loop of the dEclat algorithm is be executed in a level-wise fashion. Consider the first loop iteration with $k = 2$. The first part of the loop involves generating C_k from L_{k-1} by using the same candidate generation method (i.e., performing a self-join on L_{k-1} and pruning the resulting set). Next, it forms a diffset corresponding to each candidate k-itemset in C_k. Suppose that, for some candidate k-itemset $X \in C_k$, W is a $(k - 2)$-itemset containing the first (k–2) items in X, y is the second last item in X, and z is the last item in X. Then, $X = W \cup \{y\} \cup \{z\}$ and $X' = W \cup \{y\}$, where $X' \subseteq X$ and X can be considered as $X = X' \cup \{z\}$. The algorithm computes the diffset of X as the set difference between diffsets of ($W \cup$ {z}) and ($W \cup$ {y}), i.e., diffset(X) = diffset($W \cup$ {z}) – diffset($W \cup$ {y}). It also captures the difference of extending X' into X = ($X' \cup \{z\}$). For example, diffset({a, b}) = diffset({b}) – diffset({a}) = {}, which implies that no transaction is absent from {a, b} when {a} is extended to become {a, b}. As another example, diffset({b, c}) = diffset({c}) – diffset({b}) = {t1}, which implies that t1 becomes absent from {b, c} when {b} is extended to become {b, c}.
4. Next, it computes the support of each pattern X' in Ck by subtracting the number of transaction IDs in the resulting set difference from the support of X = ($W \cup$ {y} $\cup \{z\}$), i.e., sup(X) = sup($W \cup$ {y} $\cup \{z\}$) = sup($X' \cup \{z\}$) = sup(X') - | diffset(X)|. For example, sup({a, b}) = sup({a}) – |diffset({a, b})| = 2 – 0 = 2, and sup({b, c}) = sup({b}) – |diffset({b, c})| = 3 – 1 = 2.
5. The frequent k-itemsets in Lk are computed as the candidate k-itemsets in C_k having a support ≥ minsup. At the end of a loop iteration, increase k by 1 and continue iterating through the main loop (if necessary). The loop stops iterating when L_{k-1} is empty. The union of all L_k is returned as frequent patterns.

2.2 Horizontal Quantitative Frequent Pattern Mining with the MQA-M Algorithm

The MQA-M (Mining Quantitative Association rules with Multiple comparison operators) [48] is an algorithm for mining quantitative frequent patterns. The MQA-M algorithm is a level-wise bottom-up algorithm generalized to handle quantitative transaction databases.

For any positive integer k, let C_k be the set of candidate itemexpsets containing k itemexps and let L_k be the set of candidate itemexpsets containing k itemexps. Then, $L_k \subseteq C_k$. The MQA-M algorithm starts by generating C_1. Suppose that $item\_\max[p]$ represents the maximum number of times an item p appears in a transaction.

Example 2. If a quantitative dataset consists of three transactions $t1 \supseteq \{(b, 3)\}$, $t2 \supseteq \{(b, 3)\}$ and $t3 \supseteq \{(b, 6)\}$, then itemmax[b] = 6 because the highest number of times a appears in a transaction is 4.

Then, for each item p appearing in the quantitative transaction database, add every itemexpset of the form $\{(p, \otimes, q)\}$ to C_1, where $\otimes \in \{=, \geq, \leq\}$ and $q \in \{1, \ldots, item\_max[p]\}$. The algorithm computes the support of each itemexpset in C_1 by iterating through the transactions and checking each itemexpset in C_1 to see if it should increment the support of that itemexpset. It increments the support of an itemexpset if the transaction satisfies that itemexpset. Let k=1. Then, L_1 becomes the set of all itemexpsets with a support that is at least $minsup$. The algorithm removes some itemexpsets from L_1 using two pruning rules [48]:

1. Suppose that X contains an itemexp of the form $(z \leq r)$, where z is an item and r is a positive integer. The first pruning rule states that if there is another itemexpset Y in L_k with the same support as X which is the same as X except that $(z \leq r)$ is replaced by $(z \leq r+1)$, then Y can be pruned from L_k.
2. Suppose that X contains an itemexp of the form $(z \geq r)$, where z is an item and r is a positive integer. The second pruning rule states that if there is another itemexp-set Y in L_k with the same support as X which is the same as X except that $(z \geq r)$ is replaced by $(z \geq r-1)$, then Y can be pruned from L_k.

The MQA-M has a main loop. It first runs the loop with k=2. The loop body begins with generating C_k from L_{k-1}. C_k is initially generated using a self join on L_{k-1}, like in the Apriori algorithm. If two itemexpsets in L_{k-1} have the same first $(k-2)$ itemexps, then it generates an itemexpset in C_k con-sisting of those $(k-2)$ itemexps and the last itemexp in the two itemexpsets in L_{k-1}. However, it imposes an additional restriction that it does not create an itemexpset in C_k where there are two itemexps referring to the same item.

Example 3. Although L_1 contains $\{(b = 3)\}$ and $\{(b \geq 6)\}$, MQA-M does not form $\{(b = 3), (b \geq 6)\} \in C_2$.

After the join step, it prunes itemexpsets from C_k with a subset containing $(k-1)$ itemexps where that subset is not in L_{k-1}. It gets L_k from C_k using the same procedure that

was used to obtain L_1. Using the two aforementioned pruning rules, it removes some itemexpsets from L_k. At the end of the loop body, it increments k and repeats the previous steps (if necessary). The loop terminates when L_{k-1} is empty. Afterwards, it returns $\bigcup_k L_k$, which contains all the interesting frequent itemexpsets.

3 Q-dEclat for Mining Quantitative Frequent Patterns from Dense Data

3.1 Computing diffsets with Quantity Stored in ItemExpSet

To represent quantitative transaction databases in a vertical format, for each item that occurs in the transaction database, we store it as a set of pairs. Each pair contains a transaction ID associated with that item and the number of occurrences of the item in the transaction. Since we are storing a pair, we can call these sets “pairsets". It is useful to convert the quantitative transaction database to this vertical format when implementing the Q-dEclat algorithm.

Example 4. A horizontal database containing two transactions t1 $\supseteq$ {(b, 3)}, t2 $\supseteq$ {(b, 3)} and t3 $\supseteq$ {(b, 6)} can be represented vertically using pairset(b) = {(t1, 3), (t2, 3), (t3, 6)}.

To mine quantitative frequent patterns, we define diffset (X) of any itemexpset X to be the set of transaction IDs corresponding to transactions which do not satisfy X. When X is an itemexpset containing exactly one itemexp (i.e., X is a singleton), diffset(X) captures a set of transaction IDs in which X is absent from. The support of an itemexpset X—where |X|=1, i.e., singleton {({x}, $\otimes$, q)—can be computed by subtracting the number of transaction IDs in its corresponding diffset from the number of transactions:

$$\text{sup (X)} = |\text{transactions}| - \text{diffset(X)} \quad (1)$$

When X is an itemexpset containing at least two itemexps, we can break down $X = W \cup \{y\} \cup \{z\}$ where (i) W is an itemexpset with two fewer elements than X and (ii) y and z are itemexps. Like diffsets for Boolean frequent itemset mining, we have the recursive equation:

$$\text{diffset}(X) = \text{diffset}(W \cup \{z\}) - \text{diffset}(W \cup \{\text{y}\}) \quad (2)$$

We use this equation to generate diffsets for itemexpsets containing at least two elements when running our Q-dEclat algorithm. The support of an itemexpset X—where |X| > 1, i.e., non-singleton—can be computed by subtracting the number of transaction IDs in the resulting set difference from the support of X = ($W \cup \{\text{y}\} \cup \{z\}$):

$$\text{sup}(X) = \text{sup}(W \cup \{\text{y}\} \cup \{z\}) \quad (3)$$

$$= \sup(X' \cup \{z\}) \quad (4)$$

$$= \sup(X') - |\text{diffset}(X)| \quad (5)$$

3.2 More Powerful Generalized Pruning Rules

Observed from Section 2.2 that, while the two pruning rules help remove some itemexp-sets from L_k, they keep a number of unnecessary or uninteresting itemexpsets. To elaborate, these two pruning rules remove the neighboring itemexpset Y of X if its support is the same as support of X (i.e., if sup(Y) = sup(X)). To prune more than just one itemexp-set (say, the previous/next N neighboring itemexpsets), one may need to apply these two pruning rules recursively.

To speed up the mining process by pruning those unnecessary itemexpsets (e.g., the N preceeding/proceeding neighboring itemexpsets) and empowering the pruning rules, we modified the two pruning rules to become the following:

1'. Suppose that X contains an itemexp of the form $(z \leq r)$, where z is an item and r is a positive integer. The first pruning rule states that if there is another itemexpset Y in L_k with the same support as X which is the same as X except that $(z \leq r)$ is replaced by $(z \leq r+s)$ for some positive integer s, then Y can be pruned from L_k.
2'. Suppose that X contains an itemexp of the form $(z \geq r)$, where z is an item and r is a positive integer. The second pruning rule states that if there is another itemexp-set Y in L_k with the same support as X which is the same as X except that $(z \geq r)$ is replaced by $(z \geq r-s)$ for some positive integer s, then Y can be pruned from L_k.

The difference between the original pruning rules and our new pruning rules is that the new pruning rules can handle differences in quantity greater than 1. Instead of considering itemexpsets of the form $(z \leq r+1)$ or $(z \geq r-1)$, we consider the more general cases of $(z \leq r+s)$ or $(z \geq r-s)$ for some positive integer s. As a result, these rules eliminate at least as many itemexpsets from L_k as the original pruning rules.

Example 5. Suppose that L_2 contains $\{(a=4), (b\geq3)\}$ and $\{(a=4), (b\geq6)\}$ before pruning and that those itemexpsets have the same support. Using the original pruning rules 1 & 2 that were used in MQA-M, neither itemexpset would be pruned. However, using the new pruning rules 1' & 2', we would be able to prune $\{(a=4), (b\geq3)\}$ from L_2.

3.3 Further Enhancement by Directly Generating Itemexpsets with Quantities Only at the Borders

We further observed that, even with our new pruning rules 1' & 2', lots of itemexpsets are needed to be generated and then pruned. Many of these itemexpsets do not need to be generated in the first place. More precisely, changes of support occur on the “borders”. See Example 6.

Example 6. Consider a transactional dataset with three transactions: t1 = {a:4, b:3}, t2 = {a:4, b:3, c:2}, and t3 = {b:6, c:1}. Then, itemmax[a] = 4, itemmax[b] = 6 and itemmax[c] = 2. With the original pruning rules 1 & 2 (or our enhanced pruning rules 1' & 2'), one would generate:

- 4×3=12 itemexpsets containing a: {(a=1)}, …, {(a=4)}, {(a≥1)}, …, {(a≥4)}, {(a≤1)}, …, {(a≤4)}.
- 6×3=18 itemexpsets containing b: {(b=1)}, …, {(b=6)}, {(b≥1)}, …, {(b≥6)}, {(b≤1)}, …, {(b≤4)}.
- 2×3=6 itemexpsets containing c: {(c=1)}, {(c=2)}, {(c≥1)}, {(c≥2)}, {(c≤1)}, {(c≤2)}.

Hence, a total of 12 + 18 + 6 = 36 itemexpsets are generated. They can be clustered based on their similarity (itemexpsets and support):

- 3 itemexpsets {(a=1)}, …, {(a=3)}, all with sup=0.
- 1 itemexpset {(a=4)}, with sup=2.
- 4 itemexpsets {(a≥1)}, …, {(a≥4)}, all with sup=2.
- 3 itemexpsets {(a≤1)} …, {(a≤3)}, all with sup=0.
- 1 itemexpset {(a≤4)}, with sup=2.
- 2 itemexpsets {(b=1)} and {(b=2)}, both with sup=0.
- 1 itemexpset {(b=3)}, with sup=2.
- 2 itemexpsets {(b=4)} and {(b=5)}, both with sup=0.
- 1 itemexpset {(b=6)}, with sup=1.
- 3 itemexpsets {(b≥1)}, …, {(b≥3)}, all with sup=3.
- 3 itemexpsets {(b≥4)}, …, {(b≥6)}, all with sup=1.
- 2 itemexpsets {(b≤1)} and {(b≤2)}, both with sup=0.
- 2 itemexpsets {(b≤3)} and {(b≤4)}, both with sup=2.
- 2 itemexpsets {(b≤5)} and {(b≤6)}, both with sup=3.
- 1 itemexpset {(c=1)}, with sup=1.
- 1 itemexpset {(c=2)}, with sup=1.
- 1 itemexpset {(c≥1)}, with sup=2.
- 1 itemexpset {(c≥2)}, with sup=1.
- 1 itemexpset {(c≤1)}, with sup=1.
- 1 itemexpset {(c≤2)}, with sup=2.

There are 20 clusters of itemexpsets. In other words, out of 36 generated itemexpsets, 16 of them can be pruned by recursively applying the original pruning rules 1 & 2 (or by applying the enhanced pruning rules 1' & 2'). Consequently, 20 of them are left:

- 3 itemexpsets {(a=1)}, …, {(a=3)}, all with sup=0.
- 1 itemexpset {(a=4)}, with sup=1.
- *1 itemexpset {(a≥4)}, with sup=2.*

- *1 itemexpset {(a≤1)}, with sup=0.*
- 1 itemexpset {(a≤4)}, with sup=2.
- 2 itemexpsets {(b=1)} and {(b=2)}, both with sup=0.
- 1 itemexpset {(b=3)}, with sup=2.
- 2 itemexpsets {(b=4)} and {(b=5)}, both with sup=0.
- 1 itemexpset {(b=6)}, with sup=1.
- *1 itemexpset {(b≥3)}, with sup=3.*
- *1 itemexpset {(b≥6)}, with sup=1.*
- *1 itemexpset {(b≤1)}, with sup=0.*
- *1 itemexpset {(b≤3)}, with sup=2.*
- *1 itemexpset {(b≤5)}, with sup=3.*
- 1 itemexpset {(c=1)}, with sup=1.
- 1 itemexpset {(c=2)}, with sup=1.
- 1 itemexpset {(c≥1)}, with sup=2.
- 1 itemexpset {(c≥2)}, with sup=1.
- 1 itemexpset {(c≤1)}, with sup=1.
- 1 itemexpset {(c≤2)}, with sup=2.

Furthermore, 5 of these 20 clusters (of a single itemexpset each) are with zero support and thus can be pruned. This results in 15 clusters or itemexpsets:

- 1 itemexpset {(a=4)}, with sup=1.
- 1 itemexpset {(a≥4)}, with sup=2.
- 1 itemexpset {(a≤4)}, with sup=2.
- 1 itemexpset {(b=3)}, with sup=2.
- 1 itemexpset {(b=6)}, with sup=1.
- 1 itemexpset {(b≥3)}, with sup=3.
- 1 itemexpset {(b≥6)}, with sup=1.
- 1 itemexpset {(b≤3)}, with sup=2.
- 1 itemexpset {(b≤5)}, with sup=3.
- 1 itemexpset {(c=1)}, with sup=1.
- 1 itemexpset {(c=2)}, with sup=1.
- 1 itemexpset {(c≥1)}, with sup=2.
- 1 itemexpset {(c≥2)}, with sup=1.
- 1 itemexpset {(c≤1)}, with sup=1.
- 1 itemexpset {(c≤2)}, with sup=2.

However, quantities associated with these 15 itemexpsets are the quantities on the borders. They are quantities appear in the database, instead of any quantities in between.

Based on the observation from Example 6, our enhancement is to *directly* generate these itemexpsets from the vertical representation of the database instead of the generate-and-prune paradigm. See Example 7.

Example 7. Consider a transactional dataset with three transactions: t1 = {a:4, b:3}, t2 = {a:3, b:3, c:2}, and t3 = {b:6, c:1}. Then, we *directly* generate the same 15 itemexpsets as in Example 6.

3.4 Algorithm Details

After describing key components of our algorithm, let us the Q-dEclat algorithm for vertically mining quantitative frequent patterns from dense data. For any integer $k \geq 1$, define C_k to be the set of candidate k-itemexpsets and L_k to be the set of frequent k-itemexpsets. First, we convert the quantitative transaction database into a vertical format if it is in its horizontal format. The vertical format is useful for computing the diffsets corresponding to the candidate 1-itemexpsets (i.e., C_1). The next step of our algorithm is to compute all candidate 1-itemexpsets in C_1. Each of those itemexpsets consists of a single itemexp of the form (item, operation, quantity) where:

- item is an item in the transaction database,
- operator $\theta \in \{=, \geq, \leq\}$, and
- quantity $q \in \{1, \ldots, \text{itemmax[item]}\}$.

After computing $C1$, we compute the diffsets associated with each candidate 1-itemexpset. The diffsets can be computed from the vertical representation of the quantitative transaction database. We then compute the support of each candidate 1-itemexpset as per Eq. (1). The frequent 1-itemexpsets are candidate 1-itemexpsets having a support $\geq$ minsup.

Then, we set $k = 2$ and begin executing the main loop. The first step in the main loop body is to generate C_k using L_{k-1}. We initially create C_k by performing a self-join on L_{k-1}. If there are two frequent $(k-1)$-itemexpsets in L_{k-1} where their first $(k-2)$-itemexps are the same and their last itemexp refer to different items, then we add to C_k a candidate k-itemexpset that consists of the first $(k-2)$-itemexps and the last itemexp of both itemexpsets. Afterwards, we prune any candidate k-itemexpset from C_k that contains a sub-itemexpset with $(k-1)$-itemexps that do not belong to L_{k-1}. The next step is to create diffsets corresponding to every candidate k-itemexpset in C_k. This can be done using the recursive definition for diffsets as per Eq. (2).

After computing the diffsets, we compute the support of each candidate k-itemexpset in C_k as per Eq. (3). Any candidate k-itemexpset having a support $\geq$ minsup is added to L_k. Using the two pruning rules, we remove some uninteresting itemexpsets from L_k if necessary. After pruning L_k, we have reached the end of the loop body. Hence, we increment k and repeat the main steps again if necessary. The main loop stops running once L_k is emp-

ty. Our Q-dEclat algorithm returns union of L_k, which contains all interesting frequent itemexpsets. See Example 8 for an illustrative example.

Example 8. Continue with Example 7. Consider a transactional dataset with three transactions: t1 = {a:4, b:3}, t2 = {a:4, b:3, c:2}, and t3 = {b:6, c:1}. Let minsup=1. Then, our Q-dEclat *directly* generate the following 9 itemexpsets as described in Section 3.3 and compute their corresponding diffsets and support as per Eq. (1):

- {(a=4)}, with diffset={t1, t2, t3}–{t1, t2}={t3} and sup=3–1=2.
- {(a≥4)}, with diffset={t3} and sup=2.
- {(a≤4)}, with diffset={t3} and sup=2.
- {(b=3)}, with diffset={t1, t2, t3}–{t1, t2}={t3} and sup=3–1=2.
- {(b≤3)}, with diffset={t3} and sup=2.
- {(b=6)}, with diffset={t1, t2} and sup=1.
- {(b≥6)}, with diffset={t1. t2} and sup=1.
- {(b≥3)}, with diffset={} and sup=3.
- {(b≤6)}, with diffset={} and sup=3.
- {(c=1)}, with diffset={t1, t2, t3}–{t3}={t1, t2} and sup=3–2=1.
- {(c=2)}, with diffset={t1, t3} and sup=1.
- {(c≥2)}, with diffset={t1, t3} and sup=1.
- {(c≥1)}, with diffset={t1} and sup=2.
- {(c≤2)}, with diffset={t1} and sup=2.
- {(c≤1)}, with diffset={t1, t2} and sup=1.

Among these 15 candidate 1-itemexpsets, all of them satisfy minsup = 1. We obtain L_1 by only keeping these 15 candidate 1-itemexpsets having support ≥ minsup. They can be simplified into the following clusters of frequent 1-itemexpsets:

- 3 itemexpsets of the form {(a θ 4)} where θ ∈ {=, ≥, ≤}, with diffset={t3} and sup=2.
- 2 itemexpsets {(b θ' 3)} where θ' ∈ {=, ≤}, with diffset={t3} and sup=2.
- 2 itemexpsets {(b θ" 6)} where θ" ∈ {=, ≥}, with diffset={t1, t2} and sup=1.
- {(b≥3)}, with diffset={} and sup=3.
- {(b≤6)}, with diffset={} and sup=3.
- {(c=1)}, with diffset={t1, t2} and sup=1.
- 2 itemexpsets{(c θ" 2)} where θ" ∈ {=, ≥}, with diffset={t1, t3} and sup=1.
- {(c≥1)}, with diffset={t1} and sup=2.
- {(c≤2)}, with diffset={t1} and sup=2.
- {(c≤1)}, with diffset={t1, t2} and sup=1.

Then, the main loop is executed with $k = 2$. We begin with the generation of C_2. The first step in generating C_2 is to perform a self-join on L_1. In this scenario, this means getting pairs of itemexps where the itemexps refer to different items. These yields $(3 \times 6) + (3 \times 6) + (6 \times 6) = 72$ different candidate 2-itemexpsets in C_2. Among these candidate 2-

itemexpsets, only 52 of them satisfy minsup = 1. We obtain initial L_2 by only keeping these candidate 2-itemexpsets having support ≥ minsup:

- 6 itemexpsets of the form {(a θ 4), (b θ' 3)} where θ ∈ {=, ≥, ≤} and θ' ∈ {=, ≤}, with diffset={t3}–{t3}={} and sup=2–0=2.
- 3 itemexpsets {(a θ 4), (b≥3)} where θ ∈ {=, ≥, ≤}, with diffset={} and sup=2–0=2.
- 3 itemexpsets {(a θ 4), (b≤6)} where θ ∈ {=, ≥, ≤}, with diffset={} and sup=2–0=2.
- 6 itemexpsets {(a θ 4), (c θ" 2)} where θ ∈ {=, ≥, ≤} and θ" ∈ {=, ≥}, with diffset={t1} and sup=2–1=1.
- 3 itemexpsets {(a θ 4), (c≥1)} where θ ∈ {=, ≥, ≤}, with diffset={t1} and sup=2–1=1.
- 3 itemexpsets {(a θ 4), (c≤2)} where θ ∈ {=, ≥, ≤}, with diffset={t1} and sup=2–1=1.
- 4 itemexpsets {(b θ' 3), (c θ" 2)} where θ' ∈ {=, ≤} and θ" ∈ {=, ≥}, with diffset={t1} and sup=2–1=1.
- 2 itemexpsets {(b θ' 3), (c≥1)} where θ' ∈ {=, ≤}, with diffset={t1} and sup=2–1=1.
- 2 itemexpsets {(b θ' 3), (c≤2)} where θ' ∈ {=, ≤}, with diffset={t1} and sup=2–1=1.
- 2 itemexpsets {(b θ" 6), (c=1)} where θ" ∈ {=, ≥}, with diffset={} and sup=1–0=1.
- 2 itemexpsets {(b θ" 6), (c≥1)} where θ" ∈ {=, ≥}, with diffset={} and sup=1–0=1.
- 2 itemexpsets {(b θ" 6), (c≤2)} where θ" ∈ {=, ≥}, with diffset={} and sup=1–0=1.
- 2 itemexpsets {(b θ" 6), (c≤1)} where θ" ∈ {=, ≥}, with diffset={} and sup=1–0=1.
- {(b≥3), (c=1)}, with diffset={t1, t2} and sup=3–2=1.
- 2 itemexpsets {(b≥3), (c θ" 2)} where θ" ∈ {=, ≥}, with diffset={t1, t3} & sup=3–2=1.
- {(b≥3), (c≥1)}, with diffset={t1} and sup=3–1=2.
- {(b≥3), (c≤2)}, with diffset={t1} and sup=3–1=2.
- {(b≥3), (c≤1)}, with diffset={t1, t2} and sup=3–2=1.
- {(b≤6), (c=1)}, with diffset={t1, t2} and sup=3–2=1.
- 2 itemexpsets {(b≤6), (c θ" 2)} where θ" ∈ {=, ≥}, with diffset={t1, t3} & sup=3–2=1.
- {(b≤6), (c≥1)}, with diffset={t1} and sup=3–1=2.
- {(b≤6), (c≤2)}, with diffset={t1} and sup=3–1=2.
- {(b≤6), (c≤1)}, with diffset={t1, t2} and sup=3–2=1.

Note that 7 of these 52 itemexpsets in the initial L_2 can be pruned by the original pruning rules used in the MQA-M algorithm:

- 3 itemexpsets of the form {(a θ 4), (c≥1)}, by original pruning rule 2 due to {(a θ 4), (c≥2)}, where θ ∈ {=, ≥, ≤}.
- 2 itemexpsets {(b θ' 3), (c≥1)}, by original pruning rule 2 due to {(b θ' 3), (c≥2)}, where θ' ∈ {=, ≤}.
- 2 itemexpsets {(b θ" 6), (c≤2)}, by original pruning rule 1 due to {(b θ" 6), (c≤1)}, where θ" ∈ {=, ≥}.

Moreover, by using our enhanced pruning rules described in Section 3.2, we can further prune away 7 more redundant 2-itemexpsets:

- 3 itemexpsets of the form {(a θ 4), (b≤6)}, by enhanced pruning rule 1' due to {(a θ 4), (b≤3)}, where θ ∈ {=, ≥, ≤}.
- {(b≥3), (c=1)}, by enhanced pruning rule 2' due to {(b≥6), (c=1)}.
- {(b≥3), (c≤1)}, by enhanced pruning rule 2' due to {(b≥6), (c≤1)}.
- 2 itemexpsets {(b≤6), (c θ" 2)} where θ" ∈ {=, ≥}, by enhanced pruning rule 1' due to {(b≤3), (c θ" 2)}.

Consequently, after applying these original and enhanced pruning rules, we get the following 52 – 7 – 7 = 38 non-redundant 2-itemexpsets:

- 6 itemexpsets of the form {(a θ 4), (b θ' 3)} where θ ∈ {=, ≥, ≤} and θ' ∈ {=, ≤}, with diffset={} and sup=2–0=2.
- 3 itemexpsets {(a θ 4), (b≥3)} where θ ∈ {=, ≥, ≤}, with diffset={} and sup=2–0=2.
- 6 itemexpsets {(a θ 4), (c θ" 2)} where θ ∈ {=, ≥, ≤} and θ" ∈ {=, ≥}, with diffset={t1} and sup=2–1=1.
- 3 itemexpsets {(a θ 4), (c≤2)} where θ ∈ {=, ≥, ≤}, with diffset={t1} and sup=2–1=1.
- 4 itemexpsets {(b θ' 3), (c θ" 2)} where θ' ∈ {=, ≤} and θ" ∈ {=, ≥}, with diffset={t1} and sup=2–1=1.
- 2 itemexpsets {(b θ' 3), (c≤2)} where θ' ∈ {=, ≤}, with diffset={t1} and sup=2–1=1.
- 2 itemexpsets {(b θ" 6), (c=1)} where θ" ∈ {=, ≥}, with diffset={} and sup=1–0=1.
- 2 itemexpsets {(b θ" 6), (c≥1)} where θ" ∈ {=, ≥}, with diffset={} and sup=1–0=1.
- 2 itemexpsets {(b θ" 6), (c≤1)} where θ" ∈ {=, ≥}, with diffset={} and sup=1–0=1.
- 2 itemexpsets {(b≥3), (c θ" 2)} where θ" ∈ {=, ≥}, with diffset={t1, t3} & sup=3–2=1.
- {(b≥3), (c≥1)}, with diffset={t1} and sup=3–1=2.
- {(b≥3), (c≤2)}, with diffset={t1} and sup=3–1=2.
- {(b≤6), (c=1)}, with diffset={t1, t2} and sup=3–2=1.
- {(b≤6), (c≥1)}, with diffset={t1} and sup=3–1=2.
- {(b≤6), (c≤2)}, with diffset={t1} and sup=3–1=2.
- {(b≤6), (c≤1)}, with diffset={t1, t2} and sup=3–2=1.

Afterwards, Q-dEclat forms 27 candidate 3-itemexpsets (i.e., $k = 3$). We begin with the generation of C_3 by joining frequent 2-itemexpsets that share common (k–1)-prefix:

- 12 itemexpsets of the form {(a θ 4), (b θ' 3), (c θ" 2)} where θ ∈ {=, ≥, ≤}, θ' ∈ {=, ≤} and θ" ∈ {=, ≥}, with diffset={t1}–{}={t1} and sup=2–1=1.
- 6 itemexpsets of the form {(a θ 4), (b θ' 3), (c≤2)} where θ ∈ {=, ≥, ≤} and θ' ∈ {=, ≤}, with diffset={t1}–{}={t1} and sup=2–1=1.
- 6 itemexpsets {(a θ 4), (b≥3), (c θ" 2)} where θ ∈ {=, ≥, ≤} and θ" ∈ {=, ≥}, with diffset={t1}–{}={t1} and sup=2–1=1.
- 3 itemexpsets {(a θ 4), (b≥3), (c≤2)} where θ ∈ {=, ≥, ≤}, with diffset={t1}–{}={t1} and sup=2–1=1.

All these 27 candidate 3-itemexpsets are frequent and become L_3. As there are only three domain items, no candidate 4-itemexpsets can be formed. Hence, at the end of this quantitative frequent pattern mining process, our Q-dEclat discovers 15 frequent non-redundant 1-itemexpsets, 38 frequent non-redundant 2-itemexpsets and 27 frequent non-redundant 3-itemexpsets, for a total of 80 frequent itemexpsets.

4 Evaluation

To evaluate our Q-dEclat algorithm, we compared it with the existing MQA-M algorithm [48] and Q-Eclat [46] (our variant of Q-dEclat algorithm, with dEclat replaced by Eclat). Recall from Sections 1 & 2.2 that MQA-M algorithm is a transaction-centric horizontal algorithm for mining quantitative frequent patterns. Recall from Sections 1 & 2 that Q-Eclat is an item-centric vertical algorithm designed to mine quantitative frequent patterns from *sparse* data. Inherited from the Eclat algorithm (for mining Boolean frequent patterns), the Q-Eclat algorithm uses transaction ID sets (tidsets) to capture transactions in which an itemexpset is present. It works well for sparse data. However, when datasets become denser, the corresponding tidsets become bigger and thus increases computation and memory consumption. In contrast, our Q-dEclat algorithm is designed to mine quantitative frequent patterns from *dense* data. Inherited from the dEclat algorithm (for mining Boolean frequent patterns), our Q-dEclat algorithm uses difference sets (diffsets) to capture changes in transactions from which an itemexpset is absent when extended a quantitative frequent k-itemexpset to a quantitative candidate (k+1)-itemexpset. Hence, when datasets become denser, the corresponding diffsets are usually smaller and thus decreases computation and memory consumption. The performance of the algorithms is assessed using three different quantitative transaction databases:

- A dense synthetic dataset: Here, we assume that there are n transactions and $|I|$ different items. Each item has a probability prob of occurring in a particular transaction, where $0 \leq \text{prob} \leq 1$. If the item appears in the transaction, then the number of occurrences of that item follows a Poisson(λ) distribution plus 1. We set $n = 1000$, $|I| = 50$, and $\lambda = 1$. The value of prob for the quantitative transaction database is 0.8 for a dense dataset.
- A modified real-life chess dataset from UCI ML Repository [49]: Here, we modified the chess dataset to make them quantitative transaction databases. Whenever an item occurs in a transaction, instead of it only occurring once, its number of occurrences follows a Poisson(λ) distribution plus 1.
- Another modified real-life mushroom dataset: We applied the same modification procedure to the mushroom dataset from UCI ML Repository.

All algorithms for quantitative frequent itemset mining are implemented in the Python language. The algorithms were run on a Windows 10 Nitro AN515-55 laptop using an

Intel Core i5-10300H CPU at 2.50 GHz and 8.00 GB RAM. To keep the comparisons between the algorithms fair, we used many of the same functions between the algorithms. These include generation of candidate itemexpsets, discovery of frequent itemexpsets, and application of our pruning rules on the frequent itemexpsets. When we implement the MQA-M algorithm, we use our enhanced pruning rules and direct generation of singleton itemexpsets used in Q-dEclat rather than the pruning rules originally used with MQA-M. This allows the simulations to emphasize the differences between the algorithms. We run the main code for each of the aforementioned quantitative transaction datasets. For each quantitative transaction database, we use a sequence of minsup values. The sequence depends on the quantitative transaction database being used to observe interesting results that the algorithms did not take too long to run. For each combination of a quantitative transaction database and a value for minsup, the algorithms were run and timed. Reported runtimes were average of multiple runs. Figures 1-3 show the runtimes of each of the algorithms for a variety of values of minsup for each of the four quantitative transaction datasets. The runtime (in seconds) is shown on the y-axis while the value of minsup is given on the x-axis. In all cases, our Q-dEclat took shortest runtimes to mine from dense synthetic or real-life datasets. Q-dEclat algorithm to return the same collections of itemsexpsets.

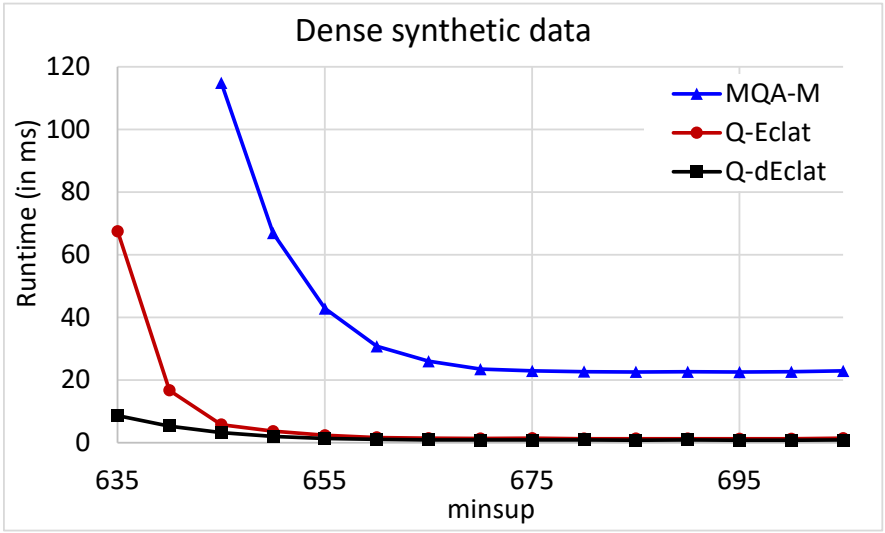


Fig. 1. Runtime comparisons of our Q-dEclat vs. existing Q-Eclat and MQA-M algorithms on dense synthetic data.

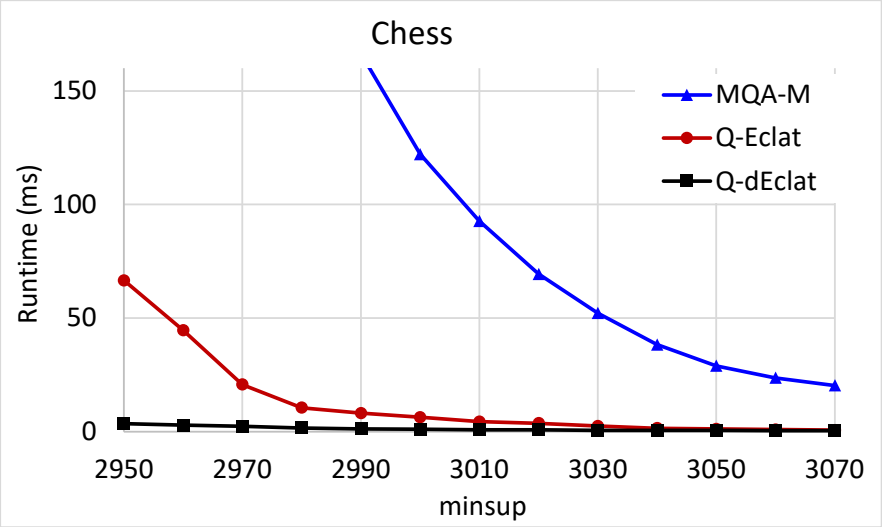


Fig. 2. Runtime comparisons of our Q-dEclat vs. existing Q-Eclat and MQA-M algorithms on (dense) real-life chess data.

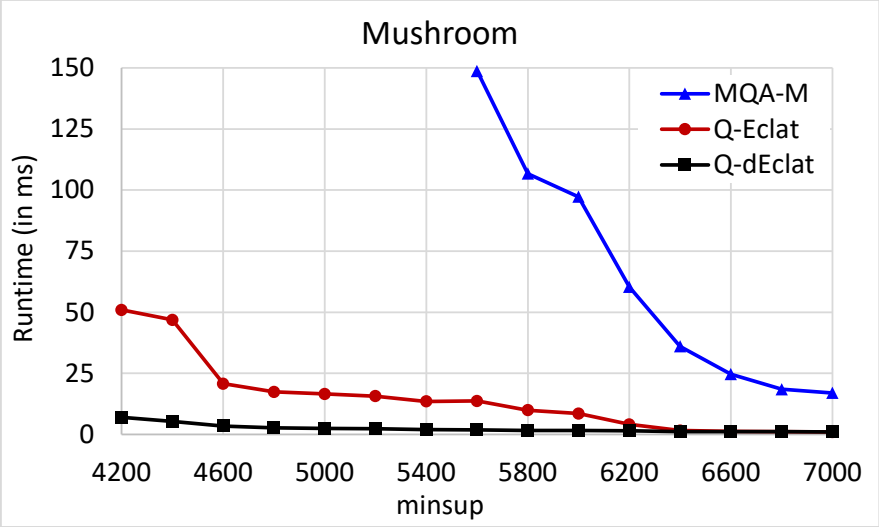


Fig. 3. Runtime comparisons of our Q-dEclat vs. existing Q-Eclat and MQA-M algorithms on (dense) real-life mushroom data.

5 Conclusions

In this paper, we designed and presented an algorithm—called Q-dEclat—that vertically mines quantitative frequent patterns from dense data. The algorithm can be considered as a non-trivial integration of quantitative frequent pattern mining and vertical frequent pattern mining. To elaborate, the algorithm first represents the big data as a collection of difference sets (diffsets), which capture the transactions from which the item is absent from for the singletons. Then, the diffsets capture the difference between two item expression sets (itemexpsets) when extending a set to its immediate superset. In other words, the diffsets capture the transactions from which the item is further absent from the set extensions. To mine quantitative frequent patterns, we adapted existing MQA-M algorithm. Observing the drawbacks of the pruning rules used in the MQA-M algorithm, we enhanced the pruning rules. Moreover, to further enhance the mining process. Based on another observation, Q-dEclat direct generates singleton itemexpsets, which avoids the time-consuming generate-and-prune process for singletons. Then, our two enhanced pruning rules effectively prune redundant itemexpsets when mining frequent non-singleton patterns. Evaluation results show the superiority of our Q-dEclat algorithm. As *ongoing and future work*, we would apply our Q-dEclat algorithm to many real-life database engineered applications, where data are usually dense.

Acknowledgments

This work is partially supported by (i) Natural Sciences and Engineering Research Council of Canada (NSERC) and (ii) University of Manitoba.

References

1. Chbeir, R., et al. (eds.): Database Engineered Applications. IDEAS 2024. LNCS, vol. 15511. Springer, Cham (2024)
2. Bergami, G., et al. (eds.) Database Engineered Applications. IDEAS 2025. LNCS, vol. 15928. Springer, Cham (2025)
3. Wegrzyn, D.T.: The adaptation of the OODA loop to the decision-making systems processing Big Data in the area of morality. In: IDEAS 2022, 144-149.
4. Sahri, S., Moussa, R.: Customized eager-lazy data cleansing for satisfactory big data veracity. In: IDEAS 2021, 157-165.
5. Zhao, Y., et al.: A zone-based data lake architecture for IoT, small and big data. In: IDEAS 2021, 94-102.
6. Cady, F.: The Data Science Handbook, 2nd edn. Wiley (2024)
7. Leung, C.K., et al.: Data science for healthcare predictive analytics. In: IDEAS 2020, 8:1-8:10.

8. Revesz, P.Z.: Data science applied to discover ancient Minoan-Indus valley trade routes implied by common weight measures. In: IDEAS 2022, 150-155.
9. Sammut, C., et al. (eds.): Encyclopedia of Machine Learning and Data Science. Springer, New York (2021)
10. Wang, J., et al. (ed.): Encyclopedia of Data Science and Machine Learning. IGI Global (2023)
11. Lee, W., et al.: Mobile web navigation in digital ecosystems using rooted directed trees. IEEE Trans. Ind. Electron. 58(6), 2154-2162 (2011)
12. Leung, C.K., et al.: Explainable data analytics for disease and healthcare informatics. In: IDEAS 2021, 12:1-12:10.
13. Leung, C.K., et al.: Smart data analytics on COVID-19 data. In: IEEE iThings-GreenCom-CPSCom-SmartData-Cybermatics 2021, 372-379.
14. Nesca, M., et al.: A scoping review of preprocessing methods for unstructured text data to assess data quality. Int. J. Popul. Data Sci. 1(7), 19:1-19:5 (2022)
15. Alam, M.T., et al.: Discovering interesting patterns from hypergraphs. ACM Trans. Knowl. Discov. Data 18(1), 32:1-32:34 (2024)
16. Leung, C.K.: Pattern mining for knowledge discovery. In: IDEAS 2019: 34:1-34:5.
17. Roy, K.K., et al.: Mining sequential patterns in uncertain databases using hierarchical index structure. In: PAKDD 2021, Part II. LNCS (LNAI), vol. 12713, 29-41.
18. Froese, R., et al.: The border k-means clustering algorithm for one dimensional data. In: IEEE BigComp 2022, 35-42.
19. Madill, E., et al.: ScaleSFL: a sharding solution for blockchain-based federated learning. In: ACM BSCI 2022, 95-106.
20. Leung, C.K., et al.: FIsViz: a frequent itemset visualizer. In: PAKDD 2008. LNCS (LNAI), vol. 5012, 644-652.
21. Leung, C.K., et al.: WiFIsViz: effective visualization of frequent itemsets. In: IEEE ICDM 2008, 875-880.
22. Zhou, X., et al.: Deep learning-empowered big data analytics in biomedical applications and digital healthcare. IEEE/ACM Trans. Comput. Biol. Bioinform. 21(4): 516-520 (2024)
23. Hao, B., et al.: A computer-aided change detection system for paediatric acute intracranial haemorrhage. In: C3S2E 2008, 109-111.
24. Leung, C.K., et al.: Big data science on COVID-19 data. In: IEEE BigDataSE 2020, 14-21.
25. Leung, C.K., Zhao, C.: Big data intelligence solution for health analytics of COVID-19 data with spatial hierarchy. In: IEEE DataCom 2021, 13-20.
26. Kolisnyk, M., et al.: Analysis of multi-dimensional road accident data for disaster management in smart cities. In: IEEE IRI 2022, 43-48.
27. Bernhard, S.D., et al.: Clickstream prediction using sequential stream mining techniques with Markov chains. In: IDEAS 2016, 24-33.
28. Choudhery, D., Leung, C.K.: Social media mining: prediction of box office revenue. In: IDEAS 2017, 20-29.
29. Hryhoruk, C.C.J., et al.: A scalable framework for social media mining to detect mental health disorders. In: IEEE SWC 2025, 1920-1927.
30. Cameron, J.J., et al.: Finding strong groups of friends among friends in social networks. In: IEEE DASC 2011, 824-831.
31. D'Souza, R.R., et al.: Discovery of patent influence with directed acyclic graph network analysis. In: IDEAS 2023, 17-24.

32. Jiang, F., et al.: Big social network mining for "following" patterns. In: C3S2E 2015, 28-37
33. Lee, W., et al.: A network-flow based influence propagation model for social networks. In: CGC 2012, 601-608.
34. Leung, C.K., et al.: Mining 'following' patterns from big sparse social networks. In: IEEE/ACM ASONAM 2016, 923-930.
35. Leung, C.K., et al.: Visual analytics of social networks: mining and visualizing co-authorship networks. In: HCII-FAC 2011. LNCS (LNAI), vol. 6780, 335-345.
36. Tanbeer, S.K., et al.: Interactive mining of strong friends from social networks and its applications in e-commerce. J. Organ. Comput. Electron. Commer. 24(2-3), 157-173 (2014)
37. Bian, F., et al.: A database engineered system for big data analytics on tornado climatology. In: IDEAS 2024, 172-185.
38. Mateo, M.A.F., et al.: Design and development of a prototype system for detecting abnormal weather observations. In: C3S2E 2008, 45-59.
39. Chowdhury, M.E.S., et al.: A new approach for mining correlated frequent subgraphs. ACM Trans. Manag. Inf. Syst. 13(1), 9:1-9:28 (2022)
40. Lakshmanan, L.V.S., et al.: The segment support map: Scalable mining of frequent itemsets. ACM SIGKDD Explor. 2(2), 21-27 (2000)
41. Agrawal, R., et al.: Mining association rules between sets of items in large databases. In: ACM SIGMOD 1993, 207-216.
42. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB 1994, 487-499.
43. Zaki, M.J.: Scalable algorithms for association mining. IEEE Trans. Knowl. Data Eng. 12(3), 372-390 (2000)
44. Zaki, M.J., Gouda, K.: Fast vertical mining using diffsets. In: ACM KDD 2003, 326-335.
45. Trieu, T.A., et al.: An improvement for dEclat algorithm. In: ICUIMC 2012, 54:1-54:6.
46. Czubryt, T.J., et al.: Q-Eclat: vertical mining of interesting quantitative patterns. In: IDEAS 2022, 25-33.
47. Srikant, R., Agrawal, R.: Mining quantitative association rules in large relational tables. In: ACM SIGMOD 1996, 1-12.
48. Hsu, P.Y., et al.: Algorithms for mining association rules in bag databases. Inf. Sci. 166(1-4), 31-47 (2004)
49. Kelly, M., et al.: The UCI machine learning repository, https://archive.ics.uci.edu

Failures of Technology

Bipin C. DESAI[0000−0002−9142−7928]

Concordia University, Montreal, Canada
BipinC.Desai@concordia.ca

Abstract. In this paper we look at the failure of technology over the last half centurey that brought in the development of computing, the interconenction of it using the internet and the introduction of cell phones, This was followed by the emergence of the world wide web and a graphical interface to it. This was used by USian techs to set up free email service followed by on-line social networks. To this was added chat bots introduced by new USian tech companies and the existing big-techs. We look at the exploitation of these systems not only by these big-techs but also bad actors who are tolerated by the same big-techs. Finally we suggest a possible alternative of setting up national systems to replace them; this would eliminate the need for giant data centers and reduce the need for power to operate them and their pollution. Furthermore, this would result in the amount and distance of data transmission and provide national control over data and its privacy.

Keywords: world wide web · search engines · free email · OSN · Artificail Intelligence · weapon systems

1 Introduction

Over the last century, progress in technology and it's exploitation has allowed great changes in the way humans interact and live. The progress in electronics and its micro/nano -miniaturization led to the development of more and more powerful computer systems and storage devices. Concurrently there has been an emergence of programming languages and systems for specific types of software development, system software, databases, algorithms and the fusion of the later two into artificial intelligence.

The start of the idea of the Web is said to have been born in 1989 with the concept of HTML(94). The web was placed in the public domain with great expectations of free sharing during a conference hosted by CERN, Geneva called World Wide Web I(99). This event, in hind- sight, is sometimes referred to as the Woodstock of the web. The web with its text and graphical browsers and their derivatives, have revolutionized the internet. For most people, the internet is the web, while some of the USian monopolist tech-corporations want the world to view their platforms to be web! The web has given rise to a number of rich powerful USian tech-monopolies which did not exist before the advent of the web.

The easy to use graphical interface and the cell phone with its tiny screen have become the de facto interface to all kinds of applications and have provided new methods of communication and connections. The control of all this by a small number of monopolistic corporations, who have amassed fortunes and vast quantities of data on people, has betrayed the promise of sharing and providing information, freely.

In this paper, we point out how all these technologies have failed to improve the lot of humans, They have not improved the distribution of the rewards of the new technology and have eroded not only human rights, won after over a century of struggle, but also democracy.

2 Exploitation of Web, Search Engines

The last decades of the 20th century show increasing attempts to popularize the use of the internet for sharing. One of the uses was the sharing of information in the form of files and file sharing programs and file search tools(96), (97). Attempts were underway in the late 1980s and early 1990s to provide easier means for a lay person to share information. This culminated in the web and a multitude of browsers and gave rise to the "dot.com" era.

Search engines and their lack of precision continues to be a problem which is exacerbated by paid ranking of results, publicity coupled with the exploitation and mining of user data. As it was reported in a number of tests in the early days of search engines, most of them were not precise. For instance in June 1995 a number of search engines, most of them developed at universities and listed in Table 1, were tested. The test shows poor results with high numbers of omissions.

Table 1. Search statistics for using search the term Bipin (AND) Desai

Search System	Number Hits	Number Duplicates	Number Mis-hits	Number missed
Aliweb	0	-	-	24
DA-CLOD	0	-	-	24
EINet	6	0	4	22
GNA Meta Lib.	0	-	-	24
Harvest	0	-	-	24
InfoSeek	7	0	0	17
Lycos	231	2	222	17
Nikos	0	-	-	24
RBSE	8	-	8	24
W3 Catalog	0	-	-	24
WebCrawler	7	3	0	20
WWWW	2	0	0	22
Yahoo	0	-	-	24

Many of the pioneering indexing systems, existing in mid 1995, were no longer accessible by 1997. In the meantime, a number of new systems, such as Altavista, OpenText, Hotbots etc. had emerged. Tests done in 1995 were repeated in fall 1997 on these new search engines Again the results, given in Table 2, were far from stellar!

Table 2. Search statistics for using the search term Bipin (AND) Desai
Total known URLs: 325 Sept-Oct 1997

Search System	Number Hits	Number Duplicates	Number Mis-hits	Number missed
AltaVista/Yahoo	97	9	23	264
Excite	114	10	29	247
Infoseek	8	2	1	319
Lycos	57	7	15	297
Hotbots	47	28	58	155
OpenText	19	-	7	318

The documents missed by the search engines could be due to the approximations used by engines which use a timeout feature to terminate the search or if parts of the search engine's database is off-line. However, the fact that these search engines could not locate all relevant documents indicates the inherent problem of the method used in indexing and determining the relevance of the web-pages contents. The bigger problem is the lack of selectivity and a measure of usefulness of the documents found by the search engines. We have collated the results by following the trail of "next" set of URLs and these could be viewed by pressing on the number of hits for each search engine in the online version(25) of Table 2. A glance at the abstract or summary presented by the search engine was not very informative in judging the relevance of the web page: following the pointers would result in a drain of the searcher's time if the corresponding web-page was not relevant in spite of being placed in the top of the search results.

The problem, raised by the author a few years ago(25), with then current automatically generated index databases is that their inadequate semantic information which still plagues us. Judging the relevance of a document is fairly difficult. The relevance based on term frequency, though attractive, has been abused on the web and given higher ranking to longer web pages containing the term. Term positioning was used in ranking but again this has been abused to artificially raise the relevance of a page. With search involving many terms, the proximity of terms may be used to increase the relevance and hence ranking of web pages. Anchor text reference and source authority were used by some search engines. In anchor text reference, the text that is used as anchor of a web link is used as a term for indexing: if the same text is used by many independent anchors to point to the same web page, the relevance of the web page is increased. This technique is protected from misuse by assigning higher weights to anchor text from more authoritative sources than from lesser ones; thus adding source

authority to anchor text. However, user has to use discretion in using the results with the current money-making scheme of paid ranking.

Table 3. 2001 Test Results of Major Search Engines Bipin (AND) Desai

Search System	Number Hits	Number Duplicates	Number Mis-hits	Number missed
AltaVista	99	24	67	230
Google	155	10	403	174
HotBot	62	21	124	262
Lycos	239	37	711	90

Table 3 summarizes the test results from a selected number of the third generation of search systems using the same keywords as in the previous tests. The test was carried out in early 2001 (66), (29). There were 329 web pages containing the search string on the Web well before the tests. The period was judged to be longer than the delay required by most engines before a new web page is indexed. As before, we show the number of hits, duplicates, miss-hits and missed pages for each search engine as well as the recall and precision

As we see, the search engine scene in 2001 had changed from the 1990s and many of the engines even from 1997 were not to be found . The early search engines had disappeared to be replaced by the late comers, who with financial resources have become dominant. However, as we see in the results of the 2001 tests, they were far from perfect and there were misses and mis-hits. We will see history repeating in the current frenzy of artificial intelligence, large language models, GPTs and bots.

The situation today is such that just one search engine is making a fortune and attempts to provide an alternative such as Cliqz(81) could not survive. New web browsers with built in search systems such as Brave (which incorporates a version of Cliqz) and Ecosia are recent entries. The advent of Dot.com and emergence of social video media streaming services has led to further mining of user data with the help of features such as auto-play and continuous scrolling and the use of influencers to enhance addictive features!

3 'Free' Email

Electronic mail abbreviated to email predates the web and was introduced in systems where users were connected by teletype terminals to a main-frame. With the introduction of the internet where many computers were connected (87(, (88). Each having a distinct address,, it became possible for users of different computers to communicate. The first email is said to be sent in 1971 on computers connected by ARPANET, the predecessor of the current internet, the back-bone of all Web based systems. It is during this time that the address syntax with the '@' symbol designating the address of the user's system was

introduced. The simple mail transport protocol(SMTP) was introduced in early 1980s. The current internet uses SMTP, Post Office Protocol(POP) and Internet Message Access Protocol (IMAP) protocols.

Email was limited to universities, government and business in the 1980s and 1990s. With the coming of the web and graphical user interfaces, web-mail came into use in the late 1990s. With the miniaturization of cell phones and introduction of the smartphone with a screen and email clients coupled with the availability of 'free' email, the general public was hooked on it.

The free email service was introduced at about the same time as the web in 1990s:(AOL mail 1993, Hotmail in 1995 re-branded Outlook in 1997, Yahoo Mail in 1997, Gmail being a late comer in 2004(86)). Free email is offered not due to the generosity of the company offering it, but the user is giving up her privacy to the corporation. All the data generated by the user are exploited by the free email service provider. the user's personal information, service use pattern including frequency of use is the price being paid(51). All this data using various mechanisms is used to generate targeted publicity and phishing emails as well as tracking. It is estimated that over 3 million deceiving/phishing emails are being sent everyday. Email accounts are open to being hacked and sold on the dark web.

The email messages also contain tracking mechanisms(invisible pixels) which are used to report back to the email servers when the message is opened. Since the email data is stored by the service provider it is subject to the providers' policy for security, usage, and the data is vulnerable to breaches affecting the providers services.

The email as well as the web design can't deal with privacy threats. The metadata of the transmission cannot be encrypted. This includes senders and recipients addresses, timestamps, subject lines, and message routing information that can be used to reveal patterns and relationships.

In addition the user's information is sold daily to thousands of companies multiple times per day. This information includes devices used, location, usage history, information collected from the data including demographic estimates. Such practice exposes the user's data to abuse.

4 Cell Phone

Since most organizations, when they develop a system for the web also make sure that it is compatible with the small cell screen and hence there is no need for most people to have a desktop or laptop or even a tablet and are hooked to the tiny screen. Telephone companies and others have added the money making angle of providing cell service with hundreds of gigabytes of data and unlimited text messages,

Communications instead of in person have become short text messages which are open to misinterpretation. There is the stress of instant notification of incoming text message and notification where a message has been read! Such

notification to the sender creates a pressure to reply promptly! The read receipts and the feature to see if a contact is typing is another addictive feature and more stress(79). One notices how people are hooked to their cells, it being always handy: walking, in public transport, driving using the cars audio system and voice activated dialing, in restaurants, museums, theatres and classrooms.

Young children are given a cell even before their teens by concerned parents. The existing cell phones, dominated by just two operating systems are controlled by two USian big-techs and have no parental control function. This exposes the children to bad actors especially with end-to-end encryption(E2EE). While E2EE, protects the user data from the big techs, bad actors can exploit the children. There is no move to build-in parental control in cell phones, tablets and other portable devices meant for the youngsters. Such control could allow parents to shepherd the young from danger of falling prey to recruiters .

5 OSN

Social networks in general consist of a set of individuals, organization etc who interact. Online social networks are run by organizations, mostly commercial and have become popular as a result of the web. The earliest system was for high school classmates and was established in the mid 1990s. (17). Since the mid -of the first decade of the 21st century a number of commercial Online Social Networks (OSN) have been established. Most of them being USian and have become very successful, rich and powerful. Their focus is a mix of personal, political, professional and other interests.

The USian Communications Decency Act(1996) protects OSNs (service providers on the internet) from all liabilities due to any posting made on their service or platform by any user's (third-party). This protects OSN giants from any misinformation and toxic prejudice and allows them a a great latitude for content monitoring. The popularity of the free platforms, used as the sole news source leads to dumbing down of users.

The OSN and free email service operators, established since the advent of the web, have perfected collecting data from calls, text messages, posting, user interaction etc. Users are led to believe that their privacy is protected by these platforms owned by the big techs! Hence, they entrust their chats, emails, photos and all type of data to them! When in fact the mammoths have no problem monetizing users' data including selling them regularly to one and all.

Furthermore, these systems allow bad actors to harm vulnerable: children, non-tech-savvy and easily gullible users. This is little difference with the mirage of the international rule-based order where the strongest exempt themselves when convenient: similarly, the international law applied with varied rigour, depending on the identity of the accused or the victim as noted in a Davos address by Carney(13). As in this address, which incorporates the "The Power of the Powerless" message (44), we cannot depend on the big-techs for our interaction since they are taking advantage of our dependence on their infrastructure while taking our jobs, creativity and control of our own data and institutes.

High speed internet and the use of the web to provide an interface has caused the re-birth of time sharing in the form of cloud computing. This has prompted the move from dedicated computing and software resources to the cloud and is coming at a cost of losing control of the data and the software to one of the cloud operators and software houses. The software is usually not compatible with the one that would have been developed in house and the users had been trained on, The new software, may not be tailored to the the one it replaces, could have glitches and requires re-training of users. There would be a large scale loss of service when a glitch hits one of these clouds. This is an example of the recent move from the organization controlling its own data centres, locally produced tailored software and personnel to manage the software and hardware to big techs who run cloud services with standardized software which fits no one. This is not only a loss of control but also loss of jobs.

Addictive features built in the OSNs result in the user's compulsive need to engage in the systems, the so called screen gazing(73), (90), (92), (93). The multi-billionaires who own these OSNs, fancy themselves as emperors of the web and use their systems to go beyond all regulatory limits, in defiance of national borders and democratic principles(60).

On top of the addictive algorithm the OSNs and other web based applications have added to their systems a role named "Influencer"(95) . Influencers, who depend on the OSNs for an income, are also addicts and continue to be so to attract 'followers". As reported in (21), (36) the influencer 's earning computation uses factors such as number of followers, engagement rate and other factors determined by yet other algorithms!

With the early use of tablets and cell phones children even though at home and physically safe, may be exposed to an entire virtual world with almost no regulation over the content they see, and not being monitored by their guardians. Their innocent social communication with friends may be mixed with bad actors pretending to be a friend; these bad actors may lead them into harmful material and fall victim to predatory influencers(78).

The number of child abuse web sites has doubled according to a recent report which noted that Internet Watch Foundation (IWF), found 15,031 commercial child sexual abuse sites in 2025, more than twice as much found in the previous year. Disgusting material is found on most OSNs and a number of underage victims of sextortion has almost doubled in one year(12). While some justices of the International Criminal Commissions, at a whim of one man, are labelled as terrorist, these bad guys are not unearthed by the OSNs operators.

OSNs have become the digital hunting grounds for traffickers of children providing them access to children and customers as detailed in (64). This is finally coming to light as illustrated in a number of legal cases. In a case in Los Angles, CA, a jury found Meta and YouTube liable for designing addictive products while failing to provide warnings of the potential dangers of their products that hooked a young user and led to her being harmed(54). In a separate case, Meta was found liable in child exploitation(65). It is obvious that these tech giants are

going to use their large fortunes to appeal these and further cases and influence corrupt officials to continue their systems to make money.

The priorities of OSNs is illustrated in a The Guardian news article from 2023 involving a child under care of a socail worker: in 2020 and 2021 she offered to train Instagram staff to help prevent child trafficking on its platforms. She says the training didn't go ahead as, after a long back and forth, on a video call Instagram executives said that they wouldn't pay her standard fee of $3,000, instead allegedly offering $300: the authors of this article claim that Meta did not deny this(64)! An article in BBC reports that Meta has removed publicity placed on its platform by law firms to reach potential clients for future lawsuits related to social media addiction: thus not allowing the law firms to make money! However, OSNs do not remove scammers from their platform, just increase the fees they are charged for putting in scamming contents(18). The resources could have been used to detect and remove contents such as child pornography(72).

OSNs ignored experts' advise about using features such as beauty filters which harmed adolescent girls' self-image. Another business practice used by some OSNs is not curbing criminals who place paid publicity for scams and banned goods. When detected, instead of banning them from the platform, the OSNs increase their charges for their publicity and perhaps make more money from them! The Canadian province Manitoba is proposing a law, the first in Canada, banning the use of social media and AI chat bots by youths. This is based on studies that show that OSNs are designed to get people addicted and cause anxiety and depression(42). This is similar to the ban in Australia in force since December 2025(82).

6 Artificial Intelligence(AI) Promises

One of the reasons that the tests of search engines presented above were not repeated after 2001 is the effort and time required to to recreate the values for the hits, mishits, duplicates etc. The tests in 1995, 1997 and 2001 were sufficient for the user to be warned to take result and its order without further verification and validation. Now the emerging application of AI is trained on millions of texts and produces output on prompt. To validate this output would be a super-human task! We have started using terms such "hallucination" to refer to results produced by the AI agents which are not possible.

In this paper we refrain from going into the details of generative pre-trained transformers(GPTs) a genre of LLM or how such models are trained on large quantity of natural language copyrighted data without permission(85). We will focus on the changes it is bringing to human life and the failures of enhancing it for the vast majority of humanity. The failures include incidence such as not telling authority of mass shooting suspect's interactions in one of these AI systems(46).

The emerging AI big-techs, new ones and add-ons to existing ones, are promising applications such as large language models(LLM) and variations based on the biological neural network. They use vast amounts of natural language

text to perform natural language understanding and hence are able to respond in natural language or perform tasks expressed in natural language. Generative AI refers to deep-learning models that can generate high-quality text, images, and other content based on the data that the the models were trained on. The input is in the form of text or speech. One concludes that AI is algorithms and data running on powerful computing systems using large amount of power and generating huge quantity of heat and pollution.

7 AI Issues

The fears of technology and automation have been with us for centuries! While replacing old jobs, new jobs are created. However these new jobs happen gradually and require training and different skills. Companies are integrating and replacing lots of customer service tasks by AI, either completely or with human supervision(43). Customer help systems using chat help features are already common though to the author's experience they have never resolved issues! Workers are monitored with AI for speed in doing the tasks: which echos the motion studies used in the early days of automation and creates pressure and mental anguish. Since the monitoring is not accurate, it creates more problems when disciplinary measures are taken based on such practices!

Many forms of rudimentary AI are already in current software systems including office software, cell phone applications: examples are: suggested text, auto-completion, recommendation engines, translation tools, and voice assistants. From the author's experience, some of these are error prone and annoying.

The CEOs of the new AI techs promise that generative AI is the panacea for the 21st century; it will create new cures for diseases, solve issues in climate change, make jobs easy and interesting, improve life and end loneliness, help businesses and government to become more responsive etc, This is hallucination not by AI but the billionaire tech lords of existing and emerging big tech. They will continue on their way to colonize our world, in addition to the internet with their tech monster(30). As noted in (57) this is hallucination not by generative AI but by these new tech czars and corrupt and unhinged politicians they support and help elect, In addition to hallucination where AI fabricates information that is either misleading or false, it plagiarizes as published for a book review in a leading newspaper(19). One wonders that everything that a GPT produces is plagiarization! Some instances of hallucination have been reported in court cases where lawyers, to cut corners, prepare briefs using GPTs.

We know that existing software, critical or otherwise, are buggy and the bugs are yet to be discovered. There being a race between the good guys and the bad ones: the good guys would try to patch them and the bad ones would exploit them. As an example, consider the current state of web search systems which has gone through decades of developments and use. Often a search produces ghost links due to the system not being updated – one can consider this as search engine hallucination!

Many software systems are released for general use with one or more so called zero-day bugs: these are security vulnerabilities, unknown to developers of the systems. These unknown issues have not been addressed by updates or patches. It is possible for hackers to exploit these vulnerabilities and hence zero-day bugs could lead to attacks and downtime. This includes loss of sensitive information and/or installation of malware. Even after updates to such buggy software are released, not all such installations may be patched promptly; this leaves some installations to the vulnerabilities(91).

AI systems have been developed to address and solve buggy software development problems (49). Such systems can find security vulnerabilities in software(76). However, if such an AI system can carry out scan and discover bugs unknown to the developers then one wonders about what all these years of software engineering has achieved! The other question is what is the likelihood that the AI system, used to find bugs, itself has glitches(80) ?

One concern with AI proliferation is the loss of jobs! It has already been noticed by one of the big-techs, which is spending billions pursuing LLM and AI, that one person with AI can do a job of a team -the size of team being undisclosed! As in manual work, existing workers, who are to be made redundant, are used to train the AI which is going to replace them. The reasons cited for tracking human interaction with computer is that the AI agent being created to replace workers, needs to be trained using current employees doing their jobs(43). As reported, some big-techs after spending billions on AI, are going to cut ten-percent of their workforce (45).

As we can see, our current system is for the most part corrupt and the super-powers, ignoring all existing international treaties including the founding principle of the United Nation, are attacking other countries, with hallucinating security concerns. The system is built to maximize the wealth and feed the greed of those at the top. As pointed out in (57) AI and its use will lead to further concentration of wealth while plundering those at the bottom. Due to the early adoption of some of these new AI tools, many corporations have suspended hiring new young employees and depend on AI tools to create and maintain their infra-structure including software. Some of the Gen-Z, not finding an entry level job, are becoming entrepreneurs: however, this path is not open to all. Other organizations are flattening their organization structures to eliminate middle management and combine some managers functions with additional duties or larger numbers of people reporting to them. AI would likely eliminate lot of jobs including software engineering. One day, the structure would have just the CEO and everyone reporting directly to that person! This will cause a further concentration of wealth and widening of the chasm between the ultra-rich and the rest.

Technology has been used over the ages by humans to take advantage by some over others. The native people of the Americas used flint arrows. The European who came to the shores of the Americas had already developed lethal arms. The technology of the gunpowder was used in guns and cannons: these were used to overpower and displace and eliminate the indigenous population: this amount

to a genocide in the Americas. As outlined in (5), the intensive surprise blanket bombing was used in the beginning of WWII and again during the six-day war in the Middle-east to neutralize the opponents air power.

The failures are many-fold some of which include the use of technology to perfect arbitrary missile and drone strikes to take out even minor leaders and the harm being done to the innocent population including children. The collateral failure is the increasing amassing of fortune by a small number of CEOs etc. while increasing the number of homeless in developed countries. The tools being used are requiring an increasing size of data center power, all of which require large expenditures while cutting back on aid to the most vulnerable(70).

The help in collection and integration of all kinds of communications, satellite imaging and old fashion espionage and integration of AI by the companies under contract and working with the defence forces of some countries has resulted in development systems such as Lavender, Maven and Gospel(2). It is difficult to determine due to layers of secrecy the actual line of command in incidence such as bombing and killing of girls at a school in Iran to determine if human decision was involved or the target was among the thousands determined by machines from databases not updated. The use of systems such as Maven in the school bombing is the result of the decision to create systems, just mentioned, to get technological superiority in battle called the third offset(98), the first being a nuclear weapon deterrent. The second offset is the use of intelligence, surveillance, and reconnaissance platforms; the development and deployment of precision-guided weapons, stealth technology, space-based military communications and navigation systems.

Present day military operations produce large amounts of data from phone calls, text messages, internet tracking, satellite and drone images. Using AI to process this data reduces manpower and increases speed. An unit of USian military reduced the number of people from 2,000 to just 20 to do this tasks(20), (58), (69). This automated set of targets is not usually analyzed deeply, since there is not sufficient time or resources for humans to determine how the AI system had generated the targets. As in assembly lines, the performance is based on the number of targets generated. With systems such as Lavender, Maven and Gospel the human becomes a cog in the mass assassination system of the strong. A bomb of two-thousand pounds of explosives is used to target a home to kill one person: everybody around, including children and women, is accepted as a collateral damage. This has been labelled automation bias of mass assassination(9). From the point of view of humanitarian law this is a concern!

As with most things, there are two sides to the AI debate(33). As one can see, while the big-techs are promoting AI in weaponry and surveillance and OSNs are allowing exploiting children and AI is being used to replace workers, there are positive sides to its use. After all it is for humans to decide which side should be pursued. One of the possibilities is for OSNs to improve the monitoring of its platforms using AI!

AI is being used for military purposes and when one leading company refuses to comply another steps in. One of the conditions of one of the new AI firms was

not to use their tools for surveillance of locals or use in autonomous weapons without a human operator. USian military had used AI tools during an operation in Venezuela in Jan 2026. AI embedded systems are continued to be used in the wiping out of the population of Gaza and the assault on Iran. These systems could use intelligence information and choose targets. Thus AI tools are used in what most observers call illegal wars where civilians and children are killed in large numbers as well as journalists(16).

8 Possible Solutions

Towards the end of the 20th century, after the advent of the world wide web in 1994, there was the so called dot.com bubble. Many new companies were established to exploit the web and a creeping reactionary fervour mixed with the Silicon Valley's worship of male power(6) prevailed. Many of them were not able to succeed due to the lack of venture capital which was concentrated in the "Silicon" valley. Hence the ones that survived, thanks to the venture capitalists, have become the hi-tech mammoths which have colonized the internet. We are re-living this bubble with AI(10). Some of the existing hi-techs are partners or have their own versions of AI GPTs. They are accumulating fortunes creating new billionaires!

We as humans have not found a system to take care of the unfortunate ones among us. Greed, selfishness, corruption and lack of charity keeps this inequality wide. Religion has become a formality for show and their principles are lip-service.

The internet and its web interface for sharing has failed while succeeding in exploitation! Any attempt to regulate it and promote local and national well being has been met with resistance. The contents and services on it have enriched a small number of USian corporations. Just as it happened during the time of colonization of a large part of the world by the Europeans, the might of the USian government is behind these limited numbers of its tech mammoths. Examples of using the USian blackmails are cited in the press including the ones from Australia(52), Canada(60), France(3)- one of the arguments used is free speech essentially for the USian big-techs and their AI agents.

Some of the newer CEOs and associates of the newer to-be-big-techs are said to be pledging to donate 80% of their wealth. However, the question is when and how. .fortunes. Recently there is a court case involving two wealthy CEOs of hi-tech about converting parts of an AI company from not-from profit to profit. One wonders at the motives of the parties in these legal battles. Another AI company is said to offer 80% of its wealth. However these companies has pirated copyrighted material and human knowledge. One possible way of righting this piracy is to offer at least a third of the annual earning to authors of the pirated material in proportion to the amount of text used. Another third should go to children's charities in the proportion of the companies earnings from each country.

One notices that sites such as Anna's Archive are suggesting to their users to send a donation to the authors of the texts that they download. This is similar to the author's earlier attempt to create CINDI and his current introduction of CopyForward[1] as a moral obligation to reward the creators and the the next generation for the works they have created. However, as the author has discovered, in spite of thousands of downloads of his texts, there is not a single word from any of these users if they had made a contribution to their favourite children;s charity!

We feel that the basis of the internet and the web on the premise that there are no bad actors is wrong. The bad actors are the greed and/or base people either in charge of mammoth big-techs or of a criminal bent of mind. In all such cases, they are more concerned with their gains than the harm they cause. Perhaps it is a bubble and may or may not bust!

However, we all know that the internet in general and the web as it exits is being exploited by a number of USian tech mammoths as well as spammers and perverts of all types (see: Cory Doctorow, 'The Internet Has Become Too American to Trust", https://thewalrus.ca/the-internet-has-become-too-american-to-trust/)'. It is time that what has become an essential service instead of being controlled by a few USian mammoths becomes national under control of elected representatives. Since it is also a form of 'tax', the motto 'no taxation without representation' must now apply.

As pointed out by the author and others, the email system and other network services such as networking etc. must be moved under control of the national government. A protocol to transmit message among such national systems must be established. One suggestion is to provide email service, free for all low-income users, by the post office which is currently struggling to survive. In Canada it was recently decided to stop door to door mail delivery(59) and transition to community mailboxes, eventually ending home delivery. Why not bring the written communication need of the people under a modernized post office with nationally supported servers. Not by big-techs and their CEOs who make billions of dollars on our data. The so called free email system controlled by the big-techs is one of the main reasons for the drop in the volume of postal mail. In addition most businesses including banks have discontinued sending statements and bills by the postal service transferring the onus of checking the status of their accounts on the customers who need to be connected. Since the customer has to pay for the internet connection, the cost is essentially transferred to the unwary consumer. This is an opportunity for the country to transform the postal system into an email provider as well as set-up a national social network. Canada can show the other national postal systems how to safeguard the citizens data from exploitation by big-techs. Once the national email service is established, there is no need to continue with the big-tech's OSN services. The national email service should be followed by national OSN services.

Privacy and security of the user data must be protected while ensuring durability. AI must be used to flag and disable unsuitable contents such as recruiting

[1] https://users.encs.concordia.ca/ bcdesai/CopyForward.pdf

of children and spam. There must be legal penalties including disabling accounts, fines and legal procedures to enforce the safeguards.

In the meantime, immediate steps should be taken by each jurisdiction and any blackmail from the big-tech or the USian govt. should be responded to with the cutting off of the internet traffic from such big-techs after adequate warning. Policymakers can take steps to strengthen safety standards and limit access in ways that make social media safer for children of all ages, better protect everyone's privacy, support digital and media literacy, and fund additional research.

The national systems should be designed to transparently assess its impact on children, share data with independent researchers to increase the collective understanding of the impacts, make design and development decisions that prioritize safety and health – including protecting children's privacy and better adhering to age minimums and improve systems to provide effective and timely responses to user concerns(48).

Devices for children must be developed to allow parental control until reliable and trustworthy national infrastructures are in place to replace the colonizing big-techs. There is room for an addition to the couple of systems that are now being used for cell phones. Perhaps, one based on the old Blackberry system with features for Linux may be in order.

As for a GPT that has used millions of copyrighted texts, it may be better to create special purpose GPTs for local use, This will allow distributed local data centres, reduce traffic and create less pollution and more jobs. Since these tasks are expensive, it is hoped that governments of middle size nations would create a structure to implement the open source replacement systems on open source software, thus preventing emergence of other big-techs.

The other long term approach would be to create a NEW internet with better security and end to end encryption; a new email protocol where only the destination address is included in the header and the recipient id is encrypted only to be de-crypted at the destination!

References

1. Abraham, Y.:"'A mass assassination factory': Inside Israel's calculated bombing of Gaza",Nov. 2023, https://www.972mag.com/mass-assassination-factory-israel-calculated-bombing-gaza/
2. Abraham, Y.: "'Lavender': The AI machine directing Israel's bombing spree in Gaza", Apr. 2024, https://www.972mag.com/lavender-ai-israeli-army-gaza
3. Agence France-Presse ,"Elon Musk snubs Paris legal summons over alleged child abuse images on X", Apr, 2026, The Guardain, https://www.theguardian.com/technology/2026/apr/20/french-prosecutors-summon-elon-musk-over-alleged-child-abuse-images-on-x
4. Baker, K. T.: "AI got the blame for the Iran school bombing. The truth is far more worrying", Mar. 2026, The Guardain, https://www.theguardian.com/news/2026/mar/26/ai-got-the-blame-for-the-iran-school-bombing-the-truth-is-far-more-worrying

5. Batool, H. S. A.: "General Giulio Douhet: Theory Of Air Power – Analysis"., Nov, 2023, https://www.eurasiareview.com/25112023-general-giulio-douhet-theory-of-air-power-analysis/
6. Becca Lewis, B.:"'Headed for technofascism': the rightwing roots of Silicon Valley", Jan. 2025, https://www.theguardian.com/technology/ng-interactive/2025/jan/29/silicon-valley-rightwing-technofascism
7. Boulanin, V.; Bo, M: "Three lessons on the regulation of autonomous weapons systems to ensure accountability for violations of IHL", Mar. 2023, https://blogs.icrc.org/law-and-policy/2023/03/02/three-lessons-autonomous-weapons-systems-ihl/
8. Brigadier General Y.S, "The Human-Machine Team: How to Create Synergy Between Human & Artificial Intelligence That Will Revolutionize Our World", 2021, eBookPro Publishing, ISBN 9798749152210
9. Bruun. L.; Bo, M.: "Bias in Military Artificial Intelligence and Compliance with International Humanitarian Law", Aug. 2025 . SIPRI. https://doi.org/10.55163/NLWV5347
10. Cadwalladr, C.: "The Great AI Bubble", Nov. 2025, https://broligarchy.substack.com/p/the-great-ai-bubble
11. Cadwalladr, C.: "It's a journocid", Mar. 2026, https://broligarchy.substack.com/p/its-a-journocide
12. Campbell, S. "Criminal gangs profiting as child sexual abuse websites double, experts say", Apr. 2026, The Guardian, https://www.theguardian.com/technology/2026/apr/23/criminal-gangs-child-sexual-abuse-websites-double-report
13. Carney, M, "The Rules No Longer Protect You", Jan. 2026, World Economic Forum, https://thewalrus.ca/carney-speech-world-economic-forum/
14. Cavendish, C.: "AI is dressing up greed as progress on creative rights", Financial Times, Mar. 2026, https://www.ft.com/content/48532284-9244-4ee6-be46-f3bde22b7232
15. "How AI could help conservation work", Apr. 2026 CBC, https://www.cbc.ca/news/climate/what-on-earth-ai-conservation-9.7165676
16. Team Citizens, "Is this the first AI war?" Mar. 2026, https://dispatch.thecitizens.com/is-this-the-first-ai-war/
17. Classmate, https://www.classmates.com/
18. Cress, L.: "Meta pulls Facebook ads recruiting for social media addiction lawsuits" Apr. 2026, BBC, https://www.bbc.com/news/articles/czjw0zgz9zyo
19. Cyca, M.: "The New York Times Got Caught Using AI Hallucinations in Its Reporting", May 2026,The Walrus, https://thewalrus.ca/the-new-york-times-got-caught-using-ai-hallucinations-in-its-reporting/
20. Davies, H.; McKernan, B.; Sabbagh, D.: "The Gospel': how Israel uses AI to select bombing targets in Gaza", Dec. 2023, The Guardian, https://www.theguardian.com/world/2023/dec/01/the-gospel-how-israel-uses-ai-to-select-bombing-targets
21. Derbyshire, L., "How Much Do Influencers Really Make?", Apr. 2026, https://www.influize.com/blog/how-much-do-influencers-make
22. Desai, B. C.; "Report of the Metadata Workshop", Mar. 1995, https://spectrum.library.concordia.ca/id/eprint/990940/1/metadata-workshop-report-complete.pdf
23. Desai, B. C.; Pinkerton, B.: "Workshop A: Web wide Indexing Semantic Header or Cover Page", Apr 1995,

https://spectrum.library.concordia.ca/id/eprint/985374/1/WWW-III-WrkShpA.pdf
24. Desai, B.C; Swiercz, S.: "WebJournal: Visualization of a Web Journey", LNCS V1082, https://doi.org/10.1007/BFb0024604
25. Desai, B. C.: "Test: Internet Indexing Systems vs List of Known URLs: Revisited", Oct. 1997, https://spectrum.library.concordia.ca/id/eprint/983876/1/test-of-index-systems-revisited.html
26. Desai, B. C. , Shinghal, R., Shyan, N. and Zhou, Y.: "CINDI: A System for Cataloguing, Searching, and Annotating Electronic Documents in Digital Libraries". ISMIS'99, Warsaw, Poland, June 8-11, 1999: proceedings. Springer, Berlin; New York.
27. Desai, B. C.: "Search and Discovery on the Web", Sept. 2001, Spectrum, https://spectrum.library.concordia.ca/id/eprint/983874/1/rerevisit-2001.pdf
28. Desai, B. C.: "Colonization of the Internet", IDEAS2-21, Jul. 2021, https://doi.org/10.1145/3472163.3472179
29. Desai, Bipin C. "Meta-stasis of the Internet". Aug. 22-24, 2022, Budapest, Hungary, https://spectrum.library.concordia.ca/id/eprint/990817/
30. Desai, Bipin C. "Test: Internet Indexing Systems vs List of Known URLs: 2001", Aug. 2002, https://spectrum.library.concordia.ca/id/eprint/990943/1/Results-of-test-on-search-engines-2001.pdf
31. Down, A.; Booth, R.: "Palantir manifesto described as 'ramblings of a super villain' amid UK contract fears", Apr. 2026, The Guardian, https://www.theguardian.com/technology/2026/apr/21/palantir-manifesto-uk-contract-fears-mps
32. Dreyfus Emily: "Our Kids Are Living in a Different Digital World", New York Times, Jan. 2024, https://www.nytimes.com/2024/01/12/opinion/children-nicotine-zyn-social-media.html
33. Duhigg, C. "Silicon Valley, the New Lobbying Monster", Oct, 2024, The New Yorker, https://www.newyorker.com/magazine/2024/10/14/silicon-valley-the-new-lobbying-monster
34. Dublino, J.: "Social Media Stars: How Much Do They Really Make?", Jan 2026, https://www.business.com/articles/social-media-stars-how-much-do-they-really-make/
35. Fenlon, B.: "How CBC News will manage the challenge of AI", Jun. 2023, https://www.cbc.ca/news/editorsblog/cbc-twitter-news-1.6873270?cmp=rss
36. Freedland, J.: "The future of AI is chilling – humans have to act together to overcome this threat to civilisation" The Guardain, May 2023, https://www.theguardian.com/commentisfree/2023/may/26/future-ai-chilling-humans-threat-civilisation
37. Frenkel, Sheera; Mac, Ryan: "Twitter Sues Nonprofit That Tracks Hate Speech", https://www.nytimes.com/2023/07/31/technology/twitter-x-center-for-countering-digital-hate.html#gateway-content
38. Froese, I.: "Manitoba to ban social media, AI chatbots for youth, premier says", Apr 25, CBC, https://www.cbc.ca/news/canada/manitoba/manitoba-social-media-age-restrictions-9.7177470
39. Goldberg, E.: "'Training My Replacement': Inside a Call Center Worker's Battle With A.I.", NY Time, July 2023, https://www.nytimes.com/2023/07/19/business/call-center-workers-battle-with-ai.html

40. Havel, V.: "The Power of the Powerless", 2015, Routledge, Ed. John Keane, ISBN 13: 9780873327619
41. Hays, K.: "Meta to track workers' clicks and keystrokes to train AI", Apr. 2026, BBC, https://www.bbc.com/news/articles/cvglyklz49jo
42. Hays, K.: "OpenAI boss 'deeply sorry' for not telling police of mass shooting suspect's account", Apr. 2016, BBC, https://www.bbc.com/news/articles/cq6je7e80r7o
43. Hays, K.: "Meta to cut one in 10 jobs after spending billions on AI", Apr. 2026, BBC, https://www.bbc.com/news/articles/crm1y89vek8o
44. Health and Human Service-US, Surgeon General Issues New Advisory About Effects Social Media Use Has on Youth Mental Health, May 2023, https://www.hhs.gov/about/news/2023/05/23/surgeon-general-issues-new-advisory-about-effects-social-media-use-has-youth-mental-health.html
45. Islam, F.; McMahon. L.: " Finance ministers and top bankers raise serious concerns about Mythos AI model", Apr. 2026, BBC, https://www.bbc.com/news/articles/c2ev24yx4rmo
46. Jackson, L: "A Driver's License for the Internet", NY Times, July 2023, https://www.nytimes.com/2023/07/03/briefing/age-verification.html
47. Jackson, O.; Baumgarten, C.; Sumarsono, A. Ranardo.: Nov. 2025, "https://www.getmailbird.com/truth-about-free-email-services-data-privacy/"
48. Jamali, L.; Turnbull, T.: "Australia's social media ban for children has left big tech scrambling", Dec. 2025, https://www.bbc.com/news/articles/ce86381p70eo
49. Jamali, L.; Turnbull, T.: "Australia's social media ban for children has left big tech scrambling", Dec. 2025, https://www.bbc.com/news/articles/ce86381p70eo
50. Jones. C.; Kinsella, H.M.: "Iran war shows how AI speeds up military 'kill chains'", March 2026, The Conversation, https://theconversation.com/iran-war-shows-how-ai-speeds-up-military-kill-chains-278492
51. Kerr D.: "Meta and YouTube designed addictive products that harmed young people, jury finds", The Guardain, Mar 2026, https://www.theguardian.com/media/2026/mar/25/jury-verdict-us-first-social-media-addiction-trial-meta-youtube
52. Kerr D.; Robins-Early, N: "Musk and Altman's bitter feud over OpenAI to be laid bare in court", Apr 2026, https://www.theguardian.com/technology/2026/apr/26/musk-altman-openai-court
53. Khodeir, R.: "AI in War: What the Iran War Reveals About the Pentagon's Algorithms", Mar. 2026, https://www.habtoorresearch.com/programmes/ai-war-pentagon-algorithms/
54. Klein, N.: "AI machines aren't 'hallucinating'. But their makers are", The Guardain, May 2023, https://www.theguardian.com/commentisfree/2023/may/08/ai-machines-hallucinating-naomi-klein
55. Kumar, A.: "Pentagon Used Anthropic's Claude AI and Palantir Maven to Identify 1,000 Targets in Iran Strikes", Mar. 2026, https://www.thedefensenews.com/news-details/Pentagon-Used-Anthropics-Claude-AI-and-Palantir-Maven-to-Identify-1000-Targets-in-Iran-Strikes/
56. Logan N.: "Canada Post is planning to end home delivery. Here's how community mailboxes will work", Apr. 2026, CBC News, https://www.cbc.ca/news/business/canada-post-community-mailboxes-questions-9.7148787

57. MacArthur,J. R., "Un coup d'État numérique'. Le Devoir, Sept 2023, https://www.ledevoir.com/opinion/chroniques/797387/chronique-un-coup-d-etat-numerique
58. Malone, M; "Canada's access to information system is not ready for AI"The Globe and Mail, https://www.theglobeandmail.com/opinion/article-canadas-access-to-information-system-is-not-ready-for-ai
59. Martineau, K.: "What is generative AI?", Apr. 2023, https://research.ibm.com/blog/what-is-generative-AI
60. Narayanan. M.: "Inside Palantir: Profits, Power & The Kill Machine", Jul. 2025, The Citizens, https://the-citizens.com/2025/07/inside-palantir-profits-power-the-kill-machine/1
61. McQue, K.; McNamara, M.: "How Facebook and Instagram became marketplaces for child sex trafficking:, The Guardain, Apr, 2023, https://www.theguardian.com/news/2023/apr/27/how-facebook-and-instagram-became-marketplaces-for-child-sex-trafficking
62. McQue, K.: "Meta ordered to pay $375m after being found liable in child exploitation case", The Guardain, Mar 2026, https://www.theguardian.com/technology/2026/mar/24/meta-new-mexico-jury
63. Mechouet, M. A.: "Web Based CINDI System", Masters Thesis, Concordia University, April 2001
64. Milmo, D.: "'Godfather of AI' shortens odds of the technology wiping out humanity over next 30 years", The Guardian, Dec. 2024, https://www.theguardian.com/technology/2024/dec/27/godfather-of-ai-raises-odds-of-the-technology-wiping-out-humanity-over-next-30-years
65. Modi, D. (curator), "200 Targets In 10 Days: What Is Gospel, Israel's AI For Wars?", Mar, 2026, https://www.news18.com/world/200-targets-in-10-days-what-is-gospel-israels-ai-for-wars-ws-kl-9944640.html
66. Murugesan S.: "Did the Pentagon use AI to target the Iranian school strike?", Mar., 2026, https://news.meaww.com/fact-check-did-the-pentagon-use-ai-to-target-the-iranian-school-strike
67. Mustafa, N.: "The 'dangerous' promise of a techno-utopian future" Jan. 2025, https://www.cbc.ca/radio/ideas/tech-billionaires-ai-utopia-1.7440698
68. Niranjan, A.; Schmidt, N.; Joyner, E.: "US tech firms successfully lobbied EU to keep datacentre emissions secret", Apr. 2026, The Guardian, https://www.theguardian.com/technology/2026/apr/17/microsoft-us-tech-firms-lobbied-eu-secrecy-rules-datacentre-emissions
69. Noah. T.: "How the Tech World Turned Evil", Apr., 2026, The New Republic, https://newrepublic.com/article/208876/tech-world-evil-musk-bezos-thiel
70. Nix, N: "Teenager sues Meta over 'addictive' Instagram features", Washington Post. https://www.washingtonpost.com/technology/2024/08/05/meta-lawsuit-teen-mental-health/
71. Pinto. J. S.: "The Guernica of AI", Feb 2025, Ziggurat, https://www.zig.art/p/the-guernica-of-ai-c4b
72. Prévost, H.: "C-18, médias, Meta et Google : un bras de fer en quatre questions", Jun. 2023, https://ici.radio-canada.ca/nouvelle/1991682/medias-canada-facebook-google-c18-explicatif
73. Roose, K.: "Anthropic Claims Its New A.I. Model, Mythos, Is a Cybersecurity 'Reckoning'", Apr. 2026, New York Times, https://www.nytimes.com/2026/04/07/technology/anthropic-claims-its-new-ai-model-mythos-is-a-cybersecurity-reckoning.html

74. Spectrum "Spectrum Research Repository, Concordia Univesity", Montreal", https://spectrum.library.concordia.ca/
75. Sridhar. D.: "Social media could be as harmful to children as smoking or gambling – why is this allowed?", Jul 2023, The Guardian, https://www.theguardian.com/commentisfree/2023/jul/04/smoking-gambling-children-social-media-apps-snapchat-health-regulation
76. Sridhar. D.: "SEven without social media, phones have a subtle, damaging effect on our mental health", Apr 2026, The Guardian, https://www.theguardian.com/commentisfree/2026/apr/25/phones-social-media-damaging-mental-health
77. Townsend, K.; "Critical Vulnerability in Claude Code Emerges Days After Source Leak", Apr. 2026, https://www.securityweek.com/critical-vulnerability-in-claude-code-emerges-days-after-source-leak/
78. Tunney C. "Ottawa 'very seriously' considering age restrictions for social media, AI chatbots", Apr. CBC News, https://www.cbc.ca/news/politics/social-media-ai-minimum-age-9.7164902
79. Wong, J.: "Australia's social media ban for users under 16 starts now" Dec. 2025, CBC News, https://www.cbc.ca/news/world/au-social-ban-begins-9.7007357
80. Wong, J.: "Australia is settling into age-restricted social media. Canada is mulling whether to join in" Mar. 2026, https://www.cbc.ca/news/world/australia-u16social-ban-checkin-9.7122420
81. Wiener, A: "The Age of Chat", The New Yorker, June 2023, https://www.newyorker.com/culture/the-weekend-essay/the-age-of-chat
82. Wikipedia, "Generative pre-trained transformer", https://en.wikipedia.org/wiki/Generative_pre-trained_transformer
83. Wikipedia, "Comparison of webmail providers", https://en.wikipedia.org/wiki/Comparison_of_webmail_providers"
84. Wikipedia, "History of email", https://en.wikipedia.org/wiki/History_of_email#Terminology_and_usage
85. Wikipedia, "Email", https://en.wikipedia.org/wiki/Email
86. Wikipedia, "AI-assisted targeting in the Gaza Strip" , https://en.wikipedia.org/wiki/AI-assisted_targeting_in_the_Gaza_Strip
87. Wikipedia, "Problematic social media", https://en.wikipedia.org/wiki/Problematic_smartphone_use
88. Wikipedia, Zero-day vulnerability, Apr. 2026, https://en.wikipedia.org/wiki/Zero-day_vulnerability
89. Wikipedia, "AI-assisted targeting in the Gaza Strip", https://en.wikipedia.org/wiki/AI-assisted_targeting_in_the_Gaza_Strip
90. Wikipedia, "Internet addiction disorder", https://en.wikipedia.org/wiki/Internet_addiction_disorder
91. Wikipedia, "History of the web browser", https://en.wikipedia.org/wiki/History_of_the_web_browser
92. Wikipedia, "Influencer", https://en.wikipedia.org/wiki/Influencer
93. Wikipedia, "File Sharing", https://en.wikipedia.org/wiki/File_sharing
94. Wikipedia, "Time Line of File Sharing", https://en.wikipedia.org/wiki/Timeline_of_file_sharing
95. Wikipedia, "Offset strategy", https://en.wikipedia.org/wiki/Offset_strategy
96. WWW, First International Conference on the World Wide Web, May 1994, CERN, Geneva, https://archives.iw3c2.org/www1/
97. Wikipedia, "Hallucination (artificial intelligence)", https://en.wikipedia.org/wiki/Hallucination_(artificial_intelligence)

Phage Virion Protein Identification via Multi-View Feature Extraction and Hybrid Deep Learning

Alfredo Cuzzocrea[1,2], Tasmin Karim[3], Md. Shazzad Hossain Shaon[3], Md. Fahim Sultan[3], Ismail Benlaredj[1], and Mst Shapna Akter[3]

[1] iDEA Lab, University of Calabria, Rende, Italy
{alfredo.cuzzocrea, ismail.benlaredj}@unical.it
[2] Department of Computer Science LIPADE
University of Paris City, Paris, France
[3] Department of Computer Science and Engineering
Oakland University, Rochester, MI, USA
{tasminkarim, shaon, mdfahimsultan, akter}@oakland.edu

Abstract. In this paper, we present a *hybrid Deep Learning structure with multi-view feature extraction* to improve *Phage Virion Proteins* (PVPs) prediction accuracy with four different feature extraction strategies: *N-Gram Composition*, *Amino-Acid Composition at the N- and C-Terminal*, *Sequence-Order-Coupling Numbers* (SOC), and *Transformer-Based Embeddings*. To construct effective prediction models, an extensive approach was used, combining conventional *Machine Learning* techniques with advanced *Deep Learning* models. We developed a hybrid DL-based classifier, aptly named as *DLPVP*. The results showed that the proposed DLPVP hybrid technique performed proficiently in detecting PVPs in both single- and multifunctional feature sets, thus achieving an accuracy of 94.28%.

Keywords: Multi-View Feature Fusion · Transformer-Based Embeddings · N-Gram Composition · Phage Virion Protein Prediction · Hybrid Neural Networks

1 Introduction

Bacteriophages, commonly known as phages, are pathogens that infect only microorganisms. They represent the most abundant and widely distributed biological entities on Earth, with an estimated population exceeding 10^{31} particles [6]. Phages regulate the evolution of microbiological ecosystems by triggering *bacterial lysis* and facilitating the *horizontal transmission* of genomic information [12]. *Phage Virion Proteins* (PVPs) [11] contain capsid proteins, tail peptides, and phage-associated enzymes, which play a crucial role in regulating relationships among bacteriophages and their particular bacterial organisms. Considering their critical role in host recognition and infection, altering these proteins could provide a viable avenue for the creation of new antimicrobial drugs [16].

Despite the growing reliance on computational models, accurate identification of PVPs remains a nontrivial task due to the heterogeneous nature of *protein sequences* and the complexity of functional domains within virion proteins. Many conventional models, while useful, struggle to generalize across diverse phage species or to capture *long-range dependencies* and *local contextual signals* within *Amino-Acid Sequences* [24]. Furthermore, most current methods rely heavily on either handcrafted features or *black-box learning* systems, often without effectively integrating multiple biologically relevant views of the protein sequences [5].

In this context, *Multi-View Feature Learning* (e.g., [23]) emerges as a promising strategy. By incorporating diverse feature representations, such as physicochemical properties, sequence motifs, and domain-specific positional signals, models can learn richer and more discriminative patterns [19]. For instance, *terminal region characteristics* and *global sequence order* may offer complementary insights that are overlooked when features are treated in isolation [20]. Combining such heterogeneous representations within a unified *Deep Learning* (DL) framework can lead to significant improvements in classification performance and generalizability [28].

To meet these demands, recent efforts have shifted towards *hybrid models* that blend traditional *Machine Learning* (ML) approaches with DL architectures. These systems can benefit from both the interpretability and efficiency of classical techniques, and the high-capacity representation learning of *Deep Neural Networks* [22]. The hybrid models proposed in this area build upon this paradigm by integrating multiple *Feature Extraction* strategies and leveraging DL to process them jointly. The final goal is to provide a scalable and accurate solution for PVP identification that addresses the shortcomings of previous single-view or shallow approaches [26].

Given the increasing importance of phage therapy and the growing interest in bacteriophage applications in medicine and Biotechnology, reliable computational tools for PVP identification are essential [25]. Advances in this area not only facilitate *phage genome annotation* but also accelerate the discovery of *novel therapeutic proteins* and help combat antibiotic resistance through phage-based alternatives [18].

Existing research strategies for differentiating PVPs from non-PVPs (e.g., [4]) depend on different robust scientific methodologies and equipment. These consist of *Mass Spectrometry*, *Sodium Dodecyl Sulfate Polyacrylamide Gel Electrophoresis* (SDS-PAGE)*-based Proteomic Methods*, and *Protein Analysis Arrays* – everything that supports the identification and analysis of PVPs [1]. Although these approaches are generally considered the golden benchmark for PVP recognition, their large-scale implementation is difficult due to their labor-intensive nature and cost. As a result, researchers have taken significant strides in developing computer frameworks that could determine PVPs directly from sequence data, presenting a more practical and cost-effective solution. To overcome the challenge introduced by high sequence diversity, *Artificial Intelligence* methods are commonly used for the classification of PVPs and non-PVPs (e.g., [15]).

In this paper, we propose DLPVP, *a novel hybrid DL model designed to accurately identify PVPs by leveraging a multi-view feature extraction strategy*. Our approach integrates four distinct types of sequence-based features: *N-Gram Composition*, *Amino-Acid Composition at the N- and C-Terminal*, *Sequence-Order-Coupling Numbers* (SOC), and *Transformer-Based Embeddings*. By combining both traditional and DL classifiers, DLPVP captures complementary information from diverse feature perspectives. This multi-level integration enables the model to effectively learn complex patterns associated with PVPs while mitigating limitations faced by earlier single-view or shallow models. We evaluate DLPVP on multiple benchmark datasets and demonstrate its superior predictive performance compared to existing state-of-the-art methods.

2 Novel Contributions

From the analysis of current state-of-the-art, there is significant room for improvement in existing methods for recognizing PVPs by adopting new views, notably through the implementation of DL models. Many researchers evaluated their models using limited feature selection strategies, while others worked with extremely small datasets. In this work, we introduce DLPVP, a hybrid model that correctly recognizes PVPs. Our main contributions in this study are as follows:

(*i*) This study rigorously explores four feature extraction approaches for representing *peptide sequence properties*, providing useful insights into the most informative features for identifying phage virion peptides. By evaluating several feature representations, the study improves our knowledge of sequence patterns contributing to accurate PVP identification.

(*ii*) An effective hybrid DL framework, DLPVP, has been proposed that combines *Convolutional Neural Networks* (CNN) (e.g., [17]) and *Bidirectional Long Short-Term Memory* (BiLSTM) (e.g., [29]) to improve the efficiency of PVP detection. Furthermore, the research conducts a thorough ablation analysis to acquire a better understanding of the model's internal functions, evaluating the contribution of various layers and components to overall performance.

(*iii*) Overall, this study shows an elaborate pipeline that combines several feature extraction strategies, adaptation testing on an alternate dataset, and a potent hybrid DL classification strategy to detect PVPs.

The findings of this work have significant implications for *Computational Biology and Biotechnology*. The provided DLPVP model not only outperforms current PVP identification approaches in terms of accuracy, but it also provides a scalable and efficient solution for large-scale analysis. With an accuracy of 94.28%, our model outperforms earlier research with a harmonious specificity of 94.41% and sensitivity of 89.54%. The use of DL in this discipline presents the path for advances in *protein classification*, which will help with *drug discovery*, *antimicrobial resistance* research, and the development of *phage treatment*.

While DLPVP demonstrates high accuracy in PVP prediction, several open issues arise, which can stimulate other progress work in the area. One direction

is the integration of additional biological context, such as *host-phage interaction data* or *structural protein features*, to further refine predictions. Expanding the dataset with more diverse and experimentally validated PVP sequences could improve generalizability across phage families. Additionally, incorporating *eXplainable AI* (XAI) techniques may enhance the interpretability of model decisions, aiding biological insights. *Real-time PVP prediction tools* based on DLPVP could be developed for use in clinical or environmental microbiology pipelines. Finally, adapting the model to *multitask learning frameworks* may enable simultaneous prediction of multiple phage protein functions, broadening the application scope of the proposed methodology.

3 Deep Learning for PVP Detection

In this Section, we provide the details of the DLPVP DL methodology by inspecting all the phases of the process, from data collection and organization to feature extraction and processing.

The study examines the *Charoenkwan2020* dataset [3], including 626 samples (313 positive and 313 negative). Since the dataset is balanced, we focus on core preprocessing steps: removing duplicates and sequences containing non-canonical residues (e.g., B, J, O, U, and X) [30], followed by redundancy reduction using CD-HIT [13] with a 40% similarity threshold. Moreover,

Our proposed methodology provides an *innovative DL-based* technique for recognizing PVPs. Fig. 1 depicts our experimental approach. Initially, benchmark datasets from previous studies are obtained and preprocessed to improve data quality, which is already described in the above Section. Following that, four different feature extraction approaches are used, and their results are combined to form multi-view features, providing an in-depth assessment of the model's capacity to cope with various feature representations. To classify PVPs, 32 ML and DL models are evaluated. Four models, ANN, CNN, LSTM, and a CNN-BiLSTM hybrid model called DLPVP, are considered for the main discussion because of their better performance, with the remaining models covered in the supplemental materials. All models are rigorously validated with a variety of performance measures, including *Accuracy* (*Acc*), *Matthew's Correlation Coefficient* (MCC), *Cohen's Kappa Score* (Kappa), *Sensitivity* (*Sn*), *Precision* (*Pr*), *F1-score* (F1), *Area Under Curve* (AUC), and *Specificity* (*Sp*) for each feature extraction method. In addition, an ablation study is conducted to evaluate the fundamental functions of the models, and an adaptability test is performed using a different data set to verify the model's generalization capabilities. The findings show that DL-based models, notably DLPVP, perform better in detecting PVPs.

In our work, feature extraction converts protein sequences into fixed-length numerical representations suitable for ML/DL models. We consider four complementary views: N-gram composition, terminal amino-acid composition (AAC_NT), SOC, and transformer-based embeddings (BERT), which together capture local motifs, terminal signals, global order, and contextual dependencies.

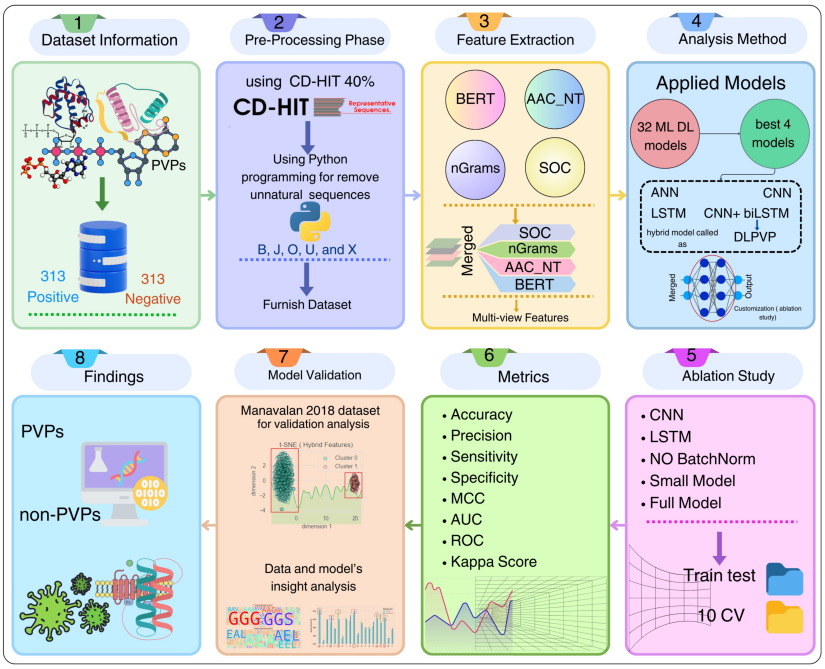


Fig. 1. Schematic Overview of the Proposed DLPVP Framework for PVP Prediction

One step of our method is focused on *N-Gram Composition*. Here, we compute normalized frequencies of di- and tri-peptide N-grams ($n \in \{2, 3\}$) and retain a fixed-length representation by selecting the top $n\_samples$ (400) N-gram features.

Further, we compute amino-acid composition at terminal regions by counting the occurrence of the 20 canonical residues at the N- and C-termini, yielding a 20-dimensional vector.

On the other hand, the *Sequence-Order-Coupling Number* (SOC) captures global sequence-order information using amino-acid distance matrices (Grantham and Schneider–Wrede) [27]. The SOC for order s is specified as follows:

$$\alpha_s = \sum_{i=1}^{T-s} (s_{i,i+s})^2, \quad s = 1, 2, 3, 4, 5 \ldots, tlag \tag{1}$$

where $tlag$ is the highest lag value (30) and T is the whole sequence size. This study used $2D$ features.

Finally, transformer-based embeddings provide contextual representations of protein sequences [21, 14]. In this study, we employ a BERT-style embedding and use a 767-dimensional feature vector per sequence.

4 Proposed Architecture

This Section provides a detailed examination of the main architectural elements and provides support for the design decisions to build the DLPVP model. We carried out an extensive evaluation across a wide range of 32 ML and DL models in order to evaluate the effectiveness and resilience of our suggested methodology. We select the top four models even though the majority produced positive results: CNN, ANN, LSTM, and a hybrid method known as DLPVP that combines CNN and BiLSTM. The ability to fully capture sequence patterns and the balanced evaluation parameters served as the basis for this selection. We aim to determine the ideal setup for precise PVPs prediction by analyzing how various models react to distinct feature sets.

DL methods are becoming crucial in the *Bioinformatics* and *Bioengineering* fields for detecting active and non-active sequences, which minimizes complexity, cost, and time consumption. Since ML models cannot always handle complicated sequences, DL has grown more prevalent. Fig. 2 illustrates the full architecture of our proposed DLPVP classifier. The model is designed as a *hybrid neural network* that integrates three successive *1-Dimensional Convolutional* (Conv1D) layers with two BiLSTM layers, followed by fully connected dense layers for final classification.

The feature extractor is composed of three Conv1D blocks (filters: 64-128-64, kernel size: 3) with batch normalization, ReLU activation, and max pooling (pool size: 2). The sequential modeling stage uses two BiLSTM layers (128 and 64 units), followed by a dense layer (64 units) with dropout (rate = 0.50). The final sigmoid output produces the probability of a sequence being a PVP.

For what regards the parameters of the four models based on the architecture used for our research, here we provide a detailed description. As mentioned before, we primarily focused on DL-based models, Table 1 provides a comprehensive overview of the detailed architectural parameters and hyperparameters configured for the four DL models. The main parameters are outlined as follows.

- *Layer Configurations*: This includes the number and types of layers (such as dense, Conv1D, LSTM, and BiLSTM) as well as the corresponding units, filters, or kernel sizes.
- *Activation Functions*: the specific activation functions used at various layers, such as *ReLU* and *Sigmoid*, and their application within hidden and output layers.
- *Regularization Technique*: the models implement *Batch Normalization* and *Dropout* layers, and their respective rates are configured to reduce overfitting.
- *Pooling Layers*: for CNN and DLPVP, the *MaxPooling1D* layers are configured with specific pool sizes.
- *Input Shapes*: The expected input dimensions for the initial layers of the models are well established.
- *Optimization Strategy*: The optimizer (e.g., Adam) and loss function (e.g., Binary Crossentropy) used for training are explicitly specified, reflecting the binary classification job objective.

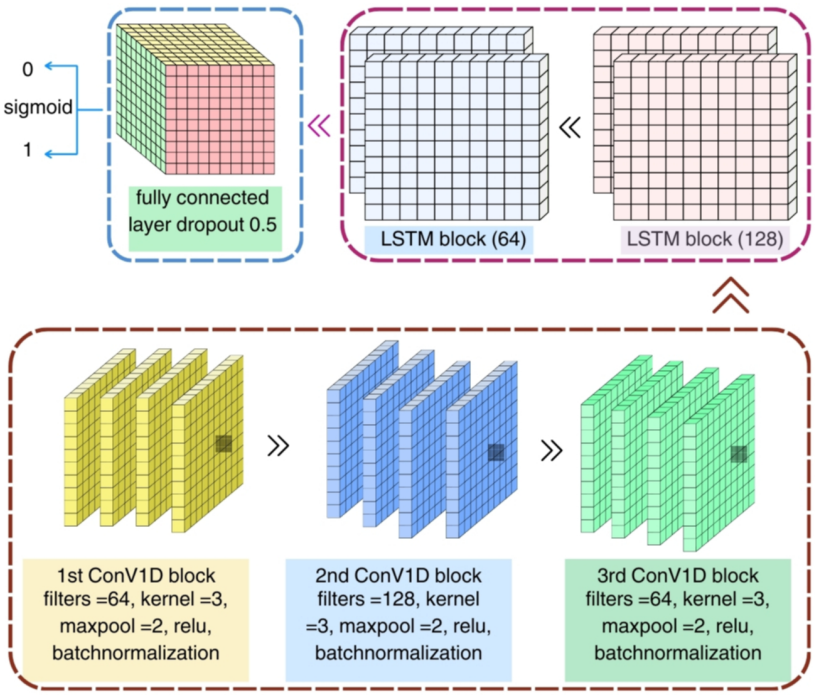


Fig. 2. DLPVP Architecture using Three ConV1D Blocks, Two Bidirectional-LSTM Layers, Fully Connected Layers, and Finally Output Layers of the Prediction

5 Experimental Results and Their Analysis

In this Section, we explain the experimental approach while evaluating the study outcomes. We used independent testing (train 80% and test 20%) and *10-fold cross-validation* with *multiple evaluation parameters*, as described in Section 3. Table 2 shows the performance of four specified models using these two analytical methods. For more insight, Fig. 3 compares the performance of all 32 models and clarifies why DLPVP was selected as the ideal strategy.

According to the results, the DLPVP model outperforms other models on all evaluation measures. Specifically, when multidimensional features ($1189D$) are used, the model improves the 10-fold cross-validation approach. In the N-Grams feature set, DLPVP improves CV accuracy by more than 10%, reaching 0.917 accuracy while maintaining balanced performance across other measures. In the independent test, the model achieves an accuracy of 0.7619, surpassing other models by over 2%. For the AAC_NT feature set, the model scores 0.8834 accuracy in CV but only 0.50 in both CV and independent testing, with a sensitivity of only 0.10, showing difficulties detecting the negative class. In contrast, employing SOC-based features, our proposed approach obtains 0.6407 and 0.6111 accuracy in CV and independent testing, respectively. When every

Table 1. Model Parameters Used in the Current Study

Model	Parameters
ANN	Number of neurons: 128, 64, 32, 1, Activation functions: ReLU, Sigmoid, Batch Normalization: Yes, Dropout rate: 0.30, Loss function: Binary Crossentropy, Optimizer: Adam
CNN	Conv1D: Filters: 64, 128, 64, Kernel size: 3, Input shape: input_shape, BatchNormalization: Yes, Activation: ReLU (for hidden layers), Sigmoid (for output layer), MaxPooling1D: Pool size: 2, Dense: Units: 64 (hidden layer), 1 (output layer), Dropout: Rate: 0.50, Optimizer: Adam, Loss Function: Binary Crossentropy
LSTM	LSTM: Units: 128 (first LSTM layer), 64 (second LSTM layer), Return sequences: True (for the first LSTM layer), False (for the second LSTM layer), Input shape: input_shape, Dropout: Rate: 0.30 (for both LSTM layers), 0.50 (for the dense layer), Dense: Units: 64 (hidden layer), 1 (output layer), Activation: ReLU (hidden layer), Sigmoid (output layer), Optimizer: Adam, Loss Function: Binary Crossentropy
CNN_BiLSTM (DLPVP)	Conv1D: filters=64, kernel_size=3, input_shape=(X_train.shape[1], 1), filters=128, kernel_size=3, filters=64, kernel_size=3, BatchNormalization: Yes, Activation: 'sigmoid', 'relu', 'relu', MaxPooling1D: pool_size=2, Bidirectional: LSTM(128, return_sequences=True), LSTM(64), Dense: 64, 1, activation='relu', activation='sigmoid', Dropout: 0.50, Optimizer: 'Adam', Loss: 'binary_crossentropy'

feature is combined into a multidimensional representation, the model scores most well in CV, with an accuracy of 0.9428 and an AUC score of 0.9808. This demonstrates that the model is capable of learning complex shapes and distinguishing between positive and negative classes. The model also achieves balanced sensitivity (0.9841) and specificity (0.8954), along with high MCC and Kappa values. On the other hand, out of all the models, DLPVP has the highest accuracy in the independent test (0.7698). A crucial research concern could arise. *Why use this model when the independent test accuracy is lower?* The response is in the dataset size because our dataset is quite small, cross-validation is the ideal strategy for robust evaluation. By performing *10-fold CV*, the model receives significant training on several dataset splits, provided that it can generalize effectively. This demonstrates that, with multi-feature integration and cross-validation, DLPVP effectively identifies PVPs.

Table 2. Performance Analysis based on Applied Models on the Feature Extraction using Independent Test and Cross-Validation Mode

Mode	Features	Model	Acc	MCC	Kappa	Sp	Sn	AUC	Pr
Cross-Validation	ngrams	ANN	0.7796	0.5605	0.5579	0.7825	0.7796	0.8338	0.8099
		CNN	0.7526	0.5142	0.5019	0.6922	0.8117	0.8410	0.8197
		LSTM	0.6214	0.2585	0.2472	0.5853	0.6673	0.7194	0.7281
		DLPVP	**0.9174**	**0.8376**	**0.8351**	**0.8912**	**0.9451**	**0.9546**	**0.9566**
	AAC NT	ANN	0.7813	0.5661	0.5603	0.7908	0.7740	0.8316	0.8072
		CNN	0.7463	0.5029	0.4928	0.6860	0.8108	0.8246	0.8096
		LSTM	0.6485	0.3064	0.2994	0.6121	0.6909	0.7238	0.7337
		DLPVP	**0.8834**	**0.7728**	**0.7631**	**0.8902**	**0.8733**	**0.9454**	**0.9414**
	SOC	ANN	**0.6774**	0.3555	**0.3495**	0.6032	0.7473	0.7158	0.6828
		CNN	0.6488	0.3596	0.2966	0.3656	**0.9327**	0.6559	0.6002
		LSTM	0.6407	0.3169	0.2772	0.4081	0.8681	0.7133	0.6937
		DLPVP	0.6646	0.3284	0.3242	**0.6394**	0.6871	**0.7169**	**0.7003**
	BERT	ANN	0.6489	0.3655	0.2963	0.3553	0.9435	0.6568	0.6093
		CNN	0.6489	0.3553	0.2961	**0.3684**	0.9288	0.6520	0.5956
		LSTM	0.5688	0.1482	0.1386	0.5807	0.5589	0.6202	0.5845
		DLPVP	**0.6649**	**0.4264**	**0.3291**	0.3494	**0.9825**	**0.6820**	**0.6347**
	Multi-view	ANN	0.7892	0.5926	0.5784	0.7744	0.8114	0.8622	0.8455
		CNN	0.7780	0.5613	0.5543	0.7695	0.7897	0.8713	0.8384
		LSTM	0.5943	0.2345	0.1973	0.5531	0.6464	0.6249	0.5873
		DLPVP	**0.9428**	**0.8913**	**0.8811**	**0.8954**	**0.9841**	**0.9808**	**0.9860**
Independent Test	ngrams	ANN	0.7460	0.4943	0.4921	0.7937	0.6984	0.8234	0.8147
		CNN	0.7460	0.5047	0.4921	0.6349	**0.8571**	**0.8360**	**0.8330**
		LSTM	0.6270	0.2579	0.2540	0.5397	0.7143	0.7329	0.7492
		DLPVP	**0.7619**	**0.5249**	**0.5238**	**0.7937**	0.7302	0.8007	0.7821
	AAC NT	**ANN**	**0.7460**	**0.4931**	**0.4921**	0.7778	0.7143	0.8403	0.8159
		CNN	0.7381	0.4792	0.4762	0.6825	**0.7937**	**0.8423**	**0.8275**
		LSTM	0.6032	0.2883	0.2063	**0.9524**	0.2540	0.6326	0.7114
		DLPVP	0.5000	0.000	0.000	0.000	1.000	0.5810	0.6203
	SOC	**ANN**	**0.6587**	**0.3269**	**0.3175**	**0.5397**	0.7778	0.6655	0.6088
		CNN	0.6032	0.2587	0.2063	0.3016	0.9048	0.6082	0.5606
		LSTM	0.5794	0.2091	0.1587	0.2540	**0.9048**	**0.6796**	**0.6161**
		DLPVP	0.6111	0.2553	0.2222	0.3651	0.8571	0.6620	0.6069
	BERT	ANN	0.6032	0.2671	0.2063	0.2857	0.9206	0.6135	**0.5763**
		CNN	0.6111	0.2830	0.2222	0.3016	0.9206	**0.6309**	0.5288
		LSTM	0.5794	0.2021	0.1587	0.2698	0.8889	0.5669	0.5468
		DLPVP	**0.6111**	**0.2830**	**0.2222**	**0.3016**	**0.9206**	0.6160	0.5704
	Multi-view	ANN	0.7578	0.5106	0.5156	0.7213	0.7143	0.8452	0.7566
		CNN	0.7457	0.5307	0.5214	0.7219	0.7795	0.8213	0.7742
		LSTM	0.5000	0.000	0.000	1.000	0.000	0.5671	0.5419
		DLPVP	**0.7698**	**0.5405**	**0.5399**	0.7500	**0.7903**	**0.8465**	**0.7756**

5.1 Ablation Study

In this study, we apply the ablation approach [10] to understand the model's functioning while evaluating the impact of various components. Table 3 shows the findings of our ablation study, where we constructed a reduced version of the model to determine if removing certain Conv1D layers would still produce satisfactory outcomes. However, our findings demonstrate that the proposed architecture remains the most appropriate strategy for detecting PVP. Specifically,

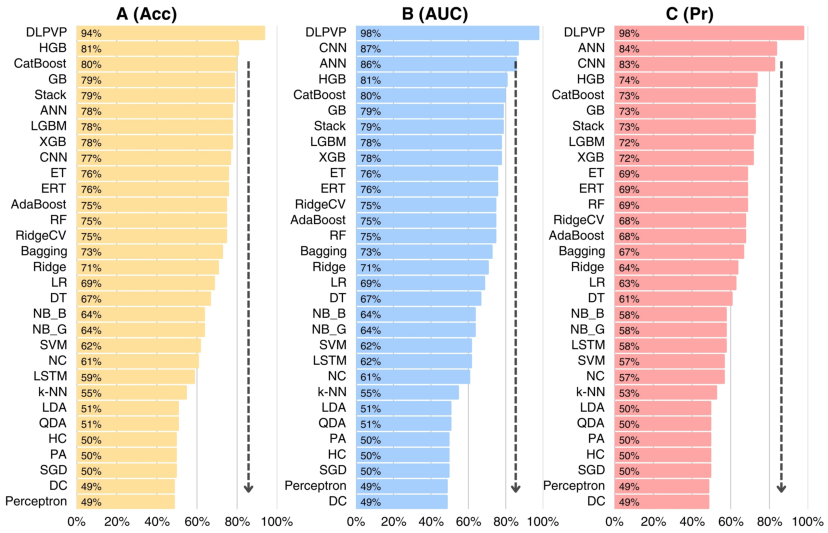


Fig. 3. Comparison between 32 Applied Models based on Accuracy (A), AUC Score (B), and Precision (C)

the ablation exploration reveals that removing batch normalization results in a significant loss in accuracy (0.4758), although eliminating dropout improves performance significantly (0.9476). This is due to the fact that dropout was only applied in the last layer, maintaining stability throughout the model. A single CNN obtains an accuracy of 0.7780, whilst a single LSTM performs more poorly at 0.5943. The smaller model, despite performing well with an accuracy of 0.8984, does not surpass the full model, which maintains an accuracy of 0.9428 ± 0.0015 with a well-balanced AUC score of 0.9808 ± 0.0003. These findings highlight the importance of each layer in the construction of the DLPVP model, highlighting the necessity for a complete model architecture for successful PVP classification.

Table 3. Ablation Study of the Proposed Framework

Model Features	Ablation Procedure	ACC	MCC	Kappa	AUC	F1
DLPVP Multiview	No BatchNorm	0.4758	0.0139	0.0100	0.6060	0.3799
	No Dropout	**0.9476**	**0.8947**	**0.8934**	0.9646	0.9502
	CNN	0.7780	0.5613	0.5543	0.8713	0.7757
	LSTM	0.5943	0.2345	0.1973	0.6249	0.5104
	Small Model	0.8984	0.7964	0.7942	0.9588	0.8516
	Full Model	**0.9428 ±0.0015**	**0.8913 ±0.019**	**0.8811 ±0.0098**	**0.9808 ±0.0003**	**0.9539**

6 Conclusions

In this study, we proposed DLPVP, a hybrid DL predictor that performed proficiently in PVPs detection using four different feature extraction approaches and multiple traditional algorithms. Our model obtained 94.28% accuracy on multidimensional features, demonstrating its adaptability and practical application. In addition, we conducted an ablation analysis to obtain more information about the contributions of various model layers, which is significant for understanding the model's architecture and predictive capabilities. While this study used a smaller dataset, in future research, we plan to broaden the dataset and integrate other feature extraction approaches to increase the model's potential in computational biology. On the other hand, we plan to integrate our framework with modern and emerging big data techniques (e.g., [7, 8, 2, 9]).

References

1. Ahmad, S., Charoenkwan, P., Quinn, J., Moni, M., Hasan, M., Lio', P., Shoombuatong, W.: Scorpion is a stacking-based ensemble learning framework for accurate prediction of phage virion proteins. Scientific Reports **12**(1), 4106 (2022)
2. Ceci, M., Cuzzocrea, A., Malerba, D.: Effectively and efficiently supporting roll-up and drill-down OLAP operations over continuous dimensions via hierarchical clustering. Journal of Intelligent Information Systems **44**(3), 309–333 (2015)
3. Charoenkwan, P., Kanthawong, S., Schaduangrat, N., Yana, J., Shoombuatong, W.: Pvpred-scm: improved prediction and analysis of phage virion proteins using a scoring card method. Cells **9**(2), 353 (2020)
4. Charoenkwan, P., Nantasenamat, C., Hasan, M., Shoombuatong, W.: Meta-ipvp: a sequence-based meta-predictor for improving the prediction of phage virion proteins using effective feature representation. Journal of Computer-Aided Molecular Design **34**(10), 1105–1116 (2020)
5. Chen, W., Nie, F., Ding, H.: Recent advances of computational methods for identifying bacteriophage virion proteins. Protein and Peptide Letters **27**(4), 259–264 (2020)
6. Cobián Güemes, A., Youle, M., Cantú, V., Felts, B., Nulton, J., Rohwer, F.: Viruses as winners in the game of life. Annual Review of Virology **3**(1), 197–214 (2016)
7. Cuzzocrea, A.: Overcoming limitations of approximate query answering in OLAP. In: IEEE (IDEAS 2005: International Conference. pp. 200–209 (2005)
8. Cuzzocrea, A.: Accuracy control in compressed multidimensional data cubes for quality of answer-based OLAP tools. In: IEEE SSDBM 2006: International Conference. pp. 301–310 (2006)
9. Cuzzocrea, A., Fortino, G., Rana, O.F.: Managing data and processes in cloud-enabled large-scale sensor networks: State-of-the-art and future research directions. In: 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013, Delft, Netherlands, May 13-16, 2013. pp. 583–588. IEEE Computer Society (2013)
10. Du, L.: How much deep learning does neural style transfer really need? an ablation study. In: IEEE WACV 2020 International Conference. pp. 3139–3148 (2020)
11. Fang, Z., Feng, T., Zhou, H.: Deepvp: Identification and classification of phage virion protein using deep learning. Gigascience **11**, giac076 (2021)

12. Fernández, L., Rodríguez, A., García, P.: Phage or foe: an insight into the impact of viral predation on microbial communities. The ISME journal **12**(5), 1171–1179 (2018)
13. Huang, Y., Niu, B., Gao, Y., Fu, L., Li, W.: CD-HIT suite: a web server for clustering and comparing biological sequences. Bioinformatics **26**(5), 680–682 (2010)
14. Huo, H., Iwaihara, M.: Utilizing BERT pretrained models with various fine-tune methods for subjectivity detection. In: Web and Big Data 2020: International Conference. pp. 270–284 (2020)
15. Kabir, M., Nantasenamat, C., Kanthawong, S., Charoenkwan, P., Shoombuatong, W.: Large-scale comparative review and assessment of computational methods for phage virion proteins identification. EXCLI Journal **21**, 11–30 (2022)
16. Lekunberri, I., Subirats, J., Borrego, C., Balcázar, J.: Exploring the contribution of bacteriophages to antibiotic resistance. Environmental Pollution **220**, 981–984 (2017)
17. Li, Z., Liu, F., Yang, W., Peng, S., Zhou, J.: A survey of convolutional neural networks: Analysis, applications, and prospects. IEEE Transactions on Neural Networks and Learning System **33**(12), 6999–7019 (2022)
18. Lin, D., Koskella, B., Lin, H.: Phage therapy: An alternative to antibiotics in the age of multi-drug resistance. World Journal of Gastrointestinal Pharmacology and Therapeutics **8**(3), 162 (2017)
19. Liu, Q., Wan, J., Wang, G.: A survey on computational methods in discovering protein inhibitors of sars-cov-2. Briefings in Bioinformatics **23**(1), bbab416 (2022)
20. Liu, S., Cui, C., Chen, H., Liu, T.: Ensemble learning-based feature selection for phage protein prediction. Frontiers in Microbiology **13**, 932661 (07 2022)
21. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized BERT pretraining approach. CoRR **abs/1907.11692** (2019)
22. Mahmud, M., Kaiser, M., Hussain, A., Vassanelli, S.: Applications of deep learning and reinforcement learning to biological data. IEEE Transactions on Neural Networks and Learning Systems **29**(6), 2063–2079 (2018)
23. Memisevic, R.: On multi-view feature learning. In: ICML 2012: International Conference (2012)
24. Meng, C., Zhang, J., Ye, X., Guo, F., Zou, Q.: Review and comparative analysis of machine learning-based phage virion protein identification methods. Biochimica et Biophysica Acta (BBA)-Proteins and Proteomics **1868**(6), 140406 (2020)
25. Oechslin, F.: Resistance development to bacteriophages occurring during bacteriophage therapy. Viruses **10**(7), 351 (2018)
26. Pan, J., You, W., Lu, X., Wang, S., You, Z., Sun, Y.: Gsphi: A novel deep learning model for predicting phage-host interactions via multiple biological information. Computational and Structural Biotechnology Journal **21**, 3404–3413 (2023)
27. Schneider, G., Wrede, P.: The rational design of amino acid sequences by artificial neural networks and simulated molecular evolution: de novo design of an idealized leader peptidase cleavage site. Biophysical Journal **66**(2), 335–344 (1994)
28. Shang, J., Peng, C., Tang, X., Sun, Y.: Phavip: Phage virion protein classification based on chaos game representation and vision transformer. Bioinformatics **39**(1), i30–i39 (2023)
29. Thireou, T., Reczko, M.: Bidirectional long short-term memory networks for predicting the subcellular localization of eukaryotic proteins. IEEE/ACM Transactions on Computational Biology and Bioinformatics **4**(3), 441–446 (2007)
30. Veltri, D., Kamath, U., Shehu, A.: Deep learning improves antimicrobial peptide recognition. Bioinformatics **34**(16), 2740–2747 (2018)

Evaluating Human Activity Recognition Methods for Variable-Duration Occupational Activities in Healthcare

Djalal Bouchekif[1 [0009−0008−3833−0108]], Imen Megdiche[1 [0000−0002−1331−8662]], and Rémi Bastide[1 [0000−0002−2231−0750]]

Toulouse University, INUC, Toulouse INP, CNRS, IRIT, Toulouse, France
{djalal.bouchekif,imen.megdiche,remi.bastide}@irit.fr

Abstract. Monitoring fatigue and physical workload contributes to occupational health, particularly for the prevention of accidents and work-related disorders in high-risk environments. Human Activity Recognition (HAR) methods enable the automatic identification of activities in professional contexts. However, most state-of-the-art HAR approaches based on wearable sensors rely on algorithms trained to classify fixed-size data windows. These algorithms are moderately suited to occupational activities with variable durations. In this study, we evaluate several existing deep learning-based HAR architectures on benchmark HAR datasets as well as on a real annotated dataset collected in healthcare environments, including occupational activities with variable durations. We analyze their ability to accurately recognize activities across different temporal scales. The results indicate that classical HAR methods experience significant performance degradation when confronted with long- or variable-duration activities. This work highlights the limitations of current HAR techniques for monitoring occupational activities and emphasizes the need to adapt algorithms to variable-duration activities in real-world settings. This study is exploratory in nature and aims to better understand the behavior of HAR models under real-world conditions.

Keywords: Human Activity Recognition · Wearable Sensors · Activity Segmentation · Variable-Duration Activities · Healthcare

1 Introduction

Work-related physical workload represents a major issue in terms of health and performance, both for workers and organizations. It manifests through repeated physical efforts, high workload, and an increased risk of musculoskeletal disorders (MSDs), leading to discomfort, pain, or functional impairment [1]. Monitoring this physical workload is therefore essential to prevent accidents and occupational diseases, improve ergonomics, and optimize workers' performance. Traditionally, the assessment of physical workload relies on subjective methods, such as questionnaires [2], perceived exertion scales [3], or self-reported data. Although widely used, these approaches present several limitations: they lack objectivity,

do not allow continuous monitoring, and are sensitive to inter-individual variability. They also struggle to capture the complexity of real-world situations, in which the duration, intensity, and nature of activities can vary considerably [4].

To overcome these limitations, more objective approaches have emerged based on data collected from various inertial measurement units (IMUs), which comprise wearable sensors such as accelerometers, gyroscopes, and magnetometers that are commonly embedded in smartphones [5], smartwatches, activity trackers, and other available devices [6]. These technologies provide continuous, reproducible measurements of movement and posture, thereby enabling a more precise assessment of physical workload under real working conditions [7].

In this context, Human Activity Recognition (HAR) plays a central role in effectively exploiting this sensor data, as it enables the automatic detection and classification of human activities from raw inertial signals measured by these devices [8]. HAR approaches are now applied in various domains, including healthcare monitoring, occupational risk prevention, and ergonomics research [9]. The combined use of movement and physiological signals further enables fine-grained monitoring of physical workload, allowing precise identification of periods of high effort or potentially fatiguing tasks [10].

However, despite significant advances in the field of Human Activity Recognition (HAR), this task remains challenging due to the diversity and complexity of human activities. One of the main challenges is to develop models capable of effectively recognizing complex, long-duration activities that exhibit substantial inter-individual variability. Most existing studies are limited to relatively simple experimental protocols or well-segmented activities. Researchers have explored a wide range of HAR techniques, ranging from traditional machine learning methods [11] to advanced deep learning architectures [12]. Conventional algorithms such as Support Vector Machines (SVM), k-Nearest Neighbors (KNN), and Decision Trees have demonstrated effectiveness in activity classification when using handcrafted features extracted from sensor data [13]. However, these methods are limited in their ability to capture the complex temporal dependencies inherent in human activities, which poses challenges for recognizing sequential patterns over time [14]. To overcome these limitations, the research community has increasingly adopted deep learning models such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and other artificial neural network (ANN) architectures, with the aim of improving recognition accuracy across a wide range of HAR tasks [15]. Deep learning approaches can automatically learn hierarchical representations from raw sensor signals, facilitating more robust feature extraction and better modeling of both spatial and temporal patterns in activity data [16].

The objective of this study is to evaluate several HAR methods on benchmark datasets and to compare their performance with that on a real-world dataset representing occupational activities of variable duration, collected in healthcare environments. We analyze their ability to accurately recognize activities of varying complexity and identify both current limitations and necessary improvements to adapt HAR approaches to the requirements of continuous workload monitor-

ing under real-life conditions. This study adopts an exploratory perspective and aims to identify the underlying factors that may explain the observed performance degradation, as well as to provide insights into the limitations of current HAR approaches in real-world settings.

The contributions of this study can be summarized as follows:

- Comprehensive evaluation of deep learning-based HAR models on both benchmark datasets (UCI-HAR, WISDM, PAMAP2, OPPORTUNITY) and a real-world healthcare dataset (PEMESA HAR) representing variable-duration occupational activities.
- Identification of limitations of current HAR architectures for variable-duration activities and suggestions for improvements in model design and temporal segmentation strategies.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces the problem formulation and the datasets used in this study. Section 4 reports the experimental setup and results. Section 5 discusses the limitations and future research directions, and Section 6 concludes the paper.

2 Related Work

Human activities consist of physical actions and movements performed in daily life, such as walking, sitting, running, or eating. Human Activity Recognition (HAR) aims to automatically recognize these activities from sensor data streams and has evolved significantly over the years [17]. Early HAR approaches were primarily based on traditional signal processing and machine learning techniques. These methods relied on handcrafted feature extraction from inertial signals, followed by classification using supervised learning algorithms. Techniques such as the Fourier transform, wavelet transform, and Principal Component Analysis (PCA) were commonly employed to reduce dimensionality and extract discriminative features in both the time and frequency domains [18]. After feature extraction, several machine learning algorithms such as Decision Trees, Random Forests, k-NN, ANN, and SVM were applied for activity classification. Overall, traditional machine learning methods are effective in learning discriminative patterns from labeled data. However, these approaches exhibit notable limitations, particularly in capturing complex temporal dependencies inherent in human activities, which are sequential and dynamic in nature. In addition, they rely heavily on handcrafted feature engineering and lack the ability to automatically learn high-level representations from raw sensor data.

To address these limitations, recent HAR research has progressively shifted toward deep learning and hybrid architectures capable of automatically learning spatio-temporal representations from raw sensor data and improving recognition performance [19]. Recent advances in HAR have been largely driven by deep learning models, which have demonstrated strong performance compared to traditional machine learning approaches.

Various architectures have been explored, including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), as well as hybrid and attention-based models [15]. Hybrid deep learning architectures have been widely investigated in recent studies. In particular, CNN–LSTM models combine convolutional layers for spatial feature extraction with LSTM layers for sequential pattern modeling, leading to improved recognition performance [21]. For example, [22] reported that LSTM-based models achieved an accuracy of 92%, while hybrid CNN–LSTM models reached up to 99% on both an internal dataset and the UCI HAR dataset, illustrating the benefit of combining spatial and temporal representations. More recently, attention-based and transformer-based architectures have attracted increasing interest in HAR. For instance, [23] showed that a transformer-based model can effectively capture temporal dependencies in inertial sensor data, achieving an accuracy of 99.2% on a smartphone-based dataset, compared to 89.67% obtained with conventional machine learning approaches. These results indicate that attention mechanisms can enhance the modeling of long-range dependencies in time-series data, although their performance may depend on the dataset and experimental setup. Despite the strong performance of deep learning models, several challenges remain. In particular, many HAR approaches rely on fixed-size sliding window segmentation, which limits their ability to model activities of variable duration and to capture long-range temporal dependencies across windows. This limitation is well documented in recent studies, where sliding window approaches restrict the modeling of continuous real-world activities. To address these limitations, recent work has explored Temporal Action Localization (TAL) approaches, originally developed for video-based activity recognition and later adapted to inertial sensor data. These methods enable the detection of activity segments with arbitrary duration without predefined segmentation. Experimental results suggest that TAL-based approaches can improve performance and temporal consistency in HAR tasks, achieving significant gains in F1-score compared to conventional window-based models [24]. However, the evaluation of these approaches on long-duration occupational activities performed in real-world healthcare environments remains limited.

3 Overview of Human Activity Recognition

3.1 Hierarchical Structure of Human Activities

Humans perform a wide range of activities in their daily lives. The structure of these activities can be hierarchical, ranging from elementary actions to complex tasks. In the literature, human activity is defined as a sequence of actions performed by one or more individuals [9]. As illustrated in Figure 1, human activity can be divided into three distinct levels (gestures, actions, and activities) that vary in complexity and duration [27]. At the lowest level are gestures, corresponding to elementary, repetitive movements (e.g., hand movements). At an intermediate level are actions, which consist of combinations of gestures directed toward specific objectives. At the highest and most complex level are activities, which

represent coordinated sets of actions forming meaningful occupational tasks or professional processes. In practice, most existing HAR approaches primarily focus on the two first levels, namely gestures and actions, while the recognition of complex, long-term professional activities remains relatively less explored. In this context, occupational activities are considered the most complex category, involving extended temporal structure, multiple coordinated actions, and goal-oriented behavior. The ultimate goal of Human Activity Recognition (HAR) is to train algorithms to accurately identify and recognize these different types of activities, using various state-of-the-art techniques [28].

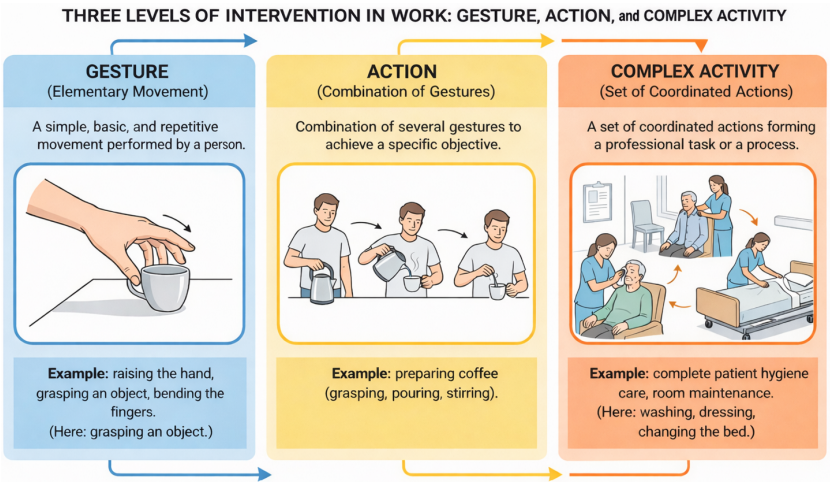


Fig. 1. Hierarchy of human activities: simple gestures, actions, and complex activities.

3.2 Mathematical Formulation of Human Activity Recognition

Suppose a person performs a predefined set of n activities, A, which can be represented as:

$$A = \{A_0, A_1, \ldots, A_{n-1}\} \tag{1}$$

To recognize these activities, multiple sensors are employed to measure a set of attributes S, consisting of k time series over a time interval $I = [t_\alpha, t_\omega]$:

$$S = \{S_0, S_1, \ldots, S_{k-1}\} \tag{2}$$

The objective of Human Activity Recognition (HAR) is to determine a temporal partition of the interval I, using the set of attributes S, along with a collection of classes of Activity A describing the activities.

The main assumption is that each subinterval I_j is continuous and non-overlapping, and that the union of all subintervals covers the entire interval I:

$$\bigcup_{j=0}^{r-1} I_j = I, \quad I_j \cap I_{j'} = \emptyset \quad \text{for } j \neq j'. \tag{3}$$

This formulation provides a mathematical foundation for modeling sensor- and vision-based HAR systems, treating activities as partitioned temporal sequences [29].

3.3 Benchmark Datasets for Human Activity Recognition

A key aspect of our study lies in the careful selection of suitable datasets recording primarily inertial sensor data from adult subjects, widely adopted in human activity recognition (HAR) research. To ensure a fair and relevant comparison, four publicly accessible benchmark datasets were selected: UCI-HAR (Human Activity Recognition Using Smartphones, cited 3575 times), PAMAP2 (Physical Activity Monitoring, cited 1901 times), WISDM (Smartphone and Smartwatch Activity and Biometrics (cited 3950 times), and Opportunity Activity Recognition (cited 980 times). The following sections present an overview of these datasets and outline their main characteristics.

UCI-HAR

The *UCI-HAR* [30] dataset is a collection of inertial sensor measurements obtained from 30 volunteers aged between 19 and 48 years. The signals were recorded using triaxial accelerometers and gyroscopes embedded in a smartphone worn at the waist, with a sampling frequency of 50 Hz. Each participant performed six daily activities: walking, walking upstairs, walking downstairs, sitting, standing, and lying. From the raw signals, a total of 561 handcrafted features were extracted for each sensor window. The dataset was split into training (70% of the participants) and test (30% of the participants) sets according to the standard benchmark protocol. Overall, it contains 748,406 individual samples. Table 1 summarizes the proportion of data associated with each activity class in the dataset.

Table 1. Class distribution in the UCI-HAR dataset

Class	Proportion (%)
Lying	18.88
Standing	18.51
Sitting	17.25
Walking	16.72
Walking upstairs	14.99
Walking downstairs	13.65

PAMAP2

The *PAMAP2* [31] dataset comprises multimodal recordings collected from inertial sensors placed on three body locations (wrist, chest, and ankle), along with heart rate data, from 9 subjects performing 18 distinct physical activities. The sensors include triaxial accelerometers, gyroscopes, and magnetometers, sampling at a frequency of 100 Hz. This dataset represents a rich benchmark for training and evaluating human activity recognition algorithms, as it supports the full pipeline of data processing, segmentation, feature extraction, and classification, owing to the diversity of activities and sensor modalities. Table 2 reports the proportion of samples associated with each class in PAMAP2.

Table 2. Class distribution in the PAMAP2 dataset

Class	Proportion (%)
Walking	12.29
Ironing	12.29
Lying	9.90
Standing	9.78
Nordic walking	9.68
Sitting	9.53
Vacuum cleaning	9.02
Cycling	8.47
Ascending stairs	6.03
Descending stairs	5.40
Running	5.05
Rope jumping	2.54

WISDM (Wireless Sensor Data Mining)

The *WISDM* [32] dataset was collected from triaxial accelerometer signals recorded from 36 participants performing six common human activities, including walking, jogging, walking upstairs, walking downstairs, sitting, and standing. The signals were sampled at 20 Hz using a smartphone carried in the participants' pocket. This dataset has been widely adopted in the literature for training and evaluating human activity recognition models due to its large size and well-structured format. The class distribution in the *WISDM* dataset is reported in Table 3.

Opportunity

The *Opportunity* [33] dataset is a multimodal benchmark designed for human activity recognition (HAR) using wearable, mobile, and ambient sensors, with the aim of evaluating activity recognition algorithms across the full pipeline, including classification, automatic segmentation, sensor fusion, and feature extraction. Sensor data were collected from participants equipped with multiple wearable

Table 3. Class distribution in the WISDM dataset

Class	Proportion (%)
Walking downstairs	38.93
Jogging	30.23
Sitting	11.42
Standing	9.33
Walking upstairs	5.58
Walking	4.51

devices placed on the wrist, chest, hip, and dominant forearm. The recordings were sampled at 30 Hz. The dataset provides precisely annotated data from four subjects (male and female), supporting the perception and learning of a wide range of human activities, including short actions, gestures, locomotion modes, and high-level behaviors. Overall, the dataset contains detailed annotations for 35 activities, organized into 13 low-level categories, and integrates measurements from 23 on-body sensors, 12 object sensors, and 21 ambient sensors.

3.4 PEMESA HAR Dataset

Unlike most open-source datasets commonly used in the Human Activity Recognition (HAR) literature, which are typically limited to simple activities performed in controlled laboratory environments, very few datasets capture the richness, variability, and complexity of professional activities as they occur in real-world contexts. This limitation significantly reduces the transferability and validity of HAR models when deployed in real occupational applications.

To address this gap, we conducted a field study in a real care environment in collaboration with a project partner. This field phase combined structured observations and semi-structured interviews with healthcare staff, enabling the identification of two professional profiles particularly exposed to physical workload and biomechanical constraints: nursing assistants and hospital service agents (ASH). This preliminary phase was essential for defining the most relevant professional activities to be captured under real working conditions. As part of our overall objective to quantify work-related physical fatigue, we designed a multimodal data collection protocol (Figure 2) that combines objective measurements from IoT sensors (inertial, physiological, and environmental) with subjective self-assessments. This approach provides a more comprehensive representation of workload than methods relying on a single data modality.

The inertial signals used in this study were collected using the sensors embedded in a commercial smartwatch (Samsung Galaxy Watch 7), namely a tri-axial accelerometer and a tri-axial gyroscope, which measure linear acceleration and angular velocity along three axes, respectively. The signals were annotated in real time using a mobile application connected to the smartwatch, allowing each data sample to be associated with the corresponding professional activity observed in the field.

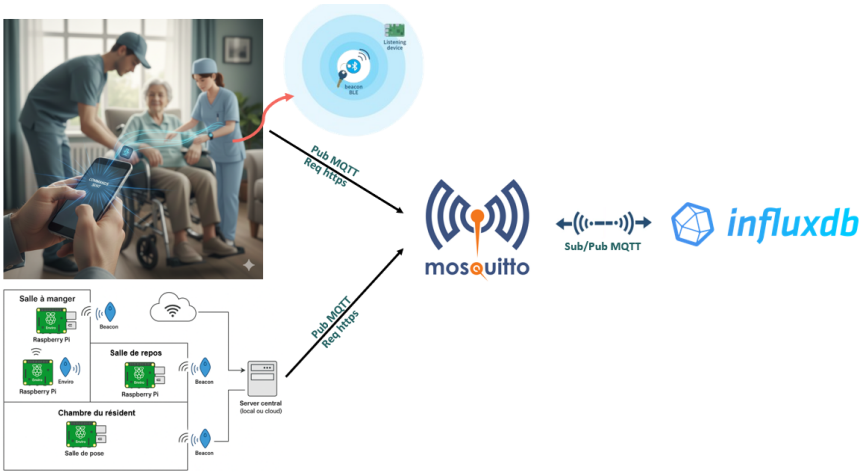


Fig. 2. Overview of the data collection protocol.

The resulting dataset, collected from four participants, comprises approximately 20 hours of annotated recordings, corresponding to more than 4 million synchronized inertial samples. The recordings are distributed across six classes of professional activities, as well as an additional "other" class representing activities not considered in this study. The distribution of activities in the PEMESA dataset is summarized in Table 4.

Table 4. Distribution of activities in the PEMESA dataset

Class	Number of Samples	Proportion
Assistance with bathroom bathing	354 400	42.2%
Assistance with bed bathing	220 177	26.2%
Positioning patient in wheelchair	40 177	4.8%
Bed preparation	50 427	6.0%
Cleaning the floors	24 939	3.0%
Cleaning the bathroom	150 395	17.9%

4 Experimentation and Evaluation

4.1 Experimental Setup

The training and evaluation of the different HAR models were conducted on the computing cluster of the OcciData platform at IRIT. All experiments were implemented using TensorFlow and executed under the same experimental conditions to ensure a fair comparison across models and datasets.

We used the Adam optimizer with an adaptive decaying learning rate to improve convergence during training. The initial learning rate was set to 1×10^{-4} and adjusted empirically for each dataset based on preliminary validation experiments. Due to the class imbalance observed in several datasets, class weights were computed according to the proportion of samples in each activity class and incorporated into the loss function during training.

Sensor signals were segmented using a fixed-size sliding window approach, which remains one of the most widely adopted segmentation strategies in HAR. Depending on the characteristics and complexity of the activities, the window length was set between 50 and 100 samples with a temporal overlap of 50%. Categorical cross-entropy was used as the optimization loss function. All models were trained for a maximum of 40 epochs with a batch size of 128. Early stopping based on validation loss was employed to reduce overfitting and improve model generalization. To comprehensively evaluate model performance, four commonly used metrics in HAR research were considered: accuracy, precision, recall, and F1-score [17]. Since several datasets exhibit significant class imbalance, particular attention was given to the F1-score, which provides a more reliable assessment of recognition performance across activity classes.

In this study, we evaluate several state-of-the-art deep learning architectures commonly used in HAR, including CNN, LSTM, Bi-LSTM, GRU, InceptionTime, FCN, CNN–LSTM hybrid architectures, and Transformer-based models. These architectures were selected because they represent different strategies for modeling spatial and temporal dependencies in inertial time-series data. In addition, we investigate Temporal Action Localization (TAL) approaches, which aim to detect activity segments with explicit temporal boundaries directly from continuous inertial sequences rather than classifying pre-segmented fixed windows.

4.2 Experimental Results

This section presents a comprehensive evaluation of the proposed deep learning models on the datasets introduced in Section 3.3. The numerical results are summarized in Table 5.

On the *UCI-HAR* dataset, all deep learning architectures achieve strong performance. Transformer-based models obtain the best overall results, reaching an accuracy and F1-score of 0.97, closely followed by the CNN–LSTM architecture with an F1-score of 0.96. Conventional recurrent architectures such as LSTM and GRU exhibit lower performance compared to hybrid and attention-based models. These results suggest that combining spatial feature extraction with long-range temporal modeling improves recognition performance on structured HAR datasets.

The *WISDM* dataset is more challenging due to the higher variability and noisier sensor signals. In this context, Transformer-based models again achieve the best performance with an F1-score of 0.72, while CNN–LSTM architectures obtain moderate results with an F1-score of 0.66. In contrast, standalone CNN and recurrent models achieve lower F1-scores ranging from 0.53 to 0.55. These

observations indicate that attention mechanisms are more robust to noisy and heterogeneous temporal sequences.

On the *PAMAP2* dataset, all evaluated architectures achieve very high performance, with F1-scores between 0.96 and 0.99. CNN and Bi-LSTM models obtain the best overall performance with an F1-score of 0.99, while Transformer-based and CNN–LSTM models also maintain excellent results. These consistently high scores indicate that the well-structured and highly segmented signals of PAMAP2 are effectively modeled by most deep learning architectures.

For the *OPPORTUNITY* dataset, which contains more complex and heterogeneous activities, CNN models achieve the best overall performance with an F1-score of 0.90. Transformer-based models also obtain competitive results with an F1-score of 0.88, followed by CNN–LSTM architectures with an F1-score of 0.85. In contrast, standalone recurrent approaches such as LSTM and GRU exhibit lower recognition performance. These results suggest that convolutional architectures remain highly effective for extracting local discriminative patterns in complex inertial signals, while hybrid and attention-based models provide improved temporal modeling capabilities.

The *PEMESA HAR* dataset represents the most challenging scenario among all evaluated datasets, as reflected by the lower overall performance across all architectures. Unlike benchmark datasets collected in controlled laboratory environments, PEMESA HAR contains long-duration occupational activities performed under real-world healthcare conditions. In this context, the Temporal Action Localization (TAL) approach achieves the best overall F1-score (0.60), slightly outperforming InceptionTime (0.58) and Transformer-based models (0.56). In addition, TAL also achieves the highest precision (0.62), while sharing similar accuracy (0.62) with InceptionTime. Conventional CNN and recurrent architectures achieve lower F1-scores around 0.50. These results suggest that approaches explicitly modeling temporal boundaries are better suited for variable-duration activities than conventional fixed-window classification methods.

Overall, the experimental results reveal a substantial performance gap between controlled benchmark datasets and real-world occupational activities. While most deep learning architectures achieve excellent performance on highly segmented laboratory datasets, their performance decreases significantly when confronted with long-duration, weakly structured, and heterogeneous activities. In particular, TAL-based approaches appear more suitable for handling variable-duration occupational activities in real-world healthcare environments. These findings highlight the limitations of current HAR approaches and emphasize the need for more adaptive temporal modeling strategies.

Table 5. Performance Metrics of Deep Learning Models

Dataset	Technique	Accuracy	Precision	Recall	F1-Score
UCI-HAR	CNN	0.95	0.95	0.95	0.95
	LSTM	0.82	0.83	0.82	0.82
	Bi-LSTM	0.85	0.85	0.84	0.84
	GRU	0.82	0.83	0.82	0.82
	CNN-LSTM	0.96	0.95	0.96	0.96
	Transformers	**0.97**	**0.96**	**0.97**	**0.97**
WISDM	CNN	0.63	0.62	0.57	0.55
	LSTM	0.63	0.61	0.56	0.54
	Bi-LSTM	0.63	0.61	0.57	0.54
	GRU	0.63	0.62	0.57	0.53
	CNN-LSTM	0.70	0.69	0.67	0.66
	Transformers	**0.75**	**0.74**	**0.72**	**0.72**
PAMAP2	CNN	**0.99**	**0.99**	**0.99**	**0.99**
	LSTM	0.98	0.98	0.98	0.98
	Bi-LSTM	**0.99**	**0.99**	**0.99**	**0.99**
	GRU	0.98	0.98	0.98	0.98
	CNN-LSTM	0.97	0.97	0.96	0.96
	Transformers	0.98	0.98	0.97	0.97
OPPORTUNITY	CNN	**0.89**	**0.91**	**0.90**	**0.90**
	LSTM	0.69	0.71	0.71	0.69
	Bi-LSTM	0.79	0.81	0.81	0.80
	GRU	0.70	0.69	0.74	0.70
	CNN-LSTM	0.85	0.86	0.85	0.85
	Transformers	0.88	0.89	0.88	0.88
PEMESA HAR	CNN	0.55	0.52	0.50	0.51
	LSTM	0.52	0.50	0.48	0.49
	Bi-LSTM	0.54	0.52	0.49	0.50
	GRU	0.53	0.51	0.48	0.49
	CNN-LSTM	0.58	0.56	0.53	0.54
	Transformers	0.60	0.58	0.55	0.56
	InceptionTime	0.62	0.60	0.57	0.58
	FCN	0.60	0.58	0.55	0.56
	TAL	**0.62**	**0.62**	**0.59**	**0.60**

5 Limitations and Future Work

Among the evaluated datasets, the PEMESA HAR dataset appears particularly challenging and reveals a substantial gap between the performance achieved on benchmark datasets and that obtained under real occupational conditions. This discrepancy can largely be explained by the intrinsic characteristics of professional activities, which are typically less structured, more heterogeneous, and considerably longer than the activities represented in public HAR benchmarks. In some cases, activities in the PEMESA HAR dataset may last up to 30 minutes, resulting in variable temporal structures and complex transitions between sub-actions. Under these conditions, segmentation strategies based on fixed-size sliding windows become insufficient, as they fail to capture long-range temporal dependencies and dynamic activity boundaries. Consequently, model performance decreases significantly in unconstrained operational environments.

Another important limitation concerns the generalization capability of current HAR approaches. Most existing models are trained and evaluated on datasets collected in controlled laboratory settings with limited variability in participants, sensor placement, environmental conditions, and acquisition devices. However, such settings only partially reflect the complexity and unpredictability of professional environments encountered in practice. From this perspective, the performance degradation observed on the PEMESA HAR dataset should not be interpreted as a dataset-specific issue, but rather as evidence of the broader challenges associated with deploying HAR systems in realistic occupational scenarios. Similar limitations are therefore likely to emerge when applying current HAR models to unseen operational domains.

Furthermore, this study mainly relies on global performance metrics such as accuracy and F1-score. While these indicators provide an overall assessment of model performance, they may conceal significant disparities between activity classes, particularly in the presence of class imbalance. Future work should therefore include more detailed per-class analyses and confusion matrix evaluations to better assess the robustness of models when recognizing minority, ambiguous, or highly variable activities.

Several research directions emerge from these observations. Future research should focus on adaptive temporal segmentation strategies and advanced hybrid architectures capable of modeling long-range temporal dependencies in variable-duration activities. Finally, integrating multimodal sensor data, including inertial, physiological, and environmental signals, represents a promising direction for enhancing the reliability and applicability of future HAR systems in occupational monitoring scenarios.

6 Conclusion

This study evaluated deep learning models for Human Activity Recognition (HAR) on several datasets (UCI-HAR, OPPORTUNITY, PAMAP2, WISDM, and PEMESA HAR). The results show strong performance on structured datasets

representing short activities, but a noticeable decline on datasets involving more complex activities collected in real-world conditions. These limitations highlight the inadequacy of fixed-size temporal windows and emphasize the need for more robust methods capable of better modeling the temporal variability and complexity of human activities.

Acknowledgments. This research was funded under the PEMESA project, supported by the European Union and awarded by the Occitanie Region. The authors would like to thank the medical partners involved in this project, in particular the AGIR nursing home (EHPAD), for granting access to their facilities and for their support in conducting the experimental studies.

References

1. Motta, F., Varrecchia, T., Chini, G., Ranavolo, A., Galli, M.: The use of wearable systems for assessing work-related risks related to the musculoskeletal system—A systematic review. International Journal of Environmental Research and Public Health **21**(12), 1567 (2024). https://doi.org/10.3390/ijerph21121567
2. Arsalan, A., Majid, M., Nizami, I.F., Manzoor, W., Anwar, S.M., Ryu, J.: Human stress assessment: A comprehensive review of methods using wearable sensors and non-wearable techniques. arXiv preprint arXiv:2202.03033 (2023). https://arxiv.org/abs/2202.03033
3. Morishita, S., Tsubaki, A., Takabayashi, T., Fu, J.B.: Relationship between the rating of perceived exertion scale and the load intensity of resistance training. Strength and Conditioning Journal **40**(2), 94–109 (2018). https://doi.org/10.1519/SSC.0000000000000373
4. Matthews, G., de Winter, J., Hancock, P.: What do subjective workload scales really measure? Operational and representational solutions to divergence of workload measures. Theoretical Issues in Ergonomics Science **21** (2019). https://doi.org/10.1080/1463922X.2018.1547459
5. Shoaib, M., Bosch, S., Incel, O.D., Scholten, H., Havinga, P.J.M.: A survey of online activity recognition using mobile phones. Sensors **15**(1), 2059–2085 (2015). https://doi.org/10.3390/s150102059
6. Naranjo, J.E., Mora, C.A., Bustamante Villagómez, D.F., Mancheno Falconi, M.G., Garcia, M.V.: Wearable sensors in industrial ergonomics: Enhancing safety and productivity in Industry 4.0. Sensors **25**(5), 1526 (2025). https://doi.org/10.3390/s25051526
7. Donisi, L., Cesarelli, G., Pisani, N., Ponsiglione, A.M., Ricciardi, C., Capodaglio, E.: Wearable sensors and artificial intelligence for physical ergonomics: A systematic review of literature. Diagnostics **12**(12), 3048 (2022). https://doi.org/10.3390/diagnostics12123048
8. Abdel-Salam, R., Mostafa, R., Hadhood, M.: Human activity recognition using wearable sensors: Review, challenges, evaluation benchmark. In: Deep Learning for Human Activity Recognition, pp. 1–15. Springer, Singapore (2021). https://doi.org/10.1007/978-981-16-0575-8_1
9. Arshad, M.H., Bilal, M., Gani, A.: Human activity recognition: Review, taxonomy and open challenges. Sensors **22**(17), 6463 (2022). https://doi.org/10.3390/s22176463

10. Kakhi, K., Jagatheesaperumal, S.K., Khosravi, A., Alizadehsani, R., Acharya, U.R.: Fatigue monitoring using wearables and AI: Trends, challenges, and future opportunities. Computers in Biology and Medicine **195**, 110461 (2025). https://doi.org/10.1016/j.compbiomed.2025.110461
11. Kulsoom, F., Narejo, S., Mehmood, Z., Chaudhry, H., Butt, A., Bashir, A.: A review of machine learning-based human activity recognition for diverse applications. Neural Computing and Applications **34** (2022). https://doi.org/10.1007/s00521-022-07665-9
12. Kumar, N.S., Deepika, G., Goutham, V., Buvaneswari, B., Reddy, R.V.K., Angadi, S., Dhanamjayulu, C., Chinthaginjala, R., Mohammad, F., Khan, B.: HARNet in deep learning approach-a systematic survey. Scientific Reports **14**(1), 8363 (2024). https://doi.org/10.1038/s41598-024-58074-y
13. Nguyen, H.D., Tran, K.P., Zeng, X., Koehl, L., Tartare, G.: Wearable sensor data based human activity recognition using machine learning: A new approach. arXiv preprint arXiv:1905.03809 (2019). https://arxiv.org/abs/1905.03809
14. Alikhani, M.: KAN-HAR: A human activity recognition based on Kolmogorov-Arnold network. arXiv preprint arXiv:2508.11186 (2025). https://arxiv.org/abs/2508.11186
15. Qureshi, T.S., Shahid, M.H., Farhan, A.A., Alamri, S.: A systematic literature review on human activity recognition using smart devices: Advances, challenges, and future directions. Artificial Intelligence Review **58**(9), 276 (2025). https://doi.org/10.1007/s10462-025-11275-x
16. Upadhyay, S., Dutta, D.: A review of deep learning models for real-time activity recognition in wearable sensor systems. African Journal of Biomedical Research **27** (2024). https://doi.org/10.53555/AJBR.v27i6S.8880
17. Hossen, M.A., Abas, P.E.: Machine learning for human activity recognition: State-of-the-art techniques and emerging trends. Journal of Imaging **11**(3), 91 (2025). https://doi.org/10.3390/jimaging11030091
18. Rehman, S., Ali, A., Khan, A., Okpala, C.: Human activity recognition: A comparative study of validation methods and impact of feature extraction in wearable sensors. Algorithms **17**, 556 (2024). https://doi.org/10.3390/a17120556
19. Gupta, S.: Deep learning based human activity recognition (HAR) using wearable sensor data. International Journal of Information Management Data Insights **1**(2), 100046 (2021). https://doi.org/10.1016/j.jjimei.2021.100046
20. Kumar, P., Suresh, S.: Deep-HAR: An ensemble deep learning model for recognizing the simple, complex, and heterogeneous human activities. Multimedia Tools and Applications (2023). https://doi.org/10.1007/s11042-023-14492-0
21. Ordóñez, F.J., Roggen, D.: Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors **16**(1), 115 (2016). https://doi.org/10.3390/s16010115
22. Mutegeki, R., Han, D.S.: A CNN-LSTM approach to human activity recognition. In: 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), pp. 362–366 (2020)
23. Dirgová Luptáková, I., Kubovčík, M., Pospíchal, J.: Wearable sensor-based human activity recognition with transformer model. Sensors **22**(5), 1911 (2022). https://doi.org/10.3390/s22051911
24. Bock, M., Moeller, M., Van Laerhoven, K.: Temporal action localization for inertial-based human activity recognition. arXiv preprint arXiv:2311.15831 (2024). https://arxiv.org/abs/2311.15831

25. Pang, Y.H., Ping, L.Y., Ling, G.F., et al.: Stacked deep analytic model for human activity recognition on a UCI HAR database. F1000Research **10**, 1046 (2022). https://doi.org/10.12688/f1000research.73174.3
26. Kann, B., Castellanos-Paez, S., Lalanda, P.: Evaluation of regularization-based continual learning approaches: Application to HAR. arXiv preprint arXiv:2304.13327 (2023). https://arxiv.org/abs/2304.13327
27. Ziaeefard, M., Bergevin, R.: Semantic human activity recognition: A literature review. Pattern Recognition **48**, 2329–2345 (2015). https://doi.org/10.1016/j.patcog.2015.03.006
28. Beddiar, R., Nini, B., Sabokrou, M., Hadid, A.: Vision-based human activity recognition: A survey. Multimedia Tools and Applications **79** (2020). https://doi.org/10.1007/s11042-020-09004-3
29. Dang, L.M., Min, K., Wang, H., Piran, M.J., Lee, C.H., Moon, H.: Sensor-based and vision-based human activity recognition: A comprehensive survey. Pattern Recognition **108**, 107561 (2020). https://doi.org/10.1016/j.patcog.2020.107561
30. Anguita, D., Ghio, A., Oneto, L., Parra, F., Reyes-Ortiz, J.: A public domain dataset for human activity recognition using smartphones. In: Proceedings of the conference (2013)
31. Reiss, A., Stricker, D.: Introducing a new benchmarked dataset for activity monitoring. In: Proceedings of the conference (2012). https://doi.org/10.1109/ISWC.2012.13
32. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. ACM SIGKDD Explorations Newsletter **12**(2), 74–82 (2011)
33. Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S.T., Tröster, G., del R. Millán, J., Roggen, D.: The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. Pattern Recognition Letters **34**(15), 2033–2042 (2013). https://doi.org/10.1016/j.patrec.2012.12.014
34. Wang, J., Chen, Y., Hao, S., Peng, X., Hu, L.: Deep learning for sensor-based activity recognition: A survey. Pattern Recognition Letters **119**, 3–11 (2019). https://doi.org/10.1016/j.patrec.2018.02.010

A Computer-Based Analysis of the Dravidian Language Family Using Congruent Sound Groups

Nikitha Reddy Baddam[1][0009−0007−6927−1131] and
Peter Z. Revesz[1,2][0000−0002−1145−1283]

[1] School of Computing,
University of Nebraska-Lincoln, Lincoln, NE 68508, USA
[2] Department of Classics and Religious Studies,
University of Nebraska-Lincoln, NE 68508, USA
✉ prevesz1@unl.edu

Abstract. A central topic in historical linguistics is the study of linguistic relationships among related languages by manually identifying cognate words based on regular sound changes. Since this manual approach is very tedious, this study proposes a computational approach to analyze structural similarities among a set of languages based on congruent sound groups, which capture common sound changes.
The Dravidian languages Kannada, Malayalam, Tamil and Telugu were used to test the feasibility of this computational approach. A database consisting of approximately 1200 words from these languages was compiled and converted into the International Phonetic Alphabet representation. Cognate relationships were identified using congruent sound groups. A distance matrix was constructed based on the number of shared cognates between each pair of languages, and the UPGMA clustering algorithm was used to generate a Dravidian language family tree.
The automatically-generated language family tree matched the traditional Dravidian language family tree, demonstrating the feasibility of the approach to the study of linguistic relationships.

1 Introduction

The classification of languages into families has long been an important topic in historical linguistics. Traditional linguistic methods identify relationships among languages by studying regular sound changes and identifying cognate words that share common origins. In recent years, computational approaches have increasingly been used to automate and extend such analyses, enabling large-scale comparisons across languages, such as the Indo-European Language Family [1] and Eurasian Superfamilies [8, 12]

One promising direction in computational linguistics is the use of data-driven methods to identify structural similarities among languages. These approaches combine linguistic theory with computational techniques such as phonological analysis, data processing, and clustering algorithms to reveal patterns that may not be immediately apparent through manual analysis.

A key concept in historical linguistics is that of cognates, which are words in different languages that derive from a common ancestral form. Usually, languages that have a recent common ancestor share more cognates than languages than languages that have only a distant common ancestor. Hence, the pairwise number of cognates among a set of languages enables to infer their likely evolutionary relationship. However, the identification of cognates is a complex process that needs to consider the regular sound changes that occur during language evolution.

To address this issue, Revesz [11] introduced the concept of *Congruent Sound Groups (CSGs)*, which represent groups of consonants that frequently change into one another across related languages. These groups capture common phonological transformations and therefore allow linguists to identify cognate relationships even when phonetic forms differ slightly.

In this study, we analyze lexical data from four Dravidian languages: Tamil, Malayalam, Kannada, and Telugu. Approximately 300 lexical items were collected and analyzed to identify cognate relationships using the CSG framework. The resulting cognate counts were used to construct a similarity matrix that measures the degree of shared lexical structure between language pairs.

Using this similarity matrix, the UPGMA clustering algorithm was applied to generate a phylogenetic tree that illustrates the structural relationships among the four languages. The goal of this work is to demonstrate how CSG-based cognate identification combined with computational clustering methods can reveal structural similarities within the Dravidian language family.

2 Background

2.1 International Phonetic Alphabet (IPA)

Languages are written using a wide variety of scripts, many of which do not directly reflect the actual pronunciation of words. To address that issue, linguists frequently represent words using the *International Phonetic Alphabet (IPA)* [5], which provides a standardized system for representing spoken words.

The IPA assigns a unique symbol to each phoneme, allowing linguists to represent pronunciation consistently across languages. Many IPA symbols correspond directly to letters in the Latin alphabet, while others represent sounds that are not present in English.

For example, the symbol /ŋ/ represents the final sound in the English word *sing*, while /θ/ represents the initial sound in *thin*, and /ʃ/ represents the initial sound in *she*. Using IPA notation allows phonetic comparisons to be made systematically across languages. This applies to both modern and ancient languages whose pronunciations can be reconstructed using some linguistic method.

In the work presented below, words from four different Dravidian languages were converted into an IPA-based phonetic representation for the purpose of a consistent phonological comparison.

2.2 Congruent Sound Groups

During the evolution of languages, phonemes often change in systematic ways. These changes are known as *regular sound changes* and play an important role in identifying cognate relationships among languages.

Revesz [11] introduced the concept of *Congruent Sound Groups (CSGs)* to represent sets of consonants that frequently transform into one another during language evolution. In 2025, Revesz and Siddha [12] extended the concept of CSGs in their study of Eurasian superfamilies to the following seven CSGs:

Congruent Sound Group ID	Set of Phonemes
CSG 1	b, p, m, n, ŋ
CSG 2	d, t, θ
CSG 3	f, v, w
CSG 4	g, k, h
CSG 5	l, r
CSG 6	s, z, ʃ, ʒ
CSG 7	j

We simplified the IPA notation of Dravidian phonetics that contains additional details regarding stress and length, and we also simplified the hard-to-print IPS signs ʃ and ŋ as shown in the table below:

Original Symbol	Normalized Form
ɳ	n
ñ	n
ʈ	t
ɖ	d
ɭ	l
ʔ	v
s‘	s
ʃ	s
ŋ	ng

The above transformations simplify phonetic comparison while preserving the essential phonological structure of the Dravidian words.

The congruent sound groups capture many phonological transformations that are commonly observed in the evolution of Eurasian languages. For example, /b/ and /p/ form a voiced–voiceless pair, while /l/ and /r/ frequently interchange in many languages.

For example, the words *nan* (Tamil), *nan* (Malayalam), *nanu* (Kannada), and *nenu* (Telugu), all meaning the pronoun 'I,' exhibit similar phonological structures, including the shared word initial /n/ and the shared second consonant /n/ that belong to the same Congruent Sound Group 1, which supports these words' identification as cognates.

Note that the variation in the vowels is greater than the variation in the consonants in the case of these words. The greater stability of consonants is a general observation in the study of linguistic evolution. Hence, we will concentrate on the automatic detection of consonantal similarities in evaluating whether a pair of words are cognates. In particular, we will consider two words cognates if their first consonants, which are their word initial consonants unless they start with some vowels, belong to the same congruent sound group, and simultaneously their second consonants also belong to the same congruent sound group, and they have the same or closely similar meaning. These criteria clearly apply to the four Dravidian words mentioned above because their word initial consonants belong to CSG 1, their second consonants belong to CSG 1, and they all mean the pronoun 'I.'

2.3 Data Sources

The dataset used in this study consists of approximately 300 lexical items collected from the following four Dravidian languages: Kannada, Malayalam, Tamil, and Telugu. Our dataset contains the Swadesh list of Dravidian words [15] that are listed from row 'I' to row 'name' in the first 207 rows of Table 1 and 93 additional groups of cognate Dravidian words that were collected from other publicly available resources, including the Telugu dictionary of Brown [2], the Malayalam dictionary of Gundert [4], the Kannada dictionary of Kittel [6], the Tamil dictionary of the University of Madras [16], and the Dravidian etymological dictionary of Burrow [3] and Krishnamurti [7]. Each word was converted into a cleaned phonetic representation to ensure consistent comparison across languages. We concentrated on words that are likely cognates instead of borrowings according to the etymological dictionaries.

In Table 1, words identified as cognates are marked using two typographic styles to distinguish between two different cognate groups within the same row. **Bold** text indicates words belonging to the first cognate group, and *italic* text indicates words belonging to a second, separate cognate group. A word may appear in bold in some languages and italic in others within the same row if those languages participate in different cognate relationships for that particular item. Unmarked cells indicate words with no cognate relationship identified in that row.

Table 1: A Dravidian words database with 300 rows. Each row presents a set of words with the same meaning in the following four Dravidian languages: Tamil, Malayalam, Kannada, and Telugu. Each Dravidian word is represented in an IPA form. In each row the words that are in bold face are cognates, and those that are italic are also cognates.

English	Tamil	Malayalam	Kannada	Telugu
I	**nan**	**nan**	**nanu**	**nenu**
you (singular)	**ni**	**ni**	**ninu**	**nuvvu**

Continued on next page

Table 1 – *continued from previous page*

English	Tamil	Malayalam	Kannada	Telugu
he	**avan**	**avan**	**avanu**	atadu
we	**nam**	**nammal**	**navu**	manamu
you (plural)	**ningkal**	**ningngal**	nivu	miru
they	**avar**	**avar**	**avaru**	vallu
this	**itu**	**itu**	**idu**	**idi**
that	**atu**	**atu**	**adu**	**adi**
here	ingke	ivite	illi	ikkada
there	angke	avite	alli	akkada
who	**yar**	aru	**yaru**	evaru
what	**enna**	entu	**enu**	emiti
where	engke	evite	elli	ekkada
when	**eppo**	**eppol**	yavaga	**eppudu**
how	eppati	engngane	hege	ela
not	**illai**	**alla**	**alla**	kadu
all	**anaittu**	*ellam*	*ella*	**andaru**
many	niraiya	kure	hala	tsala
some	**tsila**	**tsila**	**kela**	kondaru
few	kontsam	kuratstsu	svalpa	konni
other	marra	**vere**	**bere**	**vere**
one	**onru**	**onnu**	**ondu**	**okati**
two	**irantu**	**rantu**	**eradu**	**rendu**
three	**munru**	**munnu**	**muru**	**mudu**
four	**nanku**	**nalu**	**nalku**	**nalugu**
five	**aintu**	**antsu**	**aidu**	**aidu**
big	periya	valiya	**dodda**	**pedda**
long	**nilamana**	**nilamulla**	nidu	podavaina
wide	**akalamana**	vitiyulla	**agala**	vedalpaina
thick	tatitta	kattiyulla	dappa	mandamaina
heavy	**kanamana**	**kanatta**	tuka	baruvaina
small	**tsinna**	**tseriya**	sanna	**tsinna**
short	**kuru**	**kuriya**	gidda	potti
narrow	kurukiya	itungngiya	sapura	irukaina
thin	**melliya**	**melinna**	telu	sannani
woman	**pen**	**pennu**	henggasu	adadi
man (adult male)	**an**	**aan**	gandasu	magavadu
man (human being)	**manitan**	**manusjan**	**manusya**	**manisi**
child	kuantai	kutti	magu	pilla
wife	manaivi	bharya	hendati	pellam
husband	kanavan	bharttavu	ganda	mogudu
mother	**amma**	**ama**	**amma**	**amma**
father	**appa**	**atsan**	**appa**	nanna
animal	vilangku	mrgam	jantu	jantu
fish	**min**	**min**	**minu**	tsepa
bird	paravai	kili	hakki	pitta
dog	**nay**	patti	**nayi**	kukka
louse	**pen**	**pen**	henu	**penu**

Continued on next page

Table 1 – *continued from previous page*

English	Tamil	Malayalam	Kannada	Telugu
snake	**pampu**	**pampu**	havu	**pamu**
worm	**puu**	**puu**	hula	purugu
tree	**maram**	**maram**	**mara**	tsettu
forest	**katu**	**katu**	**kadu**	adavi
stick	kutstsi	vati	kolu	karra
fruit	**pazham**	**pazham**	hannu	pandu
seed	**vitai**	**vittu**	bittane	vittanam
leaf	**ilai**	**ila**	**ele**	aku
root	**ver**	**veru**	**beru**	**veru**
bark (of a tree)	pattai	tol	togate	beradu
flower	**pu**	**puvu**	**huvu**	**puvvu**
grass	**pul**	**pullu**	**hullu**	gaddi
rope	kajiru	kajar	haga	tadu
skin	**tol**	**toli**	tsarma	**tolu**
meat	iraitstsi	**mamsam**	**mamsa**	**mamsam**
blood	**rattam**	tsora	rakta	**raktamu**
bone	**elumpu**	**ellu**	**elubu**	emuka
fat (noun)	**kozhuppu**	**kozhuppu**	*kobbu*	*kovvu*
egg	**muttai**	**mutta**	**motte**	guddu
horn	**kompu**	**kompu**	**kombu**	**kommu**
tail	**val**	**val**	**bala**	toka
feather	iraku	tuval	gari	ika
hair	**muti**	**muti**	kudalu	ventrukalu
head	**talai**	**tala**	**tale**	**tala**
ear	katu	**tsevi**	**kivi**	**tsevi**
eye	**kan**	**kann**	**kannu**	**kannu**
nose	**mukku**	**mukku**	**mugu**	**mukku**
mouth	**vayi**	**vaya**	**bayi**	noru
tooth	**pal**	**pallu**	hallu	pannu
tongue (organ)	nakku	nav	**nalige**	**naluka**
fingernail	**nakam**	**nakham**	uguru	goru
foot	**ati**	*padam*	**adi**	*padamu*
leg	**kal**	**kalu**	**kalu**	**kalu**
knee	**muttu**	**muttu**	mandi	mokalu
hand	**kai**	**kai**	**kai**	tseyyi
wing	irakkai	tsiraku	**rekke**	**rekka**
belly	tonti	vayaru	hotte	potta
guts	**kutal**	**kutal**	ketstsu	pegulu
neck	**kazhuttu**	**kazhuttu**	**kattu**	meda
back	**mutuku**	**mutuku**	bennu	vipu
breast	marpu	mula	mole	tsannu
heart	**itayam**	**hrdayam**	**hrdaya**	gunde
liver	iral	karal	ili	kaleyam
to drink	**kuti**	**kutikkuka**	**kudi**	tagu
to eat	tsappitu	kazhikkuka	**tinnu**	**tinu**
to bite	**kati**	**katikkuka**	**kadi**	koruku

Continued on next page

Table 1 – *continued from previous page*

English	Tamil	Malayalam	Kannada	Telugu
to suck	urintsu	tsappuka	**tsipu**	**tsiku**
to spit	**tuppu**	**tuppuka**	ugulu	ummu
to vomit	**kakku**	tsharddikkuka	karu	**kakku**
to blow	**utu**	**utuka**	**udu**	**udu**
to breathe	mutstsuvitu	shvasikkuka	usiru adu	upiri piltsu
to laugh	tsiri	tsirikkuka	**nagu**	**navvu**
to see	par	kanuka	nodu	tsutsu
to hear	**kel**	**kelkkuka**	**kelu**	vinu
to know	**ari**	**ariyuka**	tili	telusukonu
to think	ninai	tsintikkuka	yotsisu	alotsintsu
to smell	mo	manakkuka	musu	vasana tsutsu
to fear	antsu	petikkuka	**hedaru**	**beduru**
to sleep	tungku	urangnguka	malagu	kunuku
to live	vazh	jivikkuka	**balu**	**batuku**
to die	**sagu**	*tsakuka*	**sayi**	*tsavu*
to kill	**kol**	**kolluka**	**kollu**	tsampu
to fight	**poratu**	**poratuka**	horadu	**poradu**
to hunt	**vettaiyatu**	**vettayatuka**	**beteyadu**	**vetadu**
to hit	ati	itikkuka	hode	kottu
to cut	vettu	murikkuka	kattarisu	koyu
to split	piri	pilarkkuka	silu	tsiltsu
to stab	**kuttu**	**kuttuka**	iri	podutsu
to scratch	**kiru**	mantuka	kere	**giru**
to dig	**tontu**	kuzhikkuka	**todu**	tavvu
to swim	**nintu**	**nintuka**	*iju*	*idu*
to fly	**para**	**parakkuka**	haru	eguru
to walk	**nata**	**natakkuka**	nade	nadutsu
to come	**va**	varuka	**ba**	vatstsu
to lie (as in a bed)	**patu**	kitakkuka	**padu**	**padukonu**
to sit	utkar	irikkuka	kuru	kurtsonu
to stand	**nil**	**nilkkuka**	**nillu**	**nilutsonu**
to turn (intransitive)	**tirumpu**	**tiriyuka**	**tirugu**	tippu
to fall	**vizhu**	**vizhuka**	**bilu**	padu
to give	ta	nalkuka	kodu	itstsu
to hold	**piti**	**pitikkuka**	**hidi**	pattukonu
to squeeze	**pizhi**	**pizhiyuka**	hitsuku	pisuku
to rub	tey	uraykkuka	ujju	ruddu
to wash	**kazhuvu**	**kazhukuka**	tole	kadugu
to wipe	**tutai**	**tutaykkuka**	oresu	tudutsu
to pull	**izhu**	valikkuka	**eli**	lagu
to push	**tallu**	**talluka**	nuku	nettu
to throw	**eri**	**eriyuka**	esi	visuru
to tie	**kattu**	kettuka	**kattu**	**kattu**
to sew	**tai**	**taykkuka**	holi	kuttu
to count	**ennu**	**ennuka**	**enisu**	lekkintsu
to say	sol	parayuka	helu	tseppu

Continued on next page

Table 1 – *continued from previous page*

English	Tamil	Malayalam	Kannada	Telugu
to sing	**patu**	**patuka**	*hadu*	*padu*
to play	vilaiyatu	kalikkuka	**adu**	**adu**
to float	mita	pongnguka	**telu**	**telu**
to flow	**ozhuku**	**ozhukuka**	hari	paru
to freeze	urai	tanutt urayuka	**heppugattu**	**gaddakattu**
to swell	vingku	virkkuka	udu	vatsu
sun	**tsuriyan**	**surjan**	**surya**	**suryudu**
moon	**nila**	**nilav**	*tsandra*	*tsandrudu*
star	**nattsattiram**	**naksatram**	**naksatra**	**naksatram**
water	**nir**	velam	**niru**	**niru**
rain	**mazhai**	**mazha**	malesuri	kuriyu
river	aru	puzha	**nadi**	**nadi**
lake	eri	tatakam	kere	sarassu
sea	**katal**	**katal**	*samudra*	*samudram*
salt	**uppu**	**uppu**	**uppu**	**uppu**
stone	**kal**	**kal**	**kallu**	raji
sand	man	manal	maralu	isuka
dust	puzhuti	poti	**dumbu**	**dummu**
earth	pumi	**mann**	nela	**mannu**
cloud	**mekam**	**megham**	moda	mabbu
fog	pani	mutalmannu	manju	pogamantsu
sky	van	**akasham**	**akasha**	ninggi
wind	**karru**	**karru**	*gali*	*gali*
snow	pani	**mannu**	hima	**mantsu**
ice	panikkatti	aisu	manjugadde	gaddamantsu
smoke	**pukai**	**puka**	**hoge**	**poga**
fire	neruppu	ti	bengki	manta
ash	tsampal	tsaram	**budi**	**budida**
to burn	eri	kattikkuka	sudu	kalu
road	tsalai	rodu	hadi	bata
mountain	**malai**	**mala**	parvata	konda
red	tsivappu	tsuvapu	kempu	erra
green	**patstsai**	**patstsa**	hasiru	**patstsa**
yellow	mantsal	manna	haladi	pasupu
white	**vellai**	**vella**	**bili**	**tella**
black	**karuppu**	**karuppu**	kari	nalla
night	iravu	**ratri**	**ratri**	reyi
day	**pakal**	divasam	**hagalu**	**pagalu**
year	antu	varsam	**edu**	**yedu**
warm	**tsutu**	**tsutu**	betstsa	vetstsani
cold	**kulirtstsi**	tanuppu	**kuliru**	tsallani
full	muzhu	niranna	tumbu	nindu
new	**putu**	**putiya**	hosa	kotta
old	**pazha**	**pazhaya**	hale	**pata**
good	**nal**	**nalla**	olle	mantsi
bad	**ketta**	mosham	**ketta**	tsedda

Continued on next page

Table 1 – *continued from previous page*

English	Tamil	Malayalam	Kannada	Telugu
rotten	**azhukiya**	**azhukiya**	koleta	kullina
dirty	**azhukkana**	**azhukkaya**	kolaku	murikaina
straight	**ner**	**nere**	**nera**	**neru**
round	urunta	vattam	**gundu**	**gundrani**
sharp (as a knife)	kurmai	murtstsayulla	harita	padunaina
dull (as a knife)	mazhungkiya	murtstsa kuranna	**monda**	**mondi**
smooth	vazhuvazhuppu	minusamarnna	**nunnage**	**nunnani**
wet	iramana	nananna	odde	tadi
dry	ularnta	varanta	ona	podi
correct	**tsari**	**shari**	**sari**	**saraina**
near	arukil	atuttu	hattira	daggara
far	tolaivu	akale	**dura**	**duram**
right	**valatu**	**valatu**	bala	kudi
left	**itatu**	**itatu**	*eda*	*edama*
at	**-il**	**-il**	-alli	-vadda
in	ule	ulil	olage	-lo
with	-utan	kute	jote	-to
and	**-um**	**-um**	mattu	mariyu
if	enral	-engkil	ondu vele	-ite
because	enenral	karanam	yakendare	-valla
name	peyar	**peru**	hesaru	**peru**
actor	nadigan	**nadan**	**nada**	**nadudu**
ankle	kanukkal	kanangkal	kanakalu	madama
ant	**erumpu**	**urumpu**	iruve	tsima
ball	**pandu**	**pant**	tsendu	banti
basket	kutai	kotta	butti	gampa
bed	padukkai	kidakka	hasige	mantsam
blanket	porvai	putapu	hasige	dupati
book	puttagam	**pustakam**	**pustaka**	**pustakam**
border	ellai	atirtti	gadi	sarihaddu
branch	kilai	**kombu**	**kombe**	**koma**
bridge	palam	palam	setu	vantena
broom	tumpu	tul	dzhadu	uda
carpenter	**tatsan**	**tatsan**	badagi	vadranggi
chair	nazhkal irukkai	kasera	**kurci**	**kurci**
chef	samajalkarar	patsakakaran	adugegara	vantavadu
chin	**tatai**	**tati**	**tadi**	monadavada
clay	**man**	**mann**	**mannu**	matti
clock	**kadigaram**	**ghadikaram**	**gadijara**	**gadijaram**
Coconut	**tenggaj**	**tenga**	**tengginakji**	kobarikaja
college	kallori	koledzh	kaledzhu	kalasala
court	nidimanram	kotatimuri	najalaja	najasthanam
crow	**kagam**	**kaka**	**kage**	**kaki**
Cucumber	**velari**	**velari**	sautekaji	dosakaja
dance	nadanam	**nrttam**	**nrtja**	**nrtjam**
daughter	**magal**	**magal**	**magalu**	kumarte

Continued on next page

Table 1 – *continued from previous page*

English	Tamil	Malayalam	Kannada	Telugu
death	**maranam**	**maranang**	**marana**	mrti
defeat	**tolvi**	**tolvi**	solu	otami
doctor	maruttuvar	doktar	vaidja	vidyudu
donkey	kazhutai	kazhuta	katte	gadida
door	kadavu	vatil	bagilu	talupu
drum	murasu	tsenda	**dollu**	**dolu**
elephant	janai	**ana**	**ane**	enugu
empty	kali	sunjam	**khali**	**khali**
fast	**vegamaga**	**vegatil**	**vegavagi**	**vegangga**
fence	**veli**	**veli**	**beli**	kantse
field	**vaijal**	**vajal**	hola	polam
firewood	**viragu**	**virak**	kattige	kattelu
flight	**vimanam**	**vimanang**	**vimana**	**vimananm**
fort	**kotai**	**kota**	**kote**	**kota**
fox	**nari**	**nari**	**nari**	nakka
goat	**adu**	**adi**	**adu**	meka
grain	**arisi**	**ari**	akki	bijam
ground	man	**mann**	**nela**	**nela**
honey	**ten**	**ten**	**dzhenu**	**tene**
horse	**kudirai**	**kudira**	**kudure**	guram
island	**tivu**	**dvip**	**dvipa**	**divi**
jasmine	**maligai**	**mula**	**malige**	**male**
jump	kudippu	kutigguka	dzhigidu	kadilu
key	savi	takol	kili	talamtsevi
knife	**kati**	**kati**	tsaku	**kati**
ladder	**eni**	**eni**	mettilu	mettu
language	mozhi	**bhasa**	**bhase**	**bhasa**
late	**tamadamaga**	vaikate	**tadavagi**	alasjangga
lion	singgam	**simham**	**simha**	**simham**
Lock	**putu**	**putu**	biga	talam
milk	**pal**	**pal**	**halu**	**palu**
mirror	**kanati**	**kanati**	kanadaka	adam
muscles	tasaigal	pesikal	snajugalu	kandaralu
music	isai	**sanggitam**	**sanggita**	**sanggitam**
nest	**kutu**	**kutu**	*gudu*	*gudu*
ocean	**kadal**	**kadal**	*samudra*	*samudram*
oil	**enej**	**ena**	**ene**	nune
parrot	kili	tatta	gili	tsiluka
pillow	talaijanai	**talajana**	**talejana**	dindu
plant	**sedi**	**tsedi**	sasi	moka
Pot	panai	patram	handi	gine
potato	**urulaikizhanggu**	**urulakizhangu**	alugade	banggaladumpa
rainbow	vanavil	mazhavil	kamanabillu	vanavillu
ready	**tajaraga**	**tajjaraji**	sidha	sidhangga
roof	kurai	melkura	meltsavani	paikapu
scissors	**kattari**	**kattiri**	**kattari**	**kattera**

Continued on next page

Table 1 – *continued from previous page*

English	Tamil	Malayalam	Kannada	Telugu
search	tetu	tiraiuka	huduku	vetakadang
shirt	satai	sart	anggi	tsoka
shoulder	**tol**	**tol**	bhudzha	bhudzham
son	**magan**	**magan**	**maga**	kumarudu
song	padal	**patu**	hadu	**pata**
sound	oli	**sabdam**	**sabda**	**sabdam**
spinach	kirai	tsira	sopu	palakura
squirrel	**anil**	**anali**	**alilu**	uduta
strength	valimai	**sakti**	**sakti**	**sakti**
sudden	titirendru	apratiksitamaji	atsanaka	akasmatuga
sugar	sarkarai	pandzhasara	**sakare**	**tsakera**
talent	tizhamai	kazhiv	**pratibhe**	**pratibha**
tamarind	**puli**	**puli**	hunase	tsintapandu
thief	tirudan	mostav	kalla	dongga
thread	**nul**	**nul**	*dara*	*daram*
thunder	iti	itimuzhakam	gudugu	urumu
tiger	**puli**	**puli**	**huli**	**puli**
turmeric	mandzhal	mannal	arisina	pasupu
victory	veti	**vidzhajam**	**vidzhaja**	**vidzhajam**
well (Waterwell)	**kinru**	**kinar**	*bavi*	*bavi*
window	**dzhanal**	**dzhanal**	*kitaki*	*kitiki*
wrist	**manikkattu**	**manikkattu**	**manikattu**	**manikattu**

3 Experimental Method and Results

3.1 Experimental Method

The experimental procedure consisted of the following six steps:

1. Lexical items were collected from the four Dravidian languages.
2. The words were converted into IPA phonetic representations.
3. Cognate relationships were identified using Congruent Sound Group similarity rules as described above and then checked using the etymological dictionaries whenever possible.
4. The number of shared cognates between each pair of languages was counted.
5. A similarity matrix was constructed based on these pairwise similarity counts.
6. The UPGMA clustering algorithm was applied to the similarity matrix to generate a language family tree of Dravidian languages.

Cognate relationships were identified manually by examining whether phonetic forms could be transformed into one another through substitutions within the same Congruent Sound Groups. This process was found to be robust and matched well the information found in the Dravidian etymological dictionaries.

To illustrate the structure of the lexical database used in our computer experiment, Table 2 presents a detailed analysis of seven sample rows from the Dravidian words database shown above in Table 1.

These seven examples demonstrate how congruent sound groups can be effectively used to identify the important phonological similarities that occur among cognate set of words across the languages. The first five data rows use two congruent sound groups to test whether the words in the row are cognate or not. If both congruent sound groups match for two words, then the words are considered cognate and are indicated using bold face.

The sign - indicates that were was no second consonant in the compared cognate words. This situation occurs in the last two rows. In these cases we have to rely only on the first consonant in the set of words in the given row.

Table 2. Seven example sets of cognate Dravidian words with an explanation of the their phonetic similarities based on congruent sound groups.

Tamil	Malayalam	Kannada	Telugu	1st CSG	2nd CSG
nan	**nan**	**nanu**	**nenu**	1	1
avan	**avan**	**avanu**	atadu	3	1
nam	**nammal**	**namu**	**manamu**	1	1
miru	ningal	nivu	**miru**	1	5
vall	**avar**	**avaru**	**vallu**	3	5
itu	**itu**	**itu**	**idi**	2	-
atu	**atu**	**atu**	**adi**	2	-

3.2 Experimental Results

Table 3 shows the similarity matrix obtained from the cognate analysis. The shared cognates are also indicated by common bold or italic notations between the cognate pairs.

Table 3. The cognate similarity matrix.

Language	Tamil	Malayalam	Kannada	Telugu
Tamil	0	157	95	57
Malayalam	157	0	97	69
Kannada	95	97	0	94
Telugu	57	69	94	0

The values in the matrix represent the number of shared cognates between each pair of languages. Figure 1 visualizes the similarity matrix as a heat map. The values along the main diagonal all have a value of zero by default. The visualization shows that the highest similarity occurred between Malayalam and Tamil, while the lowest similarity occurred between Tamil and Telugu.

To further analyze the relationships among the languages, the similarity matrix was converted into a distance matrix that is shown in Table 4. Each distance

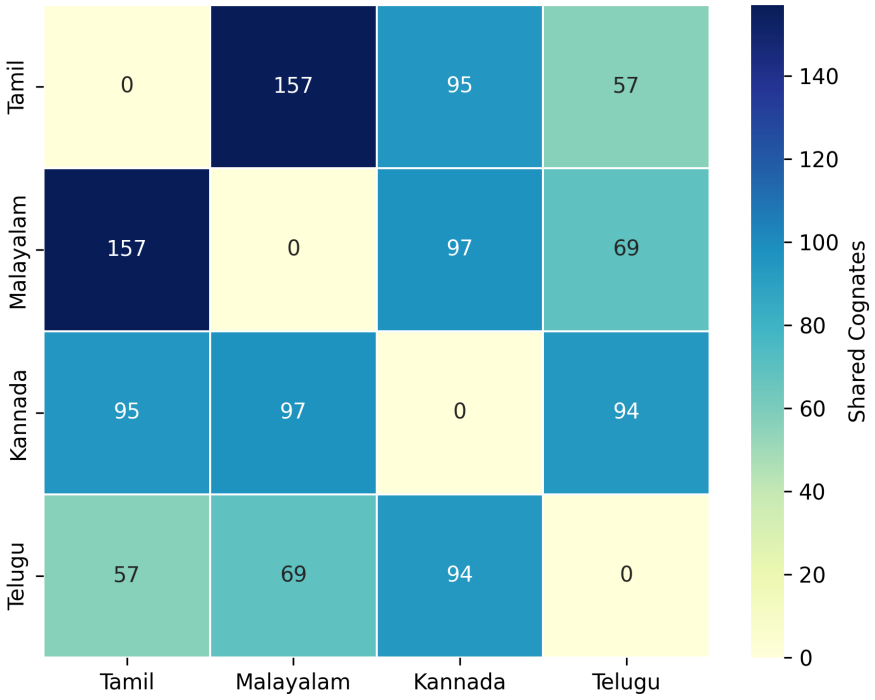


Fig. 1. Heatmap visualization of the cognate similarity matrix for the four Dravidian languages based on the words in Table 1. The maximum value is 300 between each pair of languages. The main diagonal entries are set to zero by default.

value is 300 (the number of rows of possible cognates considered) minus the similarity value except along the main diagonal, where the entries were again set to zero as a default.

Table 4. The cognate distance matrix.

Language	Tamil	Malayalam	Kannada	Telugu
Tamil	0	143	205	243
Malayalam	143	0	203	231
Kannada	205	203	0	206
Telugu	243	231	206	0

The cognate distance matrix is used as an input to the UPGMA clustering algorithm. The resulting hierarchical clustering tree generated by the UPGMA algorithm is shown in Figure 2.

4 Discussion of the Results

We first describe the traditional view about the evolution of the Dravidian language family in Section 4.1, and then we compare those with our results in Section 4.2.

4.1 Traditional Views about the Dravidian Language Family

The clustering result produced by the UPGMA algorithm can be compared with the widely accepted linguistic classification of the Dravidian language family. According to historical linguistic studies, the Dravidian languages are divided into several branches that originated from Proto-Dravidian. Among the four languages considered in this study, Tamil, Malayalam, and Kannada belong to the South Dravidian branch, while Telugu belongs to the South-Central Dravidian branch [7].

Figure 3 shows a simplified representation of the traditional Dravidian language family tree for the four languages considered in this study. In this classification, Tamil and Malayalam form the closest pair, while Kannada joins this pair within the South Dravidian branch. Telugu forms a separate branch within the Dravidian language family.

Within the South Dravidian branch, Tamil and Malayalam are known to be especially closely related. Malayalam historically developed from Middle Tamil around the 9th century CE and therefore shares many phonological and lexical similarities with Tamil. Kannada also belongs to the South Dravidian branch but represents a somewhat earlier divergence within this branch. Telugu, in contrast, belongs to the South-Central Dravidian branch and diverged much earlier from the languages of the South Dravidian branch.

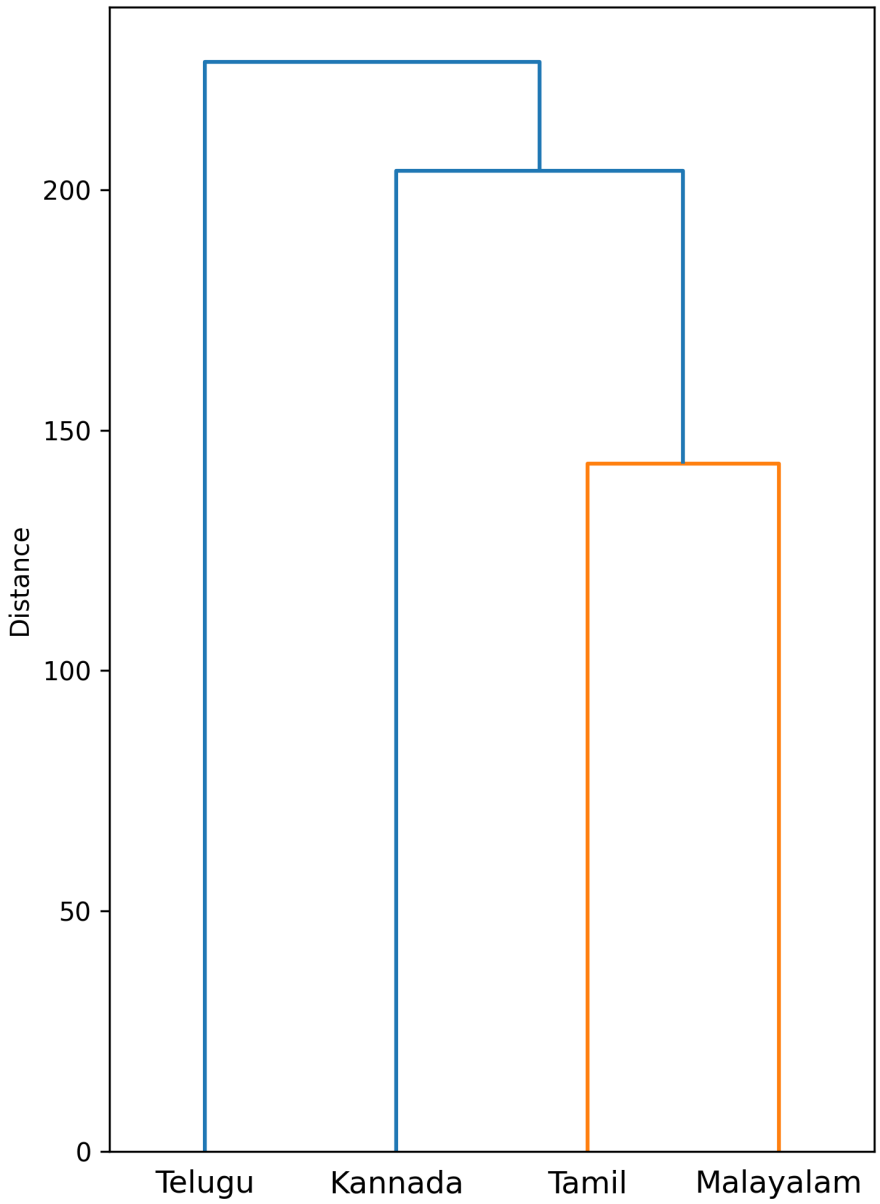


Fig. 2. UPGMA phylogenetic tree of the four Dravidian languages.

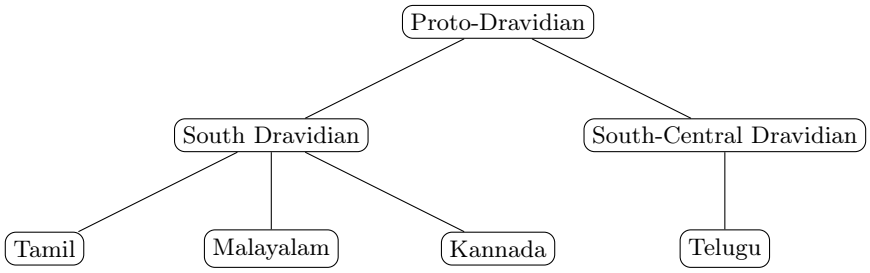


Fig. 3. A simplified traditional classification of the Dravidian language family showing the hypothetical origin language Proto-Dravidian on the top and the currently spoken Dravidian languages Kannada, Malayalam, Tamil and Telugu as leaves in the bottom of the classification tree.

4.2 Comparison with our Results

The UPGMA tree generated in our study shows a similar structure to the classification tree in Figure 3. In the UPGMA tree, Malayalam and Tamil form the closest pair, followed by Kannada, while Telugu appears as the most distant language among the four Dravidian languages. This result closely resembles the traditional linguistic classification shown in Figure **??**. In both cases, Malayalam and Tamil appear as the most closely related languages, reflecting their shared historical and structural development within the South Dravidian subgroup.

Although the overall structure of the UPGMA generated tree aligns well with the traditional linguistic classification tree, the two approaches rely on different types of evidence. The UPGMA tree is based solely on the counts of shared cognates within the Dravidian lexical database used in this study, while traditional linguistic classifications rely on a broader range of historical linguistic evidence, including systematic sound correspondences, grammatical structures, and historical reconstruction of protolanguages.

Several factors may contribute to differences between the UPGMA-generated tree and the traditional classification tree. First, lexical borrowing between neighboring languages may influence cognate counts and therefore affect similarity measurements. Second, phonological simplifications applied during data cleaning may obscure certain historical sound correspondences. Finally, the database used in this study consists of only about $300 \times 4 = 1200$ words, which likely represent only a fraction of the entire set of cognate words. Hence, expanding the database and incorporating additional linguistic features could further improve the accuracy of the computational analysis of the structure of the Dravidian language family.

5 Conclusions and Further Work

This study demonstrates how computational methods can be used to analyze structural similarities among languages. Using a database of approximately 1200

words from four Dravidian languages, we applied a Congruent Sound Group-based automated cognate identification and language family tree generation method to study the evolution of the Dravidian language family.

Our computational results indicated that Malayalam and Tamil form the closest pair among the four analyzed Dravidian languages, followed by Kannada, while Telugu appears as the most distant language in this group. Satisfyingly, these findings are consistent with the established classification of the Dravidian language family, which suggests that the method can be further extended. Future work could extend this approach by incorporating additional Dravidian languages, possibly including the extinct language of the Indus Valley Civilization [14], expanding the lexical database, and considering alternative clustering techniques such as those that are based on Common Mutation Matrices [9]. In addition, other language families such as the Uralic language [10, 13] family could be also studied in a similar way.

References

1. Bouckaert, R., Lemey, P., Dunn, M., Greenhill, S.J., Alekseyenko, A.V., Drummond, A.J., Gray, R.D., Suchard, M.A. and Atkinson, Q.D.: Mapping the origins and expansion of the Indo-European language family. Science 337 (6097), 957-960, (2012). doi: 10.1126/science.1219669
2. Brown, C.P.: A Telugu–English Dictionary. Oxford University Press (1903)
3. Burrow, T., Emeneau, M.B.: A Dravidian Etymological Dictionary, 2nd edn. Clarendon Press, Oxford (1984)
4. Gundert, H.: A Malayalam–English Dictionary. Mangalore Basel Mission Press (1872)
5. International Phonetic Association: Handbook of the International Phonetic Alphabet. Cambridge University Press (1999)
6. Kittel, F.: A Kannada–English Dictionary. Basel Mission Press (1894)
7. Krishnamurti, B.: The Dravidian Languages. Cambridge University Press (2003)
8. Pagel, M., Atkinson, Q.D., S. Calude, A. and Meade, A.: Ultraconserved words point to deep language ancestry across Eurasia. Proceedings of the National Academy of Sciences 110(21), 8471-8476, (2013). doi: 10.1073/pnas.1218726110
9. Revesz, P.Z.: An algorithm for constructing hypothetical evolutionary trees using common mutations similarity matrices, ACM International Conference on Bioinformatics and Computational Biology, ACM Press, pp. 730-732, (2013). doi: 10.1145/2506583.2512391
10. Revesz, P.Z.: Establishing the West-Ugric language family with Minoan, Hattic and Hungarian by a decipherment of Linear A, WSEAS Transactions on Information Science and Applications, 14, 306-335, (2017)
11. Revesz, P.Z.: Spatio-temporal data mining of major European river and mountain names reveals their Near Eastern and African origins. In: Proceedings of the 22nd European Conference on Advances in Databases and Information Systems (ADBIS), pp. 20–32 (2018). doi: 10.1007/978-3-319-98398-1_2
12. Revesz, P.Z., Siddha, M.: Data Mining for Language Superfamilies Using Congruent Sound Groups. In: Bergami, G., et al. (eds.) Database Engineered Applications, Lecture Notes in Computer Science, vol. 15928. Springer Nature, Cambridge, UK (2025). doi: 10.1007/978-3-032-06744-9_17

13. Revesz, P.Z.: Data Mining Archaeogenetic and Linguistic Data Gives an Improved Chronology of the Uralic Language Family, Information, 16 (10), no. 930, (2025). doi: 10.3390/info16110930
14. Tamkiya, H, Agrawal, G., Debnath, C, Revesz, P.Z.: Automated identification of allographs among the Indus Valley Script signs, In: G. Bergami et al. (eds.), Proc. 29th Int. Database Engineered Applications Symposium, Springer Nature LNCS 15928, pp. 252-261, Newcastle, UK, (2025). doi: 10.1007/978-3-032-06744-9_19
15. Wikipedia: Dravidian Swadesh lists, Available at: https://en.wiktionary.org/wiki/Appendix:Dravidian_Swadesh_lists (2026)
16. University of Madras: Tamil Lexicon. University of Madras Press (1924–1936)

Graph-Enhanced Probabilistic Spatio-Temporal Modeling for Flight Departure Delay Prediction

Mary Dufie Afrane[1 [0009−0001−8057−7314]], Yao Xu[1 [0000−0001−9356−9586]✉], and Lixin Li[1 [0000−0003−4011−0784]]

Georgia Southern University, Statesboro GA, USA
ma20542@georgiasouthern.edu, ✉ yxu@georgiasouthern.edu, lli@georgiasouthern.edu

Abstract. Flight delays remain a major challenge in modern air transportation systems. This paper proposes a Graph-enhanced Probabilistic Spatio-Temporal (GPST) framework for flight departure delay prediction that integrates operational features, temporal delay patterns, and graph-based airport embeddings capturing airport connectivity. Starting from over eleven million U.S. domestic flight records, preprocessing and filtering produced a final modeling dataset of 2.88 million Delta Air Lines flights. The model uses a probabilistic two-stage strategy that first estimates the probability of delays exceeding 15 minutes and then predicts delay exceedance magnitude using gradient-boosted tabular learning and a spatio-temporal neural network. Evaluation using rolling time-series cross-validation achieves an average MAE of about 10 minutes and 85.9% accuracy within a 15-minute tolerance window.

Keywords: Flight Delay Prediction · Air Transportation Networks · Spatio-Temporal Neural Network · Graph Neural Networks · Probabilistic Modeling · Delay Propagation · Aviation Data Analytics.

1 Introduction

Air transportation plays a central role in modern global mobility, supporting economic activity, tourism, and international connectivity. As airline networks continue to expand, the efficient management of flight operations has become increasingly important. One of the most persistent challenges faced by the aviation industry is flight delay, which can disrupt airline schedules, increase operational costs, and negatively affect passenger experience. Delays often propagate through interconnected airport networks, causing cascading disruptions that extend far beyond the original source of the delay. As a result, accurately predicting flight delays has become an important research problem for both airline operations and transportation planning.

The growing availability of large-scale aviation datasets and advances in machine learning have significantly improved research on flight delay prediction. Recent studies employ a range of approaches, including traditional machine

learning models, deep neural networks, and graph-based spatio-temporal methods, to capture operational factors influencing delays [21]. Tree-based models such as Random Forest and Gradient Boosting perform well with engineered operational features, while deep learning models, including recurrent neural networks and temporal convolution networks, capture sequential patterns in flight operations. More recently, graph-based approaches represent airports as nodes within a network to model delay propagation across flight routes, reflecting the interconnected nature of the air transportation system, where disruptions at one airport can influence operations throughout the network.

Despite these advances, several challenges remain in developing reliable delay prediction systems. Flight delays arise from complex interactions among factors such as weather conditions, airport congestion, aircraft rotations, and air traffic control constraints, requiring models that integrate heterogeneous data sources and capture both spatial and temporal dependencies. In addition, delay distributions are highly skewed, with most flights experiencing small delays while a relatively small number experience extreme disruptions, making accurate regression modeling difficult. Many existing studies focus either on binary delay classification or on delay magnitude regression, limiting their ability to jointly model delay occurrence and severity. Furthermore, although recent research has explored graph-based representations of airport networks, effectively integrating structural network information into predictive models remains an open challenge.

To address these challenges, this study proposes a *Graph-enhanced Probabilistic Spatio-Temporal (GPST)* framework for flight departure delay prediction. The proposed approach integrates three complementary sources of information: tabular operational features derived from scheduling and weather data, temporal features capturing historical delay patterns, and graph-based airport embeddings representing structural relationships within the air transportation network. These embeddings provide a compact representation of airport connectivity, enabling the model to incorporate spatial dependencies across airports alongside traditional operational and temporal features.

The GPST framework adopts a probabilistic two-stage modeling strategy designed to handle the skewed distribution of delay values. In the first stage, a classification model estimates the probability that a flight will experience a departure delay exceeding the commonly used 15-minute threshold. In the second stage, the magnitude of the delay exceedance is predicted conditional on a delay occurring using a hybrid model that combines gradient-boosted tabular learning with a spatio-temporal neural network capturing historical delay dynamics. By separating delay occurrence from delay magnitude, the framework provides more stable predictions while effectively modeling both operational conditions and temporal delay propagation. The incorporation of graph-based airport embeddings further enhances this framework by enabling the model to capture spatial delay propagation patterns across the air transportation network.

The remainder of the paper is structured as follows. Section 2 provides an overview of related research. Section 3 describes the dataset, data preprocessing, feature engineering, and the proposed modeling framework. Section 4 presents

the experimental findings and discusses the results. Finally, Section 5 concludes the paper and outlines potential directions for future work.

2 Literature Review

Research on aviation delay prediction has grown rapidly with the rise of large operational datasets and machine learning. Studies predict departure delays, arrival delays, and estimated arrival times using both regression and classification. Methods range from traditional machine learning to deep learning and graph-based spatio-temporal models [21]. Across this literature, common challenges include feature engineering, handling imbalanced and changing temporal data, and balancing accuracy, interpretability, and computational efficiency in real-world deployment [12].

Many studies model departure delay as a continuous variable, emphasizing numerical prediction accuracy and identifying influential predictors. Anguita and Olariaga [5] compare ten regression models on Colombian departure data and find that Random Forest (RF) performs best, while showing that using only the most influential variables does not significantly degrade accuracy. Dong et al. [10] predict delays linked to Ground Delay Programs with an optimized multilayer perceptron, improving on several alternative methods and identifying temporal and weather factors as important. Falque et al. [11] analyze over 10 million records from Paris-Charles de Gaulle Airport and show that combining static and real-time operational features with LightGBM greatly improves prediction accuracy, though practical deployment may still be challenging.

In contrast, arrival delay prediction is often formulated as a binary classification task, where the objective is to determine whether a flight will be delayed beyond a threshold. In this context, domain-specific and network-based features have been shown to improve predictive performance [1]. Afrane et al. [2] utilize BTS flight records, weather data, and network connectivity features, finding that Extra Trees performs best overall, while airline-specific Random Forest models reach about 92.6% accuracy and 97% precision. In related work, Afrane et al. [3] train RF on 354,452 U.S. flights with about 50 engineered features, achieving 92.36% accuracy and 96.78% AUC. Ajayi et al. [4] similarly demonstrate that adding network centrality metrics improves RF, Gradient Boosting, and CatBoost models, with RF reaching 86.2% accuracy and an F1 score near 81. However, the binary formulation and reliance on historical network structure limit insights into delay magnitude and real-time operational dynamics.

Beyond point prediction, another body of work focuses on modeling uncertainty through probabilistic or distributional forecasts. Dalmau [8] applies NGBoost with Shapley explanations to predict reactionary delays and outperforms simple operational benchmarks, though results depend on dataset definitions. Wang et al. [22] predict long-term delay distributions for Guangzhou flights and achieve about 80% prediction interval accuracy. Mamdouh et al. [15] propose a delay propagation framework that uses schedule-based information across many airports and airlines, substantially reducing short-term forecasting errors.

Another closely related research direction focuses on predicting Estimated Time of Arrival (ETA) using trajectory-aware deep learning models. Silvestre et al. [20] integrate 4D trajectories, weather, and flight plan data in an LSTM model and achieve strong ETA accuracy at Madrid–Barajas Airport, though generalizability is uncertain. Deng et al. [9] propose a clustered deep learning framework, CC-MIDNN, for Lisbon flights that improves accuracy over benchmark models, albeit with increased architectural complexity.

A growing number of studies adopt graph-based representations, modeling airports as nodes and flight connections as edges in order to capture delay propagation across the network. Graph neural networks (GNNs) and attention mechanisms have become prominent tools. Zheng et al. [26] show that STGMAGNet improves long-term delay prediction by combining dynamic node embeddings, weather data, and multi-attention mechanisms, though at high computational cost. Similarly, Wu et al. [25] improve performance with a spatio-temporal propagation network that models weather and connectivity effects. Cai et al. [7] propose an adaptive graph-based model that also outperforms baselines, but still faces challenges in interpretability and stability as networks evolve.

Subsequent research attempts to capture more refined spatial structure by separating geographic proximity from operational dependencies or by clustering airports prior to modeling. Cai et al. [13] propose GOGCN, which separates geographic and operational relationships for delay inference. Tested on 1.8 million flights across 209 Chinese airports, it outperforms several baselines with MAE 7.742 and RMSE 9.619. Wei et al. [23] apply a TS-BiLSTM-Attention model on airport clusters formed with PSO-K-means, achieving better results than single-airport models on 982,439 flights across 229 airports (MAE $\approx$ 4.494, RMSE $\approx$ 6.787). Wei et al. [24] extend this with a dual-attention BiLSTM that incorporates external factors and Bayesian optimization, improving over ARIMA and recurrent baselines on BTS and Chinese datasets. However, mean-based objectives may underestimate extreme delays.

Another emerging direction considers operational environments where data sharing across airports is restricted. Shen et al. [19] propose a hybrid federated deep learning model that combines dynamic graph convolutions for spatial dependencies with residual GRUs for temporal patterns, using Louvain partitioning for privacy-preserving collaboration. On over 12 million BTS flight records from 52 airports, it achieves 0.8532 R^2, 6.98 RMSE, and 5.33 MAE for delays under 80 minutes. Shao et al. [18] introduce TrajCNN, which transforms surface trajectories and operational context into situational images for departure delay prediction. Tests at LAX keep errors below about 20 minutes and show that tarmac complexity matters more than weather or scheduling, though the method depends on dense sensing infrastructure.

In addition to machine learning and graph-based approaches, spatio-temporal data representation has long been studied in the database community. Early work on constraint query languages [14] and subsequent spatio-temporal database systems [17] emphasized modeling interactions across space and time. These

perspectives highlight the importance of structured representations for capturing delay propagation in air transportation systems.

Overall, aviation delay prediction has advanced significantly. Traditional machine learning works well with strong operational features, while deep learning and graph-based methods better capture temporal patterns and network-wide delay propagation. However, challenges remain, including handling skewed delay distributions, ensuring consistent evaluation, and balancing accuracy with practical deployment. These challenges motivate integrated frameworks that combine probabilistic modeling, temporal learning, and network-based representations, as proposed in this study.

3 Methodology

The dataset used in this study consists of monthly U.S. domestic flight records that include operational and scheduling information for commercial airline flights, along with weather conditions at both origin and destination airports. These records were obtained from the Bureau of Transportation Statistics (BTS) On-Time Performance database [6] and the Open-Meteo weather API [16]. The combined dataset spans the period from January 2024 to June 2025 and contains 11,356,339 flights connecting 343 airports across the United States. Aircraft seat capacity information was incorporated by matching each flight record with aircraft seat data using a constructed aircraft identifier derived from the airline carrier code and aircraft tail number. This additional information allows the model to capture operational characteristics related to aircraft capacity that may influence departure delays.

Because airlines differ substantially in operational scale, fleet composition, and route structure, prior work by Afrane et al. [2] shows that airline-specific preprocessing can reveal distinct operational patterns and improve predictive performance. Motivated by this observation, the present study focuses on the airline with the largest number of scheduled flights in the dataset, Delta Air Lines. Restricting the analysis to a single airline ensures that the predictive model is trained on a large and operationally consistent set of flight observations while reducing variability arising from differences in airline-specific operational practices.

3.1 Data Cleaning and Preprocessing

Data preprocessing was performed to ensure that the dataset was consistent and suitable for predictive modeling. Flights marked as cancelled or diverted were removed because their operational characteristics differ from normally completed flights and could introduce noise into delay prediction models. Records with missing values in key variables, particularly departure and arrival delays, were also excluded since these variables are required for constructing historical delay features.

To focus the analysis on the most operationally significant segments of the U.S. air transportation network, airports were filtered based on their connectivity within the flight network. Airports were ranked using *weighted degree centrality*, defined as the total number of flights connecting an airport to other airports in the network, aggregated across both incoming and outgoing routes. Airports with higher weighted degree centrality represent major hubs or highly connected airports that play a central role in network traffic. The top 30 airports with the highest weighted degree centrality values were selected, and the dataset was restricted to flights whose origin and destination airports both belong to this set, as illustrated in Figure 1, where these airports are highlighted in orange.

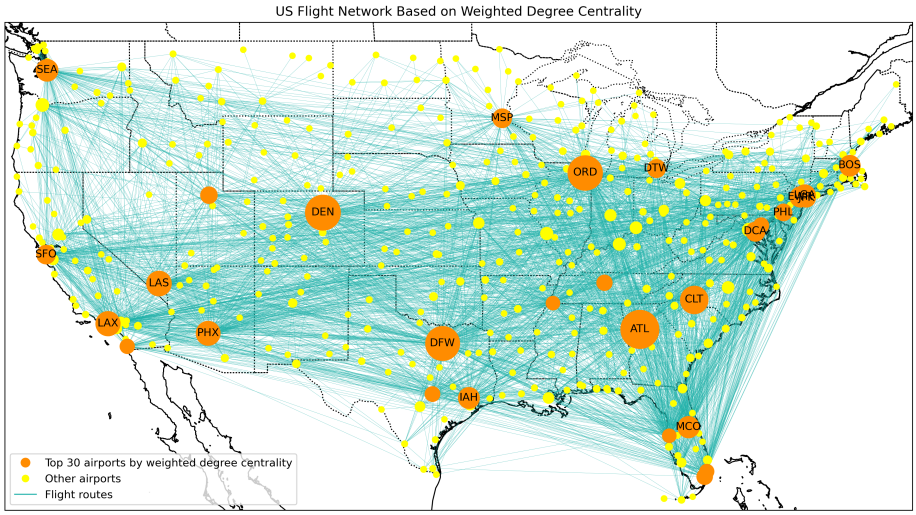


Fig. 1. U.S. flight network based on weighted degree centrality. The size of each node represents the airport's weighted degree centrality, with larger nodes indicating more highly connected airports. The top 30 airports with the highest centrality values are highlighted in orange.

Finally, the top 1% of departure delay values (above the 99th percentile) were removed to reduce the influence of rare extreme delays while preserving the majority of operational delay patterns. Airline flight numbers with very few observations were also excluded to ensure sufficient historical data for constructing reliable delay-related features.

After preprocessing, the resulting dataset contained 2,882,334 flight records with consistent attributes and validated numerical values. This cleaned dataset served as the foundation for the subsequent feature engineering and modeling stages.

3.2 Feature Engineering

Several categories of engineered features were created to capture temporal patterns, operational congestion, historical delay propagation, and structural relationships within the air transportation network.

Temporal Features. Flight operations exhibit strong cyclical patterns related to time of day, day of week, and seasonal trends. To capture these patterns, several temporal variables were constructed using cyclical encoding. A day-of-year variable was first computed by converting month and day values into a cumulative index within the calendar year. Because temporal variables are periodic in nature, sine and cosine transformations were applied to represent these variables on a circular scale and avoid discontinuities at cycle boundaries (e.g., day 365 followed by day 1). The same sine–cosine encoding was applied to day of year, day of week, scheduled departure time, and scheduled arrival time. Flight times were converted into minutes within a day before applying the transformations, producing continuous features that preserve the cyclical nature of flight schedules.

Airport Congestion Features. Airport congestion is an important contributor to flight delays. To represent operational traffic levels, several features were constructed based on the number of flights arriving at or departing from specific airports on a given day. These include the number of flights arriving at the destination airport, arriving at the origin airport, departing from the origin airport, and departing from the destination airport on the same day. These values were computed using group-by operations across airport and date combinations. Such features allow the model to capture relationships between airport traffic density and delay likelihood.

Historical Delay Features. Historical delays provide strong signals for future delays due to propagation effects in airline operations. To capture this phenomenon, rolling historical statistics were computed using chronologically ordered flight records. Features include the average departure delay over the previous seven flights of the same airline, the average arrival delay over the previous seven flights, and the average late aircraft delay for the same aircraft tail number within a rolling window. These metrics were calculated after shifting the dataset to ensure that only past information is used for prediction.

In addition to rolling averages, short-term delay propagation indicators were derived from the previous 24 hours of operations. These features measure average arrival and departure delays at origin and destination airports during the preceding 24-hour period, enabling the model to capture localized disruptions and recent operational conditions.

Graph Representation of the Air Transportation Network. Flight operations naturally form a network in which airports serve as nodes and flights

represent directed connections between them. To capture these spatial relationships, a graph representation of the U.S. air transportation system was constructed using historical flight data from 2023. In this graph, each directed edge represents flights between an origin and destination airport, and edge attributes include the number of flights, total seat capacity, and average route distance for that route. These attributes were normalized using logarithmic transformations and standardization to improve numerical stability.

Node-level features were also derived to represent airport activity levels, including the number of incoming flights, number of outgoing flights, total incoming seat capacity, and total outgoing seat capacity. Using this graph representation, a graph neural network was trained to learn vector embeddings for each airport. The learned embeddings summarize structural properties of the network, such as airport connectivity and traffic intensity, and provide a compact representation of spatial relationships among airports. These airport embeddings were subsequently incorporated as input features for the predictive models.

Graph Neural Network Architecture and Training. Airport embeddings were learned using a two-layer GraphSAGE encoder implemented in TensorFlow/Keras. The graph was constructed from 2023 segment-level flight data, with airports represented as nodes and directed flight routes represented as edges. Edge attributes included performed flights, seats, and route distance; flight and seat counts were transformed using $\log(1+x)$, while distance was standardized. Node features consisted of incoming flights, outgoing flights, incoming seats, and outgoing seats, all transformed using $\log(1+x)$.

The GraphSAGE encoder used mean neighborhood aggregation with a hidden dimension of 32. The first layer applied ReLU activation, while the second layer produced the final airport embedding. Dropout of 0.25 was applied between layers. For node v, the update operation is expressed as

$$h_v^{(k+1)} = \sigma\left(W^{(k)}\left[h_v^{(k)} \,\|\, \frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} h_u^{(k)}\right]\right).$$

The model was trained using a self-supervised link prediction objective. Observed routes were used as positive samples, while randomly sampled airport pairs were used as negative samples. Positive edges were sampled proportional to route flight volume, and negative destinations were sampled proportional to airport in-traffic. Edge scores were computed using the dot product of airport embeddings and optimized with binary cross-entropy loss. Training used Adam with a learning rate of 10^{-3}, 4096 positive and 4096 negative samples per step, and 200 epochs. The resulting 32-dimensional embeddings were joined to each flight record for both origin and destination airports and used as spatial features in GPST.

Sequential Temporal Features. Flight delays exhibit temporal dependencies that cannot be fully captured using static features alone. To model these dynamics, two sequential feature tensors were constructed.

The first tensor captures medium-term historical patterns over the previous seven days. For each flight, delay-related statistics, including average delays and flight counts, were computed for each of the preceding seven days and organized into a three-dimensional tensor representing observations, time steps, and feature channels.

The second tensor captures short-term dynamics within the previous 24 hours. The 24-hour period was divided into six four-hour intervals, and within each interval, average delays and flight counts were computed for both origin and destination airports. This representation provides high-resolution information on recent congestion and disruption patterns. By structuring these statistics as sequential tensors, the model can leverage sequence-based neural networks to capture temporal delay propagation.

3.3 Graph-enhanced Probabilistic Spatial-Temporal (GPST) Modeling

The predictive component of this study is based on a *Graph-enhanced Probabilistic Spatio-Temporal (GPST)* framework designed to model the spatial, temporal, and operational dynamics of flight delay propagation. Flight delays arise from interactions among airport connectivity, operational congestion, weather conditions, and historical delay patterns. Traditional machine learning models often treat these factors independently, limiting their ability to capture network-wide delay propagation.

The proposed GPST framework integrates three complementary sources of information: tabular operational features derived from scheduling, congestion, and weather variables; sequential temporal features capturing historical delay dynamics; and graph-based airport representations that encode structural relationships within the air transportation network. These components enable the model to jointly capture operational conditions, temporal delay patterns, and spatial interactions among airports.

The GPST framework adopts a probabilistic two-stage modeling strategy. The first stage estimates the probability that a flight will experience a departure delay exceeding the standard 15-minute threshold. The second stage predicts the magnitude of the delay conditional on a delay occurring. This formulation helps address the highly skewed distribution of flight delays while producing more stable delay magnitude predictions.

Integration of Graph-based Airport Representations. To capture spatial dependencies within the aviation network, airport representations were learned from the airport connectivity graph constructed from historical flight data. In this graph, nodes represent airports and directed edges represent flight routes between airports, with attributes including flight frequency, seat capacity, and route distance.

A graph neural network was trained on this graph to learn low-dimensional embedding vectors for each airport. These embeddings summarize structural properties of the air transportation network, such as airport connectivity and traffic intensity. The learned embeddings for the origin and destination airports of each flight are incorporated as additional features in the predictive models, allowing the framework to account for spatial interactions between airports.

Stage 1: Delay Occurrence Probability. The first stage predicts whether a flight will experience a departure delay exceeding 15 minutes. For each flight i, a binary label is defined as

$$y_i = \begin{cases} 1, & \text{if } DELAY_i \geq 15 \\ 0, & \text{otherwise.} \end{cases}$$

A LightGBM classifier is used to estimate the probability $p_i = P(y_i = 1 \mid x_i)$ that a delay will occur, where x_i represents the engineered feature vector for flight i. The input features include temporal features, airport congestion metrics, weather variables, historical delay statistics, and the learned airport embeddings. The resulting probability p_i represents the estimated likelihood that flight i will experience a departure delay beyond the threshold.

Stage 2: Delay Magnitude Prediction. While Stage 1 estimates the likelihood of a delay, Stage 2 predicts the magnitude of the delay conditional on a delay occurring. To account for the skewed distribution of delays, the regression task focuses on the delay exceedance beyond the 15-minute threshold:

$$y_i^{excess} = \max(DELAY_i - 15, 0).$$

The distribution of these exceedances is shown in Figure 2, illustrating the heavy-tailed nature of operational delay data.

The delay exceedance is predicted using a spatio-temporal neural network (STNN) that combines static operational features and sequential delay information. The model receives three types of inputs: tabular operational features (including airport embeddings), seven-day historical delay sequences, and short-term delay sequences derived from the previous 24 hours. The tabular features are processed through a multilayer perceptron, while the sequential inputs are encoded using temporal convolutional networks to capture historical delay patterns. The resulting latent representations are fused through fully connected layers to produce the predicted delay exceedance $\hat{y}_i^{excess}$.

Final Delay Prediction. The final predicted departure delay combines the probability estimate from Stage 1 with the conditional delay magnitude predicted in Stage 2:

$$\hat{y}_i = p_i \cdot \hat{y}_i^{excess}.$$

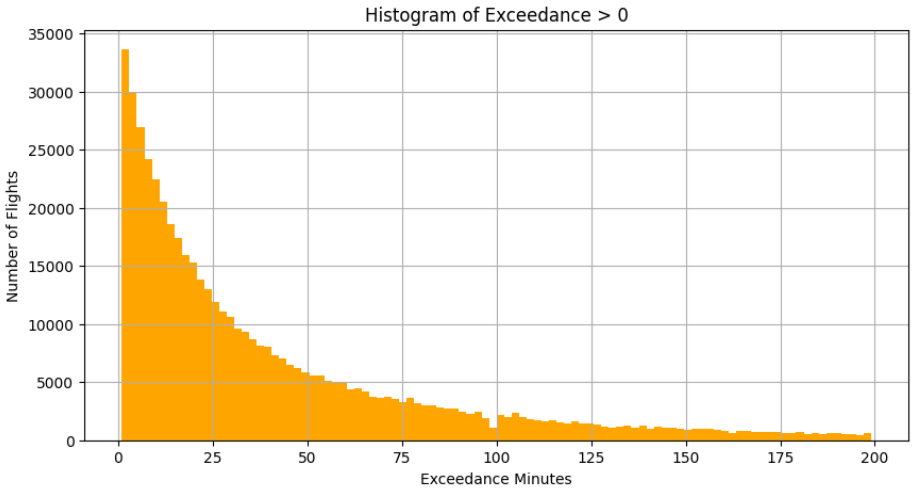


Fig. 2. Distribution of delay exceedances above the 15-minute threshold.

This formulation ensures that flights predicted to be on time (p_i close to zero) receive near-zero delay exceedance predictions, while flights predicted to be delayed receive delay estimates proportional to their predicted exceedance. By combining probabilistic delay occurrence modeling with conditional delay magnitude prediction, the GPST framework provides a robust approach for modeling the complex dynamics of flight departure delays.

3.4 Model Training and Evaluation

To evaluate the predictive performance of the proposed GPST framework, a rolling time-series cross-validation strategy was adopted. This evaluation approach preserves the chronological order of flight operations data and prevents information leakage from future observations into the training process. Unlike random cross-validation, which may mix past and future observations, rolling validation more closely reflects real-world forecasting scenarios in which models must predict delays using only historical information.

Specifically, a three-fold rolling validation scheme was implemented because the available evaluation period after the initial training window covered January–June 2025. This six-month period was divided into three consecutive two-month validation windows: January–February 2025, March–April 2025, and May–June 2025. In each fold, the model was trained on all available historical data prior to the validation window and then evaluated on the corresponding temporally unseen two-month period. This design preserves chronological ordering while maintaining sufficient validation size in each fold.

Model performance was assessed using several complementary evaluation metrics that capture both overall predictive accuracy and performance across

different delay ranges. The overall mean absolute error (MAE) and overall root mean squared error (RMSE) were computed to measure the average magnitude of prediction errors across all flights. While MAE provides a robust measure of typical prediction error, RMSE places greater emphasis on larger deviations and is therefore more sensitive to extreme delay events.

To further evaluate model performance within operationally meaningful delay ranges, additional metrics were computed for flights with delay exceedance between 0 and 60 minutes: $\text{MAE}_{0\text{–}60}$ and $\text{RMSE}_{0\text{–}60}$. These metrics focus on moderate delays, which represent the majority of operational disruptions and are often most relevant for airline schedule adjustments and passenger management.

Finally, a tolerance-based accuracy metric, denoted as acc@15, was also computed. This metric measures the proportion of predictions whose absolute error falls within 15 minutes of the observed delay exceedance. Such a tolerance-based metric provides a practical measure of prediction reliability, since small deviations within a short time window are often operationally acceptable in airline planning and delay management. Together, these evaluation metrics provide a comprehensive assessment of the GPST model's predictive performance across different delay magnitudes and operational contexts.

4 Results and Discussion

This section presents the experimental results of the proposed GPST framework under a rolling time-series cross-validation setting. We first evaluate its predictive performance across validation folds and delay ranges, and then compare it with baseline models to assess the impact of probabilistic modeling and graph-based airport representations.

4.1 GPST Model Performance

The predictive performance of the proposed GPST framework was evaluated using the rolling time-series cross-validation strategy described in Section 3. Table 1 summarizes the predictive performance across the three validation folds as well as the overall mean performance.

Table 1. GPST model performance across the three rolling validation folds and the overall mean.

Fold	overall MAE	overall RMSE	$\text{MAE}_{0\text{–}60}$	$\text{RMSE}_{0\text{–}60}$	acc@15
Fold 1	9.366	24.232	5.348	10.317	0.871
Fold 2	9.367	22.523	6.104	11.213	0.863
Fold 3	11.655	26.866	6.644	11.765	0.830
Mean	10.130	24.540	6.032	11.098	0.855

Across the three validation folds, the GPST model achieved stable predictive performance, with an average overall MAE of 10.13 minutes and an average overall RMSE of 24.54 minutes. These results indicate that the model can predict flight delay exceedance with relatively low average error, given the inherently stochastic nature of airline operations. Examining individual folds reveals consistent performance across different validation periods. In the first fold, the model achieved an overall MAE of 9.37 minutes and an overall RMSE of 24.23 minutes. The second fold produced similar results with an overall MAE of 9.37 minutes and a slightly lower overall RMSE of 22.52 minutes, indicating improved prediction of larger delays during that period. In contrast, the third fold exhibited somewhat higher prediction errors, with an overall MAE of 11.66 minutes and an overall RMSE of 26.87 minutes. This increase may reflect greater variability in operational conditions during the final validation window, potentially due to seasonal traffic changes or weather-related disruptions. Nevertheless, the results demonstrate that the GPST framework maintains stable predictive accuracy across different temporal validation periods.

To further assess model performance within practical delay ranges, prediction errors were also evaluated for flights with delay exceedance between 0 and 60 minutes. Across the three folds, the model achieved an average $\mathrm{MAE}_{0\text{–}60}$ of 6.03 minutes and $\mathrm{RMSE}_{0\text{–}60}$ of 11.10 minutes. These relatively small errors indicate that the model performs particularly well for moderate delays, which constitute the majority of operational delay events in airline networks.

The tolerance-based accuracy metric acc@15 provides an intuitive measure of forecast reliability. This metric represents the proportion of predictions whose absolute error falls within 15 minutes of the observed delay exceedance. Across the three validation folds, the GPST model achieved accuracy values of 0.871, 0.863, and 0.830, resulting in an overall average of 0.855. This indicates that approximately 85.5% of predictions fall within a 15-minute tolerance window of the observed delay exceedance. From an operational perspective, predictions within this range are often sufficiently accurate for airline planning tasks such as gate scheduling adjustments, passenger notifications, and disruption management.

4.2 Comparison with Baseline Models

To further assess the effectiveness of the proposed GPST framework, its predictive performance was compared with several baseline models, including an STNN model and a hybrid LightGBM+STNN model. These baselines represent commonly used approaches for modeling temporal delay dynamics using neural networks and combining gradient-boosted trees with sequential models. In addition, the Probabilistic Spatial–Temporal (PST) model, which adopts the same two-stage probabilistic formulation as GPST but does not incorporate graph-based airport embeddings, was evaluated to isolate the contribution of network-based spatial representations.

Table 2 and Figure 3 summarize the prediction error metrics across the baseline models, the PST model, and the proposed GPST framework. The results show that the conventional STNN model achieves an overall MAE of 11.609

minutes and an overall RMSE of 36.586 minutes. Incorporating gradient boosting through the LightGBM+STNN hybrid slightly reduces the overall RMSE to 31.013 minutes but results in a higher overall MAE of 12.906 minutes, indicating that simply combining gradient boosting with sequential neural models does not consistently improve predictive accuracy.

Table 2. Comparison of prediction error metrics across models.

Model	overall MAE	overall RMSE	$MAE_{0–60}$	$RMSE_{0–60}$
STNN	11.609	36.586	5.908	16.225
LightGBM+STNN	12.906	31.013	7.166	15.743
PST (GPST without graph embeddings)	**10.006**	**23.400**	6.265	11.251
GPST (Proposed)	10.045	24.492	**5.926**	**10.874**

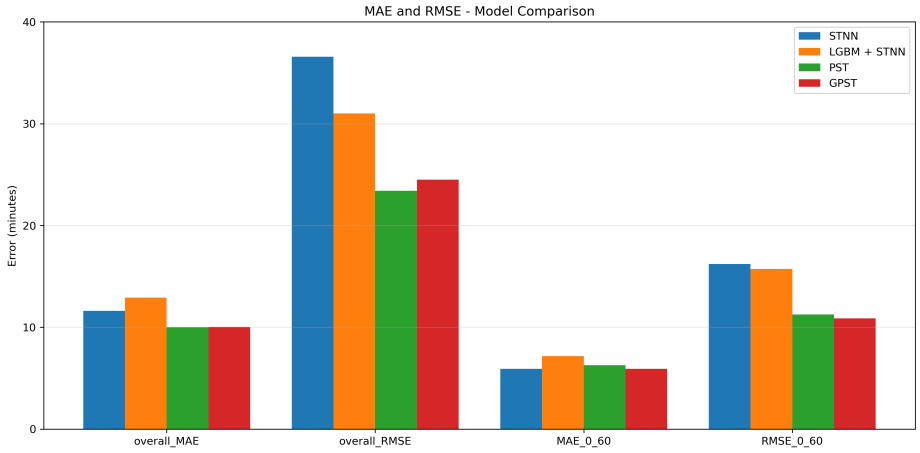


Fig. 3. Comparison of prediction error metrics (MAE and RMSE) and errors within the 0–60 minute delay range across models. Lower values indicate better predictive performance.

In contrast, the probabilistic two-stage modeling strategy used in PST leads to a substantial improvement in predictive performance. The PST model reduces the overall MAE to 10.006 minutes and the overall RMSE to 23.400 minutes, representing a significant reduction in prediction error compared with both baseline approaches. These results demonstrate the effectiveness of separating delay occurrence and delay magnitude into two complementary prediction stages, which helps address the highly skewed distribution of flight delay data.

Building on this probabilistic formulation, the proposed GPST framework further incorporates graph-based airport embeddings to capture spatial relationships within the air transportation network. While the overall MAE of GPST (10.045 minutes) remains comparable to that of PST, the GPST model achieves improved prediction accuracy within the operationally important delay range of 0–60 minutes. Specifically, GPST reduces $\text{MAE}_{0\text{–}60}$ to 5.926 minutes compared with 6.265 minutes for PST, and reduces $\text{RMSE}_{0\text{–}60}$ to 10.874 minutes compared with 11.251 minutes for PST. These improvements suggest that incorporating network-based airport representations helps the model better capture spatial delay propagation patterns, particularly for moderate delays that dominate real-world flight operations.

Prediction reliability was further evaluated using the Acc@15 metric, which measures the proportion of predictions whose absolute error falls within ±15 minutes of the true delay. As shown in Table 3 and Figure 4, the GPST model achieves the highest accuracy among all models, with an Acc@15 value of 0.859. This represents an improvement over PST (0.845), STNN (0.814), and LightGBM+STNN (0.790), indicating that approximately 85.9% of GPST predictions fall within a 15-minute tolerance of the observed delay.

Table 3. Prediction accuracy within a 15-minute tolerance (Acc@15).

Model:	STNN	LightGBM+STNN	PST	GPST (Proposed)
Acc@15:	0.814	0.790	0.845	**0.859**

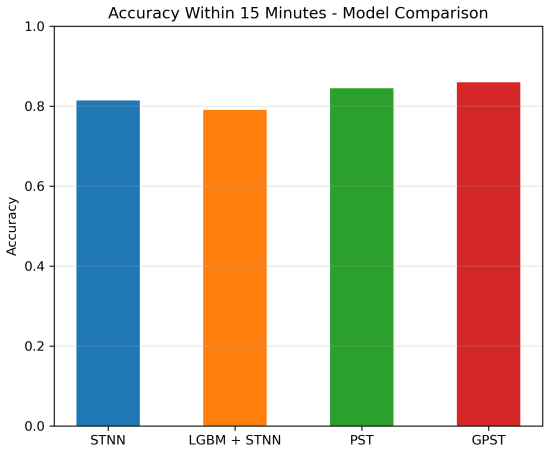


Fig. 4. Prediction accuracy within a 15-minute tolerance (Acc@15) for each model. Higher values indicate more predictions within ±15 minutes of the observed delay.

Overall, the comparison results reveal two key insights regarding the proposed modeling framework. First, the probabilistic spatial–temporal modeling strategy significantly improves predictive performance compared with conventional baseline models. The PST model consistently achieves lower overall MAE and RMSE than both STNN and the hybrid LightGBM+STNN model, demonstrating the effectiveness of separating delay occurrence prediction from delay magnitude estimation. This probabilistic formulation allows the model to better accommodate the highly skewed distribution of flight delay values and produces more stable delay predictions.

Second, incorporating graph-based airport embeddings further enhances predictive reliability within the GPST framework. Although the overall MAE of GPST is similar to that of PST, the GPST model achieves lower prediction errors within the operationally important delay range of 0–60 minutes and the highest tolerance-based accuracy among all models. These improvements suggest that the learned airport embeddings help capture spatial relationships and delay propagation effects across the air transportation network. In particular, representing airports through graph-based embeddings enables the model to incorporate structural connectivity information that cannot be easily captured using traditional tabular features alone.

Taken together, these results demonstrate that combining probabilistic modeling, temporal learning, and network-based spatial representations provides an effective framework for predicting flight delays. The proposed GPST model, therefore, offers a practical approach for capturing the complex spatial and temporal dynamics that influence delay propagation in large-scale air transportation systems.

5 Conclusion

This study introduced a Graph-enhanced Probabilistic Spatio-Temporal (GPST) framework for predicting flight departure delays using large-scale operational and weather data. The proposed approach integrates multiple sources of information, including temporal flight patterns, airport congestion indicators, historical delay propagation, and structural relationships within the air transportation network. By combining these heterogeneous representations, the GPST framework is able to capture both the temporal dynamics of flight operations and the spatial dependencies arising from airport connectivity.

Building upon a probabilistic two-stage modeling framework for delay prediction, the GPST framework incorporates graph-based airport representations to capture spatial relationships within the air transportation network. In the first stage, a classification model estimates the probability that a flight will experience a delay exceeding the commonly used 15-minute threshold. In the second stage, a spatial–temporal neural network predicts the magnitude of the delay exceedance using both tabular operational features and sequential historical delay patterns. This formulation helps address the highly skewed distribution of flight delay values and provides more stable predictions of delay magnitude. In

addition, graph-based airport embeddings learned from the air transportation network summarize structural connectivity patterns such as traffic intensity and airport influence. Integrating these embeddings with operational and temporal features enables the model to better capture spatial delay propagation across interconnected airports while accounting for both local conditions and broader network dynamics.

Experimental evaluation using rolling time-series cross-validation demonstrates that the GPST framework achieves stable predictive performance across validation periods, with an average MAE of approximately 10 minutes and high reliability under a 15-minute tolerance criterion. Comparative experiments show that the probabilistic spatial–temporal modeling framework significantly improves predictive accuracy relative to conventional baseline models, while the integration of graph-based airport embeddings further enhances prediction reliability within operationally important delay ranges. Overall, the results indicate that combining probabilistic modeling, temporal learning, and network-based spatial representations provides an effective approach for modeling complex delay dynamics in air transportation systems.

Despite these promising results, several limitations remain and suggest directions for future work. The current study focuses on a single airline and the top 30 highly connected airports, which improves data consistency but may limit generalizability; future work will extend the framework to multiple airlines and broader airport networks to evaluate robustness and scalability. In addition, the evaluation was conducted using a three-fold rolling time-series cross-validation scheme with consecutive two-month validation windows; while this reflects realistic forecasting scenarios, additional validation folds or longer evaluation periods could further strengthen the robustness of the results. Furthermore, although graph-based airport embeddings improve prediction reliability within operational delay ranges, the overall performance gain relative to the PST framework remains modest. Future work will explore more advanced spatio-temporal graph neural network architectures, including attention-based and dynamic graph models, to better capture evolving network dependencies. Finally, extending the current offline framework to real-time or near-real-time deployment, including integration with live data streams and scalable processing pipelines, represents an important direction for practical applications.

References

1. Afrane, M.D.: Network-aware airline-specific flight delay prediction using tree-based ensemble models (2026)
2. Afrane, M.D., Xu, Y., Li, L., Wang, K.: Airline-specific flight delay prediction with tree-based models and network metrics. In: 2025 6th International Conference on Artificial Intelligence, Robotics and Control (AIRC). pp. 535–540. IEEE (2025)
3. Afrane, M.D., Xu, Y., Li, L., Wang, K.: Flight delay prediction using random forest with enhanced feature engineering. In: SoutheastCon 2025. pp. 1055–1056. IEEE (2025)

4. Ajayi, J., Xu, Y., Li, L., Wang, K.: Enhancing flight delay predictions using network centrality measures. Information **15**(9), 559 (2024)
5. Anguita, J.M., Olariaga, O.D.: Prediction of departure flight delays through the use of predictive tools based on machine learning/deep learning algorithms. The Aeronautical Journal **128**(1319), 111–133 (2024)
6. Bureau of Transportation Statistics: On-Time Performance Data. `https://www.transtats.bts.gov/ONTIME/` (2025), accessed: 2025-11-24
7. Cai, K., Li, Y., Fang, Y.P., Zhu, Y.: A deep learning approach for flight delay prediction through time-evolving graphs. IEEE Transactions on Intelligent Transportation Systems **23**(8), 11397–11407 (2021)
8. Dalmau, R.: Probabilistic and explainable tree-based models for rotational reactionary flight delay prediction. CEAS Aeronautical Journal **15**(4), 1157–1173 (2024)
9. Deng, W., Li, K., Zhao, H.: A flight arrival time prediction method based on cluster clustering-based modular with deep neural network. IEEE Transactions on Intelligent Transportation Systems **25**(6), 6238–6247 (2023)
10. Dong, X., Zhu, X., Zhang, J.: Departure flight delay prediction due to ground delay program using multilayer perceptron with improved sparrow search algorithm. The Aeronautical Journal **128**(1322), 706–724 (2024)
11. Falque, T., Mazure, B., Tabia, K.: Machine learning for predicting off-block delays: A case study at paris—charles de gaulle international airport. Data & Knowledge Engineering **152**, 102303 (2024)
12. Huynh, T.K., Cheung, T., Chua, C.: A systematic review of flight delay forecasting models. In: 2024 7th International Conference on Green Technology and Sustainable Development (GTSD). pp. 533–540. IEEE (2024)
13. Kaiquan, C., Yue, L., Yongwen, Z., Quan, F., Yang, Y., Wenbo, D.: A geographical and operational deep graph convolutional approach for flight delay prediction. Chinese Journal of Aeronautics **36**(3), 357–367 (2023)
14. Kanellakis, P.C., Kuper, G.M., Revesz, P.Z.: Constraint query languages (preliminary report). In: Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. pp. 299–313 (1990)
15. Mamdouh, M., Ezzat, M., A. Hefny, H.: A novel intelligent approach for flight delay prediction. Journal of Big Data **10**(1), 179 (2023)
16. Open-Meteo: Free Weather API. `https://open-meteo.com/` (2025), accessed: 2025-11-24
17. Revesz, P.: Introduction to databases: From biological to spatio-temporal. Texts in Computer Science (2010)
18. Shao, W., Prabowo, A., Zhao, S., Koniusz, P., Salim, F.D.: Predicting flight delay with spatio-temporal trajectory convolutional network and airport situational awareness map. Neurocomputing **472**, 280–293 (2022)
19. Shen, X., Chen, J., Yan, R.: A spatial–temporal model for network-wide flight delay prediction based on federated learning. Applied Soft Computing **154**, 111380 (2024)
20. Silvestre Vilches, J., Martínez Prieto, M.A., Bregón Bregón, A., Álvarez Esteban, P.C., et al.: A deep learning-based approach for predicting in-flight estimated time of arrival. The Journal of Supercomputing **80**(12), 17212–17246 (2024)
21. Wandelt, S., Chen, X., Sun, X.: Flight delay prediction: a dissecting review of recent studies using machine learning. IEEE Transactions on Intelligent Transportation Systems (2025)
22. Wang, F., Bi, J., Xie, D., Zhao, X.: Flight delay forecasting and analysis of direct and indirect factors. IET Intelligent Transport Systems **16**(7), 890–907 (2022)

23. Wei, X., Li, Y., Shang, R., Ruan, C., Xing, J.: Airport cluster delay prediction based on ts-bilstm-attention. Aerospace **10**(7), 580 (2023)
24. Wei, Z., Zhu, S., Lyu, Z., Qiao, Y., Yuan, X., Zhao, Y., Zhang, H.: Multi-step regression network with attention fusion for airport delay prediction. IEEE Transactions on Intelligent Transportation Systems **25**(7), 7093–7105 (2024)
25. Wu, Y., Yang, H., Lin, Y., Liu, H.: Spatiotemporal propagation learning for network-wide flight delay prediction. IEEE Transactions on Knowledge and Data Engineering **36**(1), 386–400 (2023)
26. Zheng, H., Wang, Z., Zheng, C., Wang, Y., Fan, X., Cong, W., Hu, M.: A graph multi-attention network for predicting airport delays. Transportation Research Part E: Logistics and Transportation Review **181**, 103375 (2024)

An Execution-Based Framework for Automated Assessment of ER Models

Milan Todorovikj[1][0009−0004−8259−1293] and Goran Velinov[2][0000−0002−6848−9514]

Faculty of Computer Science and Engineering,
Ss. Cyril and Methodius University, 1000 Skopje, North Macedonia
{milan.todorovikj, goran.velinov}@finki.ukim.mk

Abstract. Evaluating conceptual ER diagrams manually is intensive and subjective. Current automation tools rely on static analysis and fail to verify whether designs enforce business rules functionally. We propose an execution-based framework for automatically grading models designed in the Peter Chen notation. By transforming XML models into executable schemas via an Intermediate Representation, the system evaluates core ER concepts. To resolve naming differences, a schema-matching engine aligns student schemas with the instructor's reference. Executing SQL data manipulation tests on historical exams yields scores that correlate statistically with expert grades, providing an objective assessment metric.

Keywords: Automated Assessment · Conceptual Data Modeling · Entity-Relationship Diagrams · Schema Matching · Database Education.

1 Introduction

Conceptual data modeling forms a core component of software engineering and database design education. The Peter Chen notation[1] serves as the standard formalism for teaching Entity-Relationship (ER) diagrams due to its intuitive visual language and semantic richness for capturing business requirements. In academic environments, ensuring student competency in ER modeling poses a significant assessment challenge. Evaluating student ER submissions inherently involves tedious manual processing. As class sizes grow, manual grading not only becomes unsustainable but also introduces subjective inconsistencies among different graders.

To address this, various automated assessment tools have been proposed. However, the majority of these systems rely entirely on static analysis, which compares the syntax, structural graphs, or XML nodes of a student's diagram against a reference solution. While static evaluation verifies whether a diagram visually resembles the reference answer, it does not validate the behavioral semantics of the design. A student's ER diagram often appears structurally sound yet fails to enforce specified business rules. Semantic flaws go unnoticed during a visual inspection, but upon deployment as a physical schema, these omissions compromise data integrity by permitting invalid data operations.

To overcome these limitations, we propose a transition from static visual inspection to dynamic, execution-based validation. This paper introduces an automated testing framework that evaluates student ER diagrams by generating their conceptualized databases and executing SQL data manipulation (DML) tests against them. By treating the instructor's solution as the ground-truth oracle, the system verifies whether the student's generated Data Definition Language (DDL) practically enforces the intended semantic rules.

The proposed framework consists of several key components that correspond to the subsequent sections of this paper. First, to bridge the gap between visual diagrams and executable databases, the framework leverages our previously developed Intermediate Representation (IR) pipeline [2] that translates raw visual models into DDL scripts. The current work extends this IR to capture behavioral semantics, specifically introducing tracking for participation constraints across relationship cardinalities, transforming these conceptual rules into executable relational constraints. Next, to address the semantic challenge of students using different, yet domain-equivalent, naming conventions, the framework integrates an automated schema-matching engine. This component aligns the student's schema with the instructor's ground truth before test execution. Finally, the framework generates positive and negative functional tests routed to the corresponding target structures to establish a final grade. To evaluate this approach, the study applies the framework to a preliminary dataset of historical university exams, comparing the automated scores with instructor-assigned grades. To foster reproducibility and provide a public benchmark for future research, we have anonymized and open-sourced this dataset [15].

Key contributions of this research are as follows:

- Introduced an execution-based automated assessment framework that dynamically grades conceptual ER diagrams using SQL data manipulation tests.
- Extended an XML-to-DDL IR pipeline to explicitly extract and enforce advanced semantic constraints, such as total participation, via automated database triggers.
- Integrated a hybrid schema-matching engine utilizing semantic similarity and bipartite optimization to align divergent student naming conventions with the instructor's reference solution.
- Conducted an empirical evaluation on a dataset of historical university exams, demonstrating that execution-based automated scores correlate statistically with manual expert evaluations, providing a scalable and objective assessment metric.

The remainder of this paper is organized as follows. Section 2 reviews the background and related literature, highlighting the current research gaps in the automated assessment of conceptual models. Section 3 details the proposed system architecture, explaining the transformation of ER models into executable DDL, the extraction of semantic constraints, and the schema-matching engine. Section 4 presents the dynamic testing framework alongside the grading methodology and discusses the results of our empirical evaluation using historical stu-

dent data. Finally, Section 5 concludes the paper and outlines directions for future work.

2 Background and Related Work

To contextualize the contributions of this execution-based grading framework, this section reviews relevant literature across three primary domains: automated assessment of conceptual models, conceptual model translation, and automatic schema matching.

2.1 Automated Assessment of Conceptual Models

Automated assessment tools provide scalable and immediate feedback in computer science education [11]. In the database domain, the automated grading of SQL queries relies heavily on dynamic testing, which executes student queries against a populated dataset to evaluate behavioral correctness [3]. Recent advancements have even extended this execution-based paradigm beyond basic data retrieval to actively evaluate Data Definition Language (DDL) and Data Manipulation Language (DML) statements within isolated database environments [12].

Despite the dynamic nature of SQL assessment, the automated evaluation of conceptual data models, such as ER diagrams, operates differently. A systematic literature review by Ullrich et al. [4], covering 110 publications on the automated assessment of conceptual models, indicates that existing tools rely entirely on static analysis. Specifically, modern systems utilize static model comparison (e.g., graph matching) or rule-based checking to assess syntactic and semantic correctness. Early foundational work by Thomas, Waugh, and Smith [6] [13] [14] pioneered this approach by extracting minimal meaningful units from student ER diagrams and evaluating them via structural pattern matching.

As a more recent example of static assessment, Tanaka et al. [5] propose an evaluation method for conceptual data models that calculates similarities based on sets of attributes. Rather than relying strictly on exact entity name matches, their system evaluates the semantic equivalence of models by analyzing the overlap and composition of attribute sets. While approaches similar to those of Tanaka et al. [5] and standard graph-isomorphism tools[6] [13] detect missing structural components, they evaluate the syntax and structural semantics of the model rather than its executable behavior.

Consequently, a statically graded diagram provides no guarantee that the physical database enforces critical data integrity rules. This research bridges this gap by applying the execution-based paradigm, traditionally reserved for SQL queries and DDL/DML statements, directly to the conceptual modeling phase.

2.2 Conceptual Model Translation and Intermediate Representation

The mapping of conceptual ER models to logical relational schemas is a core concept in database theory [7]. Visual modeling tools facilitate the creation of ER diagrams, and specific frameworks support the generation of DDL scripts from these representations.

However, generic diagramming tools store models as raw graphical XML that represents coordinates, shapes, and lines, without underlying database semantics. To automatically evaluate these diagrams through execution, the system first transforms the graphical syntax into a logical format. As introduced in Section 1, this framework builds upon our foundational IR pipeline [2].

This pipeline extracts graphical components and translates them into relational objects. In this research, we extend the IR by adding tracking of total participation across all relationship types. This extension ensures that the generated DDL enforces these semantic rules, enabling dynamic testing of the resulting database.

2.3 Schema Matching and Semantic Alignment

A challenge in automatically executing tests against student-generated schemas is semantic heterogeneity. In open-ended modeling exercises, students use different naming conventions or synonyms compared to the instructor's ground-truth solution. Therefore, before the system executes functional tests generated from the instructor's model on the student's database, it must formally align the two schemas.

Automatic schema matching constitutes a core area in data integration. Rahm and Bernstein [8] provide a taxonomy of matching approaches, categorizing them into element-level (linguistic and data-type comparisons) and structure-level methods. To resolve student-to-instructor mappings, the proposed framework employs a hybrid approach combining linguistic matching and structural attribute-set comparisons, similar to Tanaka et al. [5].

By combining semantic name similarity with structural metadata, such as data types and primary key constraints, the system calculates similarity scores between schema elements. The framework models schema mapping as a bipartite optimization problem [9] to produce a structural alignment. The system applies this alignment mapping during the DDL generation phase, substituting the student's original identifiers with the instructor's expected names before database instantiation. Consequently, the student's physical schema adopts the naming conventions of the reference solution, which enables the unmodified functional tests to execute directly against the student's database.

2.4 Summary and Research Gaps

Recent advancements in educational technology have opened new possibilities for automating database assessments. However, existing tools primarily offer visual and structural comparison-based assessments, falling short of offering a complete functional evaluation pipeline. Several critical research gaps remain:

1. **Reliance on Static and Syntactic Analysis:** Most systems perform structural comparisons (e.g., graph matching) but fail to validate the executable behavior of the design. Consequently, a statically graded diagram provides no guarantee that the physical database dynamically enforces critical data integrity rules.
2. **Incomplete Semantic Constraint Translation:** Existing transformation pipelines often fail to extract and enforce advanced behavioral constraints, such as total participation, preventing the generated schemas from serving as fully constrained, testable physical environments.
3. **Vulnerability to Semantic Heterogeneity:** Current ER grading tools struggle to accurately evaluate submissions when students use divergent naming conventions. Without automated semantic alignment, systems are rigid and prone to penalizing structurally correct solutions simply because of vocabulary differences.
4. **Absence of Test Generation for Conceptual Models:** While the execution-based paradigm is well-established for SQL queries, there is a lack of end-to-end frameworks that automatically generate and execute Data Manipulation Language (DML) tests to verify conceptual ER diagrams.

Addressing the Gaps: Our proposed framework bridges these gaps by applying the execution-based paradigm—traditionally reserved for SQL queries-directly to the conceptual modeling phase. By transforming ER models into executable schemas via an extended IR, utilizing a bipartite schema-matching engine to resolve naming differences, and dynamically executing SQL data manipulation tests, this end-to-end approach introduces a scalable, functional validation solution for database education.

3 Automated Assessment Framework

3.1 ER Model to DDL Transformation

The assessment framework relies on our prior transformation pipeline [2] to translate raw visual ER diagrams into executable relational schemas. This pipeline operates independently of the dynamic testing framework and bridges the gap between graphical representations and physical databases. The transformation process is divided into conceptual interpretation and schema synthesis.

In the domain of Model-Driven Engineering (MDE) and conceptual model analysis, several standard IRs exist, primarily falling into three categories: standard markup formats like XML Metadata Interchange (XMI), Abstract Syntax Graphs (ASG), and direct relational metamodels. However, these state-of-the-art representations exhibit specific limitations when applied to execution-based automated assessment.

First, standard interchange formats like XMI are highly generic and verbose, often entangling conceptual semantics with diagrammatic presentation details (coordinates and shapes), making deterministic translation to executable constraints difficult. Second, graph-based representations (ASGs) are widely used in

existing automated grading tools [6]. While ASGs are highly effective for static, structural pattern matching (graph isomorphism), they lack the explicit relational mechanics, such as dependency tracking and cardinality-driven foreign key formulation, required to generate an executable database environment. Finally, direct translation to relational metamodels, mapping directly to tables and columns, strips away the conceptual origin of the data. Once an ER diagram is reduced purely to a relational model, the system loses the ability to distinguish whether a table originated as a regular entity, a weak entity, or an N:M associative relationship. This loss of concept ual context severely limits the system's ability to apply granular, pedagogical grading rubrics and complicates semantic schema alignment.

Our extended IR bridges the gap between these extremes by acting as a normalized, execution-ready abstraction layer. By enforcing a strict separation of concerns, our IR, unlike XMI or raw XML, completely abstracts away syntactical and geometric diagram details to strictly isolate semantic relational dependencies. Furthermore, rather than reducing designs to direct relational metamodels, our representation ensures conceptual preservation by explicitly tracking the conceptual origins of elements, such as distinguishing multivalued attributes from associative tables. Finally, unlike static ASGs, the proposed IR introduces executable behavioral encoding by capturing metadata such as total participation flags across topologies. This unique feature directly enables the dynamic testing phase (Section 3.3) by allowing the system to deterministically generate the active database triggers necessary for validating semantic business rules. Consequently, this IR is not merely a translation step, but a foundational component that makes dynamic, functional parity testing of conceptual models possible.

During conceptual interpretation, the pipeline ingests XML files exported from the draw.io diagramming tool. A heuristic inference engine analyzes the geometric properties and structural connections of the XML nodes to classify them into ER constructs, such as entities, attributes, and relationships. The system processes various modeling elements, including weak entities, multivalued attributes, and n-ary relationships.

Rather than translating visual elements directly into SQL statements, the pipeline maps these constructs into a structured IR. The IR formalizes the model independent of database implementation specifics. Each relation descriptor within the IR contains a set of attribute identifiers, explicitly defined primary keys, and foreign key specifications. The representation stores foreign key dependencies explicitly, recording the referenced entity and an initial nullability flag. The pipeline maintains an internal mapping between these structural identifiers and the user-defined names present in the diagram.

During schema synthesis, the pipeline translates the IR into DDL scripts. To ensure referential integrity during database creation, the system constructs a dependency graph based on the foreign key relations in the IR. A topological sorting algorithm traverses this graph to establish a valid execution order. This sorting guarantees that the DDL script creates referenced tables before any dependent tables that contain foreign keys pointing to them.

To generate the SQL statements, the pipeline determines the data type for each attribute. It applies pattern-based rules to map attribute naming conventions to compatible SQLite data types. If an attribute name matches no specific rule, a zero-shot classification model evaluates the name to infer the SQLite data type. Finally, the pipeline translates the sorted IR relations into CREATE TABLE statements. It appends table-level constraints for primary and foreign keys, producing an executable SQLite schema.

3.2 System Architecture

The framework for automating assessments operates as a multi-stage pipeline, as illustrated in Fig. 1. It takes two conceptual ER models as input, the instructor's reference solution (the ground truth) and the student's submission, and transforms them into physical databases to conduct a comparative, execution-based analysis. The architecture isolates the conceptual modeling environment from the physical testing environment through six phases.

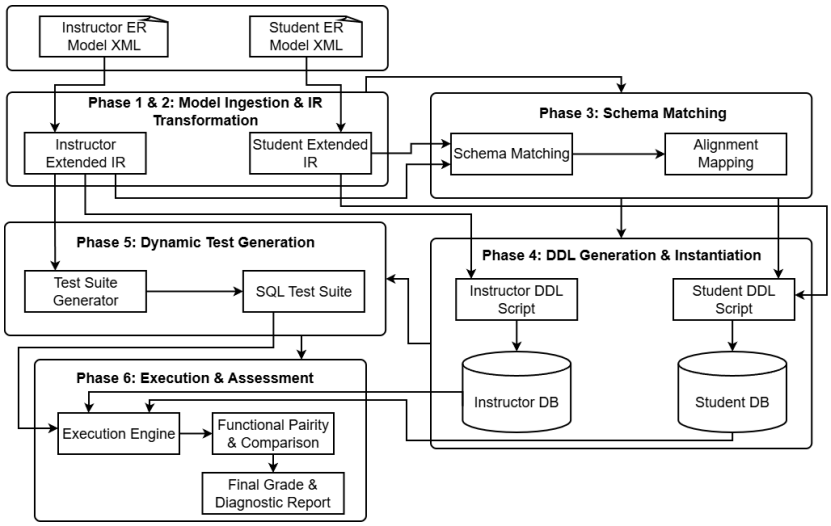


Fig. 1. System architecture of the automated execution-based assessment framework

Phase 1 begins the model ingestion process by reading raw graphical XML files exported from visual modeling tools. At this stage, the inputs represent graphical coordinates, shapes, and connecting edges, without formal relational database semantics. The system processes the instructor's and the student's XML files independently.

In Phase 2, the system parses the raw XML files and transforms them into an extended IR. This IR normalizes visual elements into logical entities, attributes, and relationships. During this phase, the system extracts semantic constraints.

By analyzing cardinality and connector annotations in the XML, it identifies and records total participation constraints across relationship types as IR's metadata.

After constructing the IR structures, Phase 3 performs schema matching and alignment. Because students use a different vocabulary from the instructor, the system aligns the two isolated IRs. The schema-matching engine evaluates the student's IR against the instructor's IR using semantic and structural similarity scores. By modeling this as a bipartite optimization problem, the system determines a 1:1 mapping between the student's tables and attributes and the instructor's corresponding elements. This alignment map translates the student's identifiers into the instructor's expected names.

In Phase 4, the system generates physical DDL scripts for both the instructor's and the student's IRs. During generation, the system applies Phase 3 alignment mapping to substitute the student's original identifiers with the instructor's names. The system executes these scripts to create two separate relational databases. The instructor's database serves as the functional oracle, defining the expected behavior, while the student's database serves as the test subject, providing the operational output.

With both schemas instantiated, the framework proceeds to Phase 5: dynamic test generation. The system generates a suite of test cases derived from the instructor's ground-truth IR. The test suite includes valid INSERT statements that populate tables and verify structural configurations, such as primary key and foreign key constraints. It also includes invalid INSERT, UPDATE, or DELETE statements designed to trigger semantic constraint violations, such as attempts to bypass a total participation rule.

Finally, Phase 6 performs execution and grading. The system executes the generated test suite concurrently against the instructor's oracle database and the student's test database. The system captures the relational engine's response to each transaction. If the student's database behaves identically to the instructor's database, committing valid data and rejecting invalid data with constraint errors, the student receives points. If the student's database allows an operation that the instructor's database rejects, or vice versa, the system records a semantic flaw. The final output is a grade accompanied by a report, indicating which tests the student's design passed or failed.

3.3 Semantic Constraint Extraction and Execution

To evaluate graphical models through SQL execution, the system first translates the raw visual XML into an IR. As detailed in our foundational pipeline[2], this IR acts as a structural blueprint. It normalizes graphical nodes and edges into object-oriented representations, records the unique identifiers of elements, maps them to corresponding tables, maps attribute IDs to their respective tables, and defines primary and foreign key dependencies.

While this IR captures the structural syntax of the model, the current framework extends it to capture behavioral semantics. The framework introduces a total participation mapping of entities within the IR. During the parsing phase, the system evaluates every relationship and records the topological context of all

participating entities. By identifying visual cues of double lines denoting total participation, the extended IR logs indicate that entities hold a total participation constraint. The system records this mapping regardless of whether the relationship produces a new table in the final DDL or resolves as a foreign key within an existing table. This mapping allows the system to determine where dependency rules apply and extract them as constraint objects.

Standard SQL DDL supports structural constraints such as primary keys and foreign key dependencies. However, translating conceptual ER constraints, such as total participation, into executable relational constraints requires additional mechanisms. While a NOT NULL foreign key ensures a record in a referencing table maps to a valid record in a referenced table, it does not enforce the inverse constraint, where a record in the referenced table must associate with at least one record from the referencing table. Because the target database engine, SQLite, lacks built-in assertions for inter-table dependencies, the framework generates database triggers to enforce total participation rules. The system generates these triggers for the entity side that has total participation. To achieve this, the system constructs AFTER INSERT triggers on the referenced entity table, alongside AFTER UPDATE and AFTER DELETE triggers on the referencing table, which is either another entity table or a relationship table.

The logic ensures that an instance of the constrained entity retains a relationship reference. The framework enforces this through the actions of each trigger. The AFTER INSERT trigger on the constrained entity table verifies that whenever a transaction inserts a new row into the referenced table, it also inserts a corresponding row into the referencing table. This ensures the referenced table instance is associated with the referencing table instance upon creation. The AFTER DELETE trigger on the referencing table activates when a transaction removes a relationship or referencing record. It executes a subquery to verify whether the referenced entity instance retains at least one foreign key referencing it. The AFTER UPDATE trigger on the referencing table activates when a transaction modifies a foreign key referencing the constrained entity. Similar to the delete trigger, it executes a validation subquery to ensure that changing the reference does not leave the associated entity instance orphaned.

If any of these trigger conditions fail, meaning the entity no longer participates in the required relationship, the trigger executes a RAISE(ABORT) statement to reject the transaction. This trigger-based execution model applies across all relationships. By deriving the trigger generation from the total participation side recorded in the IR, the algorithm constructs the validation subqueries using the same mapping logic. To support the testing phase without causing initialization deadlocks, the system binds the generated triggers to a system_flags control table. This table acts as a state toggle containing a validation_enabled integer. The generated triggers evaluate this flag and abort transactions only when it equals 1. This addition allows the database to accept initial test data before enforcing the semantic constraints during evaluation.

3.4 Schema Matching and Alignment

In open-ended conceptual modeling exercises, students use different naming conventions, synonyms, or scripts compared to the instructor's reference solution. To automate the assessment process, the framework integrates a schema-matching engine [8]. It formulates this alignment as a maximum-weight bipartite matching problem and executes it in two stages: table-level matching followed by attribute-level matching.

To establish table-level alignments, the framework calculates a pairwise similarity matrix between the instructor's tables and the student's tables. The similarity score for each pair of tables is a weighted composite of four signals. First, the system evaluates the linguistic similarity of the table names. Second, it calculates the linguistic similarity of the attribute sets. Third, it computes a structural similarity score based on the ratio of attribute counts between the two tables. Finally, it measures the data type distribution similarity by comparing the frequency of canonical SQL types in both tables.

To compute the linguistic similarities, the framework utilizes a pre-trained multilingual sentence transformer model [10] to handle cross-lingual and synonymous matches. This model generates vector embeddings for the parsed table and attribute names, and the system calculates the cosine similarity between these vectors. For the data type comparisons, the framework normalizes SQL types, such as VARCHAR and BIGINT, into canonical groups, such as TEXT and INTEGER. It assigns a compatibility score depending on whether the types belong to the same group, a related category, or are incompatible.

After the system populates the table similarity matrix, it applies the Hungarian algorithm [9] to determine the one-to-one bipartite matching that maximizes the similarity score. To prevent the mapping of unrelated entities, the system enforces a confidence threshold. It flags any pairing that falls below this threshold as a missing or extra table rather than a match. After the system aligns the tables, it performs a secondary bipartite matching to align the attributes within each matched table pair. This attribute-level alignment relies on a similarity matrix that combines the attribute name embedding similarity and the SQL type compatibility score. Similar to the table matching phase, the Hungarian algorithm extracts the attribute pairs subject to a confidence threshold.

The output of this schema matching phase is an alignment mapping that translates the student's table and column names into the instructor's schema names. Rather than modifying the DML queries during testing, the framework utilizes this translation during the DDL generation phase. It substitutes the student's names with the matched names from the instructor's reference solution before database instantiation. Consequently, the student's physical database adopts the naming conventions of the ground truth. This alignment ensures that the unmodified SQL test cases, generated from the instructor's model, execute against the student's database.

4 Dynamic Testing and Empirical Evaluation

4.1 Dynamic Testing Component

The dynamic testing component serves as the evaluation engine of the framework. Rather than inspecting the student's conceptual model, this component generates and executes a suite of SQL tests to verify whether the physical schema enforces the business logic. By deriving tests from the instructor's ground-truth IR and running them against the student database instance, the framework establishes an execution-based assessment. The generated test suite is divided into structural metadata tests and behavioral data manipulation tests.

To validate the relational mechanics, the framework generates structural metadata tests for every table in the schema. Because the relational configurations reside within the database engine's catalog, these tests utilize internal SQLite PRAGMA statements. Primary Key tests query PRAGMA table_info to verify the primary key designations. The query structure for primary key tests appears in Listing 1.

Listing 1. SQL query utilizing SQLite PRAGMA to verify primary key designations.

```
SELECT pk FROM pragma_table_info('target_table');
```

Foreign Key tests leverage PRAGMA foreign_key_list to validate inter-table dependencies, ensuring that referencing attributes map to their referenced columns, as demonstrated in Listing 2.

Listing 2. SQL query utilizing SQLite PRAGMA to validate foreign key dependencies and mapping.

```
SELECT "table", "from", "to" FROM
pragma_foreign_key_list('target_table');
```

By generating these structural tests for each table, the system confirms the physical integrity of the database before attempting data manipulation.

Beyond structural validation, the system generates positive insertion tests for each table to ensure the schema accepts valid data without violating referential integrity. Because inserting a row into a dependent table requires existing rows in its referenced tables, the testing algorithm resolves these dependencies. For a given target table, the framework identifies the hierarchical chain of referenced tables. It then inserts type-inferred dummy data, such as sequential integers, strings, and dates based on column definitions, into every referenced table. After populating the prerequisite relational hierarchy, the system inserts the target row into the tested table. The transaction sequence appears in Listing 3:

Listing 3. Generalized SQL execution sequence for validating positive insertions with hierarchical referencing dependencies.

```
INSERT INTO referencing_table1 (cols...) VALUES (vals...);
-- ... additional referencing table insertions ...
INSERT INTO target_table (cols...) VALUES (vals...);
SELECT * FROM target_table;
```

To evaluate behavioral semantics, the framework generates total participation constraint tests for every entity-relationship pairing that exhibits total participation. Executing these tests requires transactional state management to prevent initialization deadlocks. The generated SQL test sequence disables standard database foreign key checks and deactivates the triggers by updating the system_flags validation toggle to zero. Within a transactional savepoint, the system inserts dummy rows into both the target entity table and its referencing table (either a relationship table or another entity). Following these insertions, the system enables the validation flag. The test then attempts to DELETE the newly inserted row from the referencing table. If the student's database implements the constraints, this deletion activates the AFTER DELETE trigger (detailed in Section 3.3), which detects the orphaned entity and executes an ABORT command. The execution sequence structure appears in Listing 4.

Listing 4. Dynamic SQL transaction sequence for validating total participation constraints via trigger activation.

```
PRAGMA foreign_keys=0;
UPDATE system_flags SET validation_enabled = 0;
SAVEPOINT temp_insert;
INSERT INTO entity_table (cols...) VALUES (vals...);
INSERT INTO referencing_table (cols...) VALUES (vals...);
RELEASE SAVEPOINT temp_insert;
UPDATE system_flags SET validation_enabled = 1;
DELETE FROM referencing_table WHERE fk_column = value;
```

To compute the student's grade, the generated test suite executes against the instructor's database and the student's database. Because the schema-matching engine (Section 3.4) aligns the student's database structure with the instructor's naming conventions before instantiation, the framework executes the unmodified DML statements directly against both databases. The system evaluates the student through functional parity. If the student's database commits the valid insertion tests and aborts during the total participation deletion tests, matching the behavior of the instructor's database, the test passes. The system records any divergence in execution outcomes as a semantic flaw in the diagnostic report.

4.2 Grading Methodology

To ensure the automated assessment accurately reflects traditional academic evaluation, the framework maps each SQL test category directly to specific conceptual modeling competencies. Rather than arbitrarily assigning scores, the system simulates a human instructor's grading rubric by distributing points across fundamental ER components based on the outcomes of the dynamic tests. Positive insertion tests serve as the baseline verification for entity conceptualization. If a table successfully accepts the generated DML INSERT statements without structural failure, it demonstrates that the student correctly identified and instantiated the required entity from the business scenario, allowing the system to award points for basic table and attribute existence.

To assess the student's understanding of entity identity, primary key tests are executed utilizing SQLite's PRAGMA table_info. Beyond verifying basic identifiers, these tests act as an implicit validation mechanism for weak entities. By verifying that a table's primary key correctly incorporates a foreign key from an owning entity, the framework can objectively award points for the proper identification and modeling of weak entity types. Building upon this, foreign key tests using PRAGMA foreign_key_list validate the structural implementation of relationships. When analyzed alongside the primary key metadata, these tests determine whether the student correctly modeled relationship cardinality. For example, the framework verifies whether a 1:N relationship is correctly implemented as a foreign key stored in the table from the many side, or whether an N:M relationship is appropriately resolved into an associative table with a composite primary key, allocating points based on the accurate resolution of these connections.

Finally, the transactional trigger tests evaluate advanced behavioral constraints, specifically total participation. By attempting to delete a referencing record and capturing the database engine's ABORT signal, the framework awards points exclusively to students who successfully enforced mandatory participation rules. By assigning weighted point values to these specific execution outcomes, the automated system aggregates a final score. This methodology ensures that the automated grade represents a comprehensive evaluation of entities, attributes, identifiers, relationships, and semantic constraints, closely mirroring the granular point distribution an instructor applies during manual grading.

4.3 Empirical Evaluation and Results

To conduct an evaluation, the system processes a dataset of 50 historical student submissions from a university database design course. To establish a proof of concept, this dataset is composed of five distinct modeling exercises, each containing 10 corresponding student solutions. Addressing the lack of publicly available benchmarks in the domain of conceptual model assessment, this dataset has been anonymized and made publicly available [15]. The text of the modeling exercises is provided to students in a natural language format. Based on years of pedagogical experience, these textual descriptions have been iteratively refined to be highly precise, effectively eliminating structural and semantic ambiguities. Furthermore, it should be emphasized that while the public cloud version of draw.io is utilized in this evaluation to generate the model files, the framework is fundamentally tool-agnostic. Any public or on-premise diagramming tool capable of formally describing diagram semantics and exporting them to XML (or a similar structured format) can be used, provided that the initial XML-to-IR transformation step is adapted to parse the specific output. It should also be noted that Enhanced ER (EER) constructs are not fully supported in the current iteration and were therefore excluded from this phase of testing. The evaluation compares the automated scores with the manual grades using three metrics: mean absolute error (MAE), directional bias, and statistical correlation.

The analysis reveals an MAE of 11.70 points between the automated scores and the manual grades. An examination of the directional bias, illustrated in Fig. 2, indicates a mean error of -11.23 points; the automated framework assigns lower scores than human graders in 94% of the tested cases (47 out of 50 submissions). A manual review of the automated test logs confirms that the framework operates as intended and makes no technical errors in evaluating the SQL constraints. Instead, the scoring variance stems from differences in grading methodologies.

First, human graders award partial credit for partially correct structures. For example, if a student models a relationship but defines an incorrect cardinality, the professor assigns a fraction of the allocated points. The automated framework evaluates functional parity. Each generated SQL test results in a binary pass or fail. If a student implements a constraint incorrectly, the database fails the execution test and receives zero points for that atomic rule. This binary evaluation model creates a systematic negative bias in the automated scores.

Second, the framework and the human graders allocate points differently across schema components. The framework generates specific tests for structural metadata, awarding independent points for primary key designations based on PRAGMA table_info queries. Human graders evaluate primary keys implicitly as part of the overall entity definition without allocating isolated points to them.

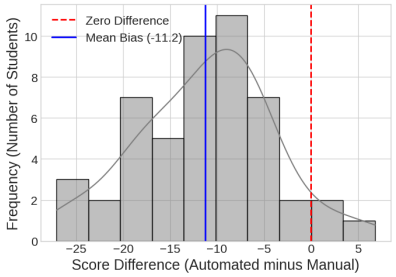


Fig. 2. Distribution of scoring differences (automated score minus manual score)

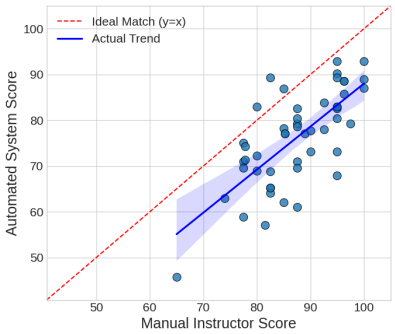


Fig. 3. Scatter plot comparing the automated scores with instructor grades

Despite the difference in absolute scores, statistical correlation measures how well the automated system ranks student performance relative to the human expert. Fig. 3 displays the relationship between the automated scores and the manual instructor grades, contrasting the actual trend against an ideal 1:1 match. The analysis yields a Pearson correlation coefficient of $r = 0.714$ ($p < 0.001$) and a Spearman rank correlation of $\rho = 0.676$ ($p < 0.001$). These metrics indicate a positive correlation between the automated assessment and the manual grading. The statistical significance of these results ($p < 0.001$) demonstrates that the framework ranks student schemas similarly to the human expert, despite stricter binary penalties.

To simulate the partial credit logic of human graders and reduce the absolute scoring gap, future development includes grouping atomic execution tests into logical expressions using AND and OR operators and extending the testing framework to evaluate constructs within Enhanced ER (EER) diagrams.

5 Conclusion and Future Work

This paper presents an automated, execution-based framework for assessing conceptual ER models. Addressing the critical research gaps identified in existing literature, our approach successfully shifts the evaluation paradigm from static, syntactic comparisons to dynamic, functional validation.

First, we overcame the reliance on static analysis and the absence of dynamic testing for conceptual models by automatically generating and executing Data Manipulation Language (DML) test suites. This ensures that a student's design physically enforces the intended business rules rather than merely resembling a reference graph. Second, we addressed the incomplete translation of semantic constraints by extending an IR pipeline to extract and enforce total participation through active database triggers, creating a fully testable physical environment. Finally, to eliminate the system's vulnerability to semantic heterogeneity, we integrated a hybrid schema-matching engine. By formalizing semantic alignment as a bipartite optimization problem, the framework accurately aligns divergent student naming conventions with the instructor's reference solution, ensuring that structurally correct models are not penalized for vocabulary differences.

Our preliminary empirical evaluation of historical university exams shows that automated, execution-based scores correlate statistically with manual instructor grades, demonstrating that the framework consistently ranks student schemas with human experts. However, the analysis also revealed a directional bias in absolute scores. This bias occurs because the system applies strict binary evaluations for each functional test, whereas human graders award partial credit for partially correct structures.

Future work will focus on refining the grading logic to better align absolute scores with human evaluation methods. To simulate the partial credit logic of human graders, future development will group atomic execution tests into logical expressions using AND and OR operators. This modification will allow the framework to evaluate interconnected constraints flexibly and award partial points for partially correct relationship definitions. Furthermore, the next version of the framework will be extended to recognize valid structural variations of the exact solution. For instance, if a student accurately models an N:M relationship by explicitly drawing an associative entity rather than a standard relationship, the system will apply a graded accuracy scale to recognize this equivalence and award appropriate credit. Crucially, we also plan to implement a pedagogical feedback mechanism that translates failed execution tests into natural language diagnostic reports, explicitly guiding students on what is structurally or semantically incorrect in their designs. Additionally, future research will expand the empirical evaluation to a larger dataset of student submissions, adapt the DDL

generation pipeline to support additional relational database management systems, and extend the testing framework to fully support and evaluate advanced EER diagram constructs.

References

1. Chen, P.P.-S.: The entity-relationship model-toward a unified view of data. ACM Transactions on Database Systems (TODS) **1**(1), 9–36 (1976)
2. Todorovikj, M., Velinov, G.: An Automated Transformation Pipeline from ER Diagrams to DDL. Submitted for publication (2026)
3. Nayak, S., Agarwal, R., Khatri, S.K.: Review of Automated Assessment Tools for grading student SQL queries. In: 2022 International Conference on Computer Communication and Informatics (ICCCI), pp. 1–4. IEEE (2022)
4. Ullrich, M., Houy, C., Stottrop, T., Striewe, M., Willems, B., Fettke, P., Loos, P., Oberweis, A.: Automated Assessment of Conceptual Models in Education: A Systematic Literature Review. Enterprise Modelling and Information Systems Architectures (EMISAJ) **18**(2) (2023)
5. Tanaka, T., Hashiura, H., Mouri, K., Hazeyama, A., Kaneko, K.: An automated evaluation method of conceptual data models considering similarities of attribute sets. In: 2018 Seventh ICT International Student Project Conference (ICT-ISPC), pp. 1–5. IEEE (2018)
6. Thomas, P., Waugh, K., Smith, N.: Experiments in the automatic marking of ER-diagrams. In: Proceedings of the 10th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE), pp. 1–6. ACM (2005)
7. Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems. 7th edn. Pearson (2015)
8. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal **10**(4), 334–350 (2001)
9. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1–2), 83–97 (1955)
10. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3982–3992 (2019)
11. Paiva, J.C., Leal, J.P., Figueira, A.: Automated assessment in computer science education: A state-of-the-art review. ACM Transactions on Computing Education (TOCE) **22**(3), 1–40 (2022)
12. Wanjiru, B., van Bommel, P., Hiemstra, D.: Clause-Driven Automated Grading of SQL's DDL and DML Statements. In: Proceedings of the 57th ACM Technical Symposium on Computer Science Education V.1, pp. 1117–1123 (2026)
13. Waugh, K., Thomas, P., Smith, N.: Toward the automated assessment of entity-relationship diagrams. In: Proceedings of the Second Workshop of the Learning and Teaching Support Network - Information and Computer Science (LTSN-ICS), Teaching, Learning and Assessment of Databases (TLAD) (2004)
14. Thomas, P., Waugh, K., Smith, N.: Using patterns in the automatic marking of ER-diagrams. In: Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, pp. 83–87 (2006)
15. Todorovikj, M., Velinov, G.: Benchmark dataset for automated assessment of ER models. GitHub repository. `https://github.com/MilanTodorovikj/grading_dataset` (2026)

Comparative Study of Explainable Intrusion Detection Models using SHAP and LIME

Solomon Owusu Kobiri[1], Kwabena Opoku Frempong-Kore[2][0009−0009−8113−4812], and Mary Dufie Afrane[3][0009−0001−8057−7314]

[1] University of Ghana, Accra, Ghana
sokobiri@st.ug.edu.gh
[2] University of Illinois, Springfield, USA
kfrem@uis.edu
[3] Georgia Southern University, Statesboro, USA
ma20542@georgiasouthern.edu

Abstract. The increasing complexity of network environments has intensified the need for reliable intrusion detection systems (IDS) capable of identifying malicious activity in large volumes of network traffic. This study evaluates post-hoc interpretability techniques applied to machine learning (ML) models for intrusion detection by comparing Logistic Regression, Extreme Gradient Boosting (XGBoost), and a Deep Neural Network (DNN) on the CIC-UNSW-NB15 dataset, which contains over 447,000 network flow records and 76 traffic features. Model performance was assessed using accuracy, precision, recall, F1-score and specificity. All models achieved accuracies above 97%, with XGBoost providing the best overall balance between detection performance and false alarm reduction, while the DNN achieved the highest recall (0.999), missing only 15 out of 17,917 attack instances. SHapley Additive exPlanations (SHAP) and Local Interpretable Model-Agnostic Explanations (LIME) were applied to provide both global feature importance and local prediction explanations. Notably, *Fwd Seg Size Min* and *Bwd Packets/s* emerged consistently as the two most influential features across all three models, suggesting that they are robust indicators of malicious traffic regardless of the learning paradigm used. These findings demonstrate that combining high-performance ML models with post-hoc interpretability analysis can enhance both the effectiveness and transparency of IDS.

Keywords: Intrusion Detection Systems · Machine Learning · Explainable Artificial Intelligence · SHAP · LIME · Network Security · XGBoost · Deep Neural Networks.

1 Introduction

The rapid growth of digital communication and interconnected systems has significantly increased the volume and complexity of network traffic, creating new

challenges for cybersecurity. Among the most widely used defensive technologies are intrusion detection systems (IDS), which monitor network activity to identify malicious behavior and potential security breaches. As cyberattacks continue to evolve in sophistication, traditional rule-based detection approaches have become increasingly insufficient, motivating the adoption of machine learning (ML) and artificial intelligence (AI) techniques capable of identifying complex attack patterns within large volumes of data.

ML-based IDS models have demonstrated strong performance in detecting various types of network intrusions by learning patterns directly from traffic data. Algorithms such as ensemble methods, deep neural networks (DNNs), and hybrid learning frameworks have been widely explored to improve detection accuracy and scalability. However, many high-performing models operate as opaque systems whose internal decision-making processes are difficult to interpret, limiting their practical use in security operations. Existing studies often focus on individual algorithms or isolated explainability techniques, leaving a gap in understanding how different learning paradigms compare in both predictive performance and interpretability. This lack of transparency presents a significant challenge for security practitioners, who must understand why a system flags certain network flows as malicious in order to build trust, diagnose model behavior, and respond effectively to emerging threats.

In response to this challenge, recent research has increasingly focused on explainable AI (XAI) techniques that provide insights into how ML models make predictions. Methods such as SHapley Additive exPlanations (SHAP) and Local Interpretable Model-Agnostic Explanations (LIME) enable both global and local interpretations of model decisions by quantifying the influence of individual features. Integrating these explainability tools into intrusion detection models can enhance transparency while preserving predictive performance.

This study investigates the performance and post-hoc interpretability of multiple ML approaches for network intrusion detection. Three models were selected to represent fundamentally different learning paradigms: Logistic Regression (LR) as a linear baseline, Extreme Gradient Boosting (XGBoost) as a powerful ensemble tree-based method known to perform well on structured tabular data, and a DNN to capture complex nonlinear relationships. SHAP and LIME were chosen as complementary explainability tools because SHAP provides global insights into feature importance across all predictions, while LIME explains individual predictions locally. Together, they offer a more complete picture of model behavior than either method alone. While the terms explainability and interpretability are sometimes used interchangeably in the literature, this study applies post-hoc interpretability techniques that analyze model behavior after training, rather than designing inherently transparent models. Key contributions of this study include:

– A comparative evaluation of three ML paradigms, LR, XGBoost, and a DNN, for intrusion detection using the CIC-UNSW-NB15 dataset.

– The integration of SHAP and LIME as complementary post-hoc explainability techniques to analyze both global feature importance and local decision behavior of the trained models.
– An interpretable analysis of network traffic characteristics that influence malicious activity detection, improving transparency and trust in ML–based IDS.

The remainder of this paper is organized as follows. Section 2 reviews related literature. Section 3 details the dataset, preprocessing steps, and modeling approach. Section 4 presents the experimental evaluation and discusses the results. Section 5 concludes the study and highlights avenues for future research.

2 Literature Review

Intrusion detection remains a fundamental component of contemporary cybersecurity, and extensive research has explored how ML and AI techniques can enhance detection accuracy, scalability, and system reliability. Araújo et al. [3] demonstrated that stacking ensemble models outperform individual classifiers across datasets such as CIC-UNSW-NB15 and HIKARI-2021, achieving F1-scores of up to 95.56%. Similarly, Sharma and Shah [17] achieved 98.98% accuracy and a 99.9% F1-score on the CICIDS2018 dataset using a hybrid approach that combines Random Forest and Support Vector Machine with SMOTE-Tomek resampling. Wardana et al. [18] proposed the CoAt-Set dataset to analyze coordinated attacks, although its synthetic generation process may introduce potential biases. These studies collectively highlight that classical ML techniques remain highly effective, particularly when combined through ensemble or hybrid frameworks.

Beyond improving predictive performance, several studies have explored IDS capable of operating in real-time and cloud environments. Guo et al. [7] developed an ensemble-based IDS that achieved 98% detection accuracy while maintaining a response latency of approximately 48 milliseconds. Fu [6] introduced a layered hybrid framework that integrates DNNs with Random Forest classifiers, reaching 98.5% accuracy while improving resilience to noisy data. Kaushik et al. [8] further demonstrated the practical implementation of IDS through containerized ML services trained on the NSL-KDD dataset. Similarly, Mohamed et al. [11] evaluated both ML and deep learning approaches on the UNSW-NB15 dataset, reporting accuracy levels up to 98.3%. However, their findings also emphasize ongoing challenges, such as class imbalance and susceptibility to adversarial manipulation. Together, these works suggest that real-time IDS deployment is becoming increasingly feasible, although maintaining operational robustness remains an open research issue.

The scarcity of minority attack samples continues to affect detection quality. Long et al. [10] addressed this problem by proposing the BOA-ACRF framework, which combines an Auxiliary Classifier Generative Adversarial Network with a Bayesian-optimized Random Forest to generate balanced training data and achieve macro-F1 scores up to 99.4%. Laghrissi et al. [9] integrated Principal

Component Analysis with an attention-based Long Short-Term Memory (LSTM) network on the NSL-KDD dataset, obtaining a binary classification accuracy of 99.09%. In contrast, Shanthi and Maruthi [16] showed that anomaly-based methods such as Isolation Forest can still achieve detection accuracy around 99% even without explicit techniques to mitigate imbalance. These results indicate that although oversampling and hybrid learning approaches significantly improve detection capability, class imbalance continues to hinder consistent performance in real-world settings.

Recent research has also emphasized the importance of transparency and robustness in IDS models, rather than focusing solely on detection accuracy. Alabbadi and Bajabe [2] proposed an interpretable ensemble model that combines LSTM base learners with a Random Forest meta-learner, achieving 97.05% accuracy on the NSL-KDD dataset. Mohammadian et al. [12] investigated the impact of adversarial attacks using a unified framework that evaluates poisoning and evasion scenarios, revealing vulnerabilities that vary depending on the dataset used. Similarly, Ables et al. [1] developed explainable IDS frameworks based on self-organizing maps and competitive learning, which provide intuitive visual representations while maintaining strong detection effectiveness. These contributions demonstrate that the reliability of IDS solutions increasingly depends on both interpretability and resilience to adversarial threats.

The growing interest in XAI has further encouraged the use of interpretability tools such as SHAP and LIME in intrusion detection research. Arreche et al. [4] proposed XAI-IDS, which applies SHAP and LIME to multiple black-box models across different datasets to generate both global and instance-level explanations with minimal computational overhead. Haripriya and Prabhudev Jagadeesh [14] combined deep learning with hyperparameter optimization and SHAP/LIME analysis, achieving 98.6% accuracy on the CSE-CIC-IDS2018 dataset. Corea et al. [5] investigated model robustness using occlusion sensitivity on the UNSW-NB15 dataset and found that Random Forest predictions remained relatively stable even when certain features were masked. In addition, Ogunseyi and Thiyagarajan [15] enhanced interpretability for Internet of Things (IoT) intrusion detection by employing an optimized LSTM model with hybrid feature selection and SHAP/LIME explanations, reaching an accuracy of 98.4%. These findings suggest that XAI techniques can provide meaningful interpretability without significantly reducing detection performance.

IDS designed for IoT and edge computing environments must also satisfy constraints related to computational efficiency and deployability. Yagiz and Goktas [20] introduced LENS-XAI, which combines knowledge distillation, variational autoencoders, and attribution-based explanation techniques to produce lightweight models capable of achieving up to 99.92% accuracy. Wei et al. [19] proposed XNIDS, a framework that uses sparse group lasso to generate stable explanations and translate them into actionable security rules that can be implemented through mechanisms such as iptables and OpenFlow. These approaches demonstrate that it is possible to design IDS solutions that are both efficient

and interpretable, making them suitable for deployment in resource-constrained environments.

Although prior studies have made significant progress in improving intrusion detection accuracy and developing explainable models, several limitations remain. Many works focus primarily on optimizing a single ML architecture or applying explainability techniques to a specific model. As a result, there is limited understanding of how different learning paradigms compare in terms of both detection performance and interpretability when evaluated under the same experimental conditions. In addition, relatively few studies combine complementary explainability approaches to provide both global insights into feature importance and local explanations for individual predictions. Addressing these limitations is essential for developing IDS that are not only accurate but also transparent and trustworthy for practical cybersecurity applications.

3 Methodology

This study utilizes the CIC-UNSW-NB15 dataset, a modern benchmark for network intrusion detection that contains both normal and malicious traffic and is sourced from the Canadian Institute for Cybersecurity [13]. The dataset comprises 447,915 network flow records and 76 features that capture temporal, size-based, and protocol-specific characteristics of network behavior. Figure 1 shows the distribution of selected traffic features, indicating skewness and variation across key packet metrics; this observation motivates the standardization step applied during preprocessing.

3.1 Data Preparation

The data preprocessing pipeline, shown in Figure 2, consists of three sequential steps applied before model training and evaluation: label binarization, train/test splitting, and feature standardization.

The original dataset contains multiple attack categories and was converted into a binary classification task distinguishing between benign and malicious traffic. All non-zero labels were mapped to 1 as malicious, while 0 represents benign traffic. The transformation is defined as

$$y' = \begin{cases} 0 & \text{if } y = 0 \\ 1 & \text{if } y \neq 0 \end{cases} \tag{1}$$

To ensure balanced class representation in both the training and testing sets, stratified sampling was applied. The dataset was divided using an 80/20 split while preserving the original class distribution.

Feature values were normalized using the StandardScaler method to ensure comparable scales across variables. The standardization process is defined as

$$x' = \frac{x - \mu}{\sigma} \tag{2}$$

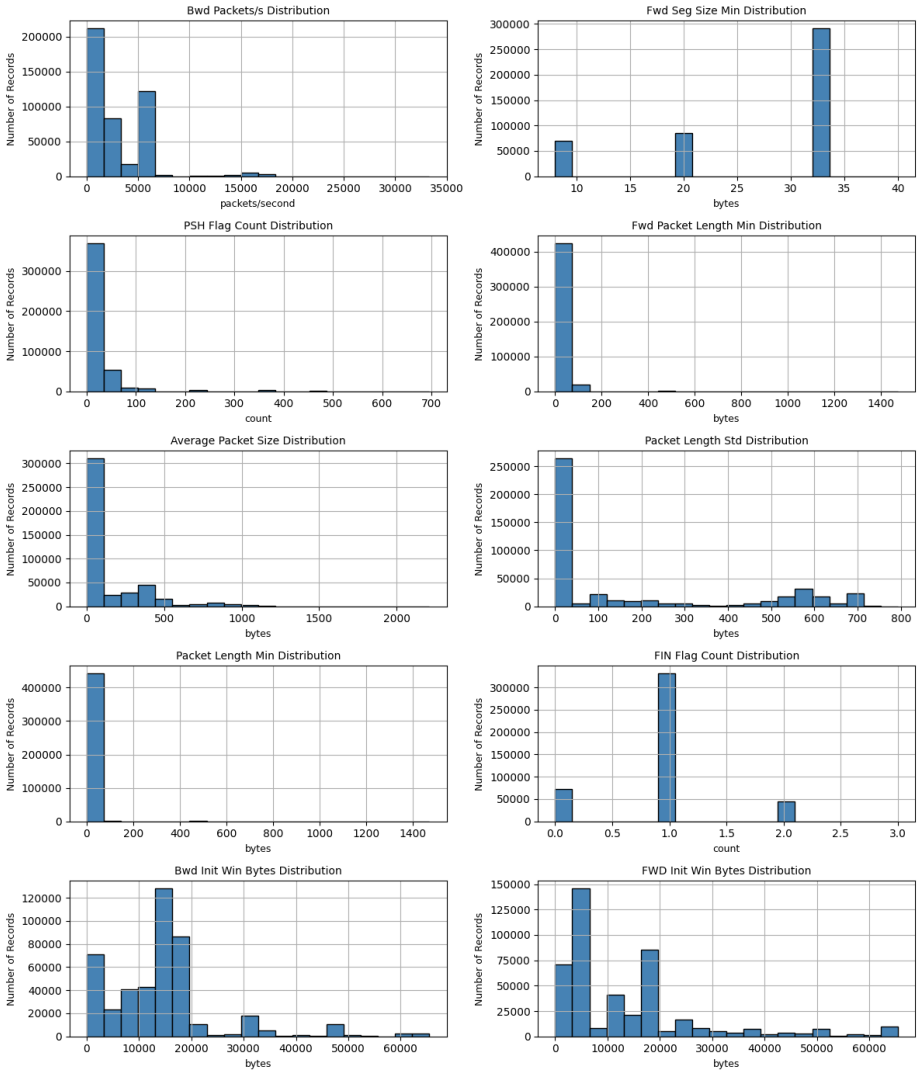


Fig. 1. Distribution of selected network traffic features in the dataset. The x-axis units reflect the measurement scale of each feature (bytes, packets per second, or count), and the y-axis represents the number of records.

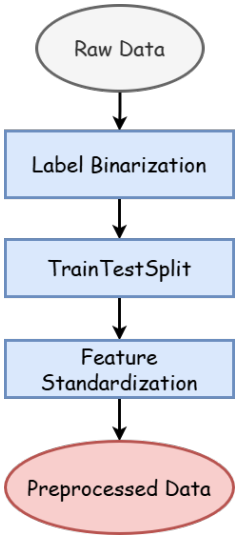


Fig. 2. Data preprocessing pipeline.

where μ represents the mean and σ represents the standard deviation computed from the training data.

3.2 Model Architectures

To evaluate the effectiveness of different ML paradigms for intrusion detection, three models were implemented: LR, XGBoost, and a DNN. These models represent linear, ensemble-based, and deep learning approaches, enabling a comparative assessment of detection performance and interpretability.

LR LR was used as a baseline linear classifier due to its simplicity, efficiency, and interpretability. The model estimates the probability of malicious traffic using the logistic (sigmoid) function applied to a linear combination of input features. Regularization was incorporated to reduce overfitting and improve generalization. The objective function minimized during training is defined as

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log\left(1-h_\theta(x^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{3}$$

Here, m denotes the number of training samples, $x^{(i)}$ represents the input feature vector for the i-th training example, $y^{(i)}$ is the corresponding true binary label, $h_\theta(x)$ denotes the predicted probability given the model parameters θ, and λ controls the strength of regularization.

XGBoost XGBoost was implemented as a powerful ensemble-based learning method well suited for structured tabular data. The algorithm constructs an additive sequence of decision trees, where each new tree is trained to correct the residual errors of the previous ensemble using gradient boosting optimization. XGBoost also incorporates regularization, tree pruning, and parallel computation, which contribute to improved predictive performance and robustness when applied to network traffic datasets. The model was trained using the default XGBoost classifier configuration, with $n\_estimators = 100$, $max\_depth = 6$, learning rate = 0.3, and subsample ratio = 1.0. No hyperparameter tuning was performed, as the default configuration produced strong results on this dataset without additional optimization.

DNN To capture complex nonlinear relationships among network traffic features, a DNN architecture was developed. The network consists of an input layer with 76 features, followed by three fully connected hidden layers containing 64, 128, and 32 neurons, respectively. Rectified Linear Unit (ReLU) and hyperbolic tangent (Tanh) activation functions were used in the hidden layers to enhance representation learning, while a sigmoid activation function was applied in the output layer to produce binary classification probabilities. The network was trained using the Adam optimizer with binary cross-entropy loss for five epochs, a batch size of 32, and a validation split of 20%.

3.3 Model Explainability

To improve transparency in the intrusion detection models, two complementary explainability techniques were applied. SHAP was used to assess feature importance at both global and instance levels, while LIME was used to examine individual classifications. Together, these methods clarify how network traffic features influence model decisions.

SHAP is an explainability method based on principles from cooperative game theory that assigns each feature a contribution value representing its influence on the model's prediction. In the context of intrusion detection, SHAP helps identify which network traffic attributes most strongly influence the classification of a flow as malicious or benign. To ensure efficient computation across the different models used in this study, model-specific explainers were employed. The *LinearExplainer* was applied to LR due to its linear structure, the *TreeExplainer* was used for XGBoost to leverage its tree-based architecture, and the *KernelExplainer* was used for the DNN with 100 background samples to approximate feature contributions. The SHAP value for feature i is defined as

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f(S \cup \{i\}) - f(S) \right] \tag{4}$$

where F represents the set of all features, S denotes a subset of features excluding i, and $f(S)$ represents the model prediction when only the features in

subset S are considered. This formulation estimates the marginal contribution of each feature across all possible feature combinations.

LIME provides explanations for individual predictions by approximating the behavior of a complex model with a simpler, interpretable model in the vicinity of a specific data instance. For this study, LIME enables the analysis of why a particular network flow was classified as malicious or benign by highlighting the most influential features in that local prediction. This is achieved by generating perturbed samples around the instance of interest and fitting a weighted, interpretable model that approximates the original model's behavior within that local region. The explanation is obtained by solving the following optimization problem:

$$\xi(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{5}$$

where f represents the original predictive model, g denotes the interpretable surrogate model, and π_x defines the locality around the instance being explained. The function L measures how closely the surrogate model approximates the original model within this local region, while $\Omega(g)$ penalizes model complexity to ensure interpretability.

4 Results and Discussion

Five key performance metrics were used to evaluate the intrusion detection models: accuracy, precision, recall, F1-score, and specificity. The evaluated models are then compared based on their predictive performance, followed by an analysis of feature importance with SHAP and instance-level explanations with LIME. These analyses provide insight into both the effectiveness of the models and the factors influencing their predictions.

4.1 Performance Comparison of Intrusion Detection Models

Table 1 presents the comparative performance of the three models evaluated in this study. LR demonstrated competitive performance despite its relatively simple linear structure. The model achieved an accuracy of 0.977 and a recall of 0.996, correctly identifying 17,841 out of 17,917 attack instances while missing only 76. Its lower precision of 0.900 and specificity of 0.972 reflect a tendency to be more aggressive in flagging traffic as malicious, resulting in 1,977 false alarms. In operational IDS environments, this trade-off between recall and precision is well recognized; maximizing attack detection often comes at the cost of increased false alerts. The corresponding confusion matrix is shown in Figure 3.

XGBoost achieved the best results, with an accuracy of 0.983, an F1-score of 0.959, and the highest specificity of 0.981, indicating that it produced the fewest false alarms among the three models. This outcome aligns with the effectiveness of gradient boosting algorithms on structured tabular data, such as network flow records. XGBoost's ensemble of regularized decision trees likely contributed

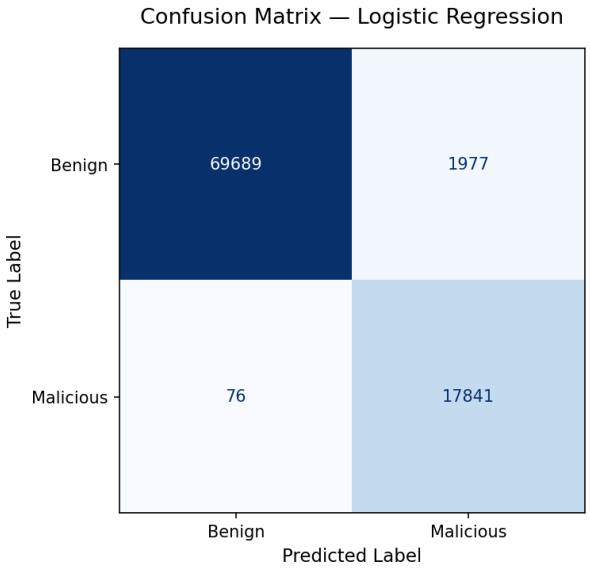


Fig. 3. LR Confusion Matrix.

to its ability to capture complex feature interactions. The confusion matrix is presented in Figure 4.

The DNN produced the most notable individual result compared to LR and XGBoost, achieving a recall of 0.999 and missing only 15 out of 17,917 attack instances in the test set. This near-perfect attack detection rate demonstrates the neural network's ability to capture subtle nonlinear patterns associated with malicious traffic. The model achieved an accuracy of 0.979, an F1-score of 0.950, and a specificity of 0.974. Although the DNN's overall F1-score was slightly lower than that of XGBoost, its exceptional recall makes it the preferred choice in security-critical deployments where missing an attack carries greater risk than raising occasional false alarms. The corresponding DNN confusion matrix is shown in Figure 5.

Overall, the results confirm that all three models provide strong classification performance for intrusion detection, with accuracy values exceeding 97%. The consistently high accuracy across the models is attributable in part to the characteristics of the CIC-UNSW-NB15 dataset, which is preprocessed and relatively balanced, with approximately 80% benign and 20% malicious traffic.

4.2 SHAP Explainability

Figure 6 presents the SHAP feature importance for the LR model. *Bwd Packets/s* is the dominant feature, with a mean absolute SHAP value of 12.59, nearly four times larger than that of the second-ranked feature. This indicates that variations in backward packet transmission rates are the strongest signal this

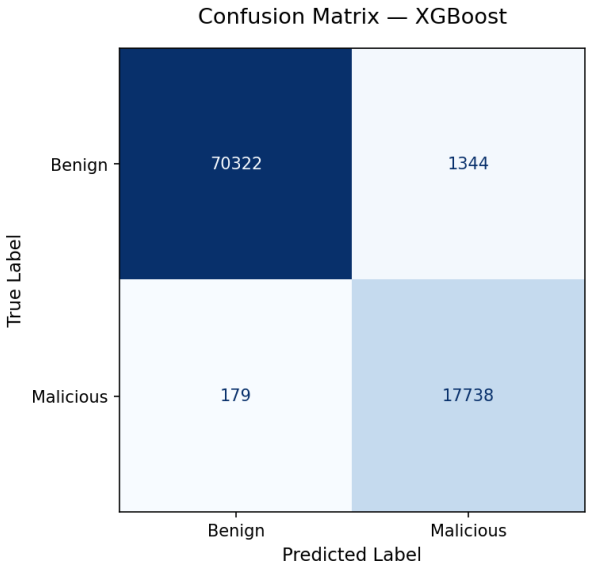


Fig. 4. XGBoost Confusion Matrix.

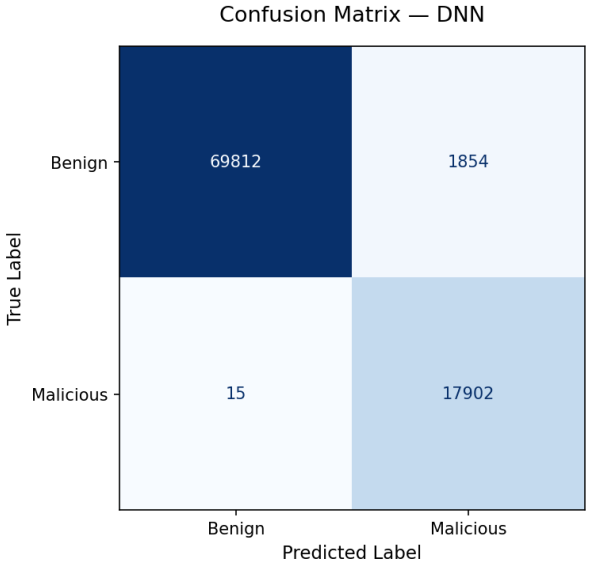


Fig. 5. DNN Confusion Matrix.

Table 1. Model Performance Comparison.

Model	Accuracy	Precision	Recall	F1-Score	Specificity
LR	0.977	0.900	0.996	0.946	0.972
XGBoost	0.983	0.930	0.990	0.959	0.981
DNN	0.979	0.906	0.999	0.950	0.974

model uses to distinguish malicious from benign traffic. Other important features include *Fwd Seg Size Min* and *PSH Flag Count*, which capture characteristics of packet segmentation and control flags that may reflect unusual communication patterns associated with malicious activity. Features such as *Fwd Packet Length Min*, *Packet Length Std*, and *Flow Duration* exhibit moderate influence, while *Average Packet Size* and *FIN Flag Count* contribute relatively little to the final prediction.

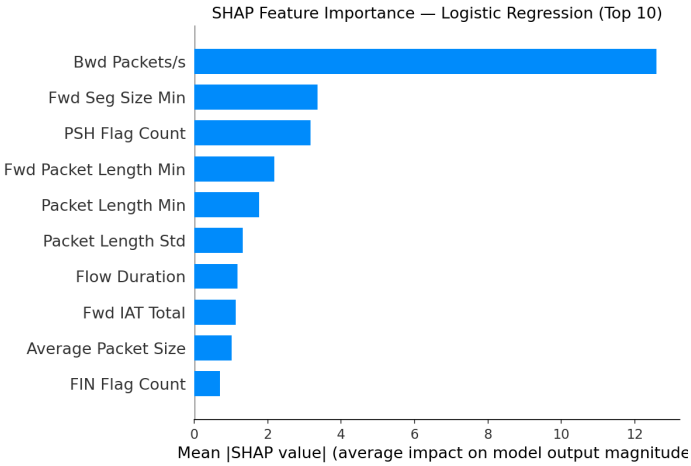


Fig. 6. SHAP feature importance for the LR model.

Figure 7 presents the SHAP feature importance for the XGBoost model. *Fwd Seg Size Min* emerges as the dominant feature, with a mean absolute SHAP value of 6.29, nearly three times larger than that of the second-ranked feature. This indicates that variations in minimum forward segment size are the primary signal XGBoost uses to identify malicious traffic. *Bwd Packets/s* ranks second, consistent with its top position in the LR analysis. Additional features, including *Bwd Packet Length Min*, *Flow Duration*, and several inter-arrival timing features (*Fwd IAT Mean*, *Fwd IAT Max*, *Flow IAT Max*), contribute supplementary information about packet structure and temporal communication patterns. Compared to LR, XGBoost places greater emphasis on flow-level timing features, reflecting

its ability to capture nonlinear relationships among traffic attributes through its ensemble of decision trees.

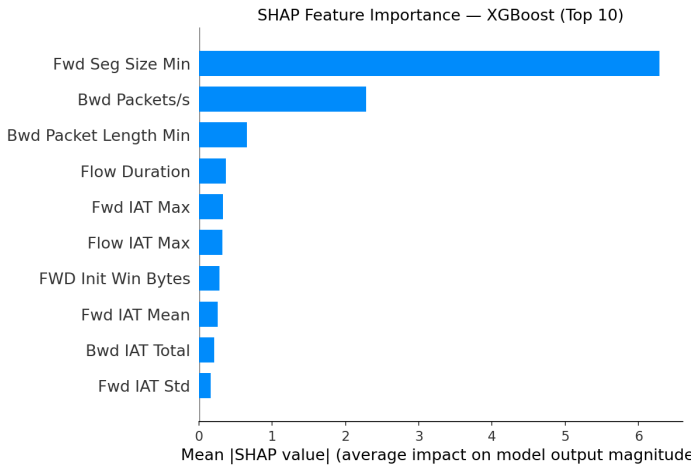


Fig. 7. SHAP feature importance for the XGBoost model.

Figure 8 presents the SHAP feature importance for the DNN model. *Fwd Seg Size Min* again emerges as the most influential feature, consistent with the XGBoost results, further reinforcing its role as a robust indicator of malicious traffic. *Bwd Packets/s* ranks second, maintaining its position among the top two features across all three models. Other contributors include *Bwd Packet Length Min*, *FIN Flag Count*, and *Bwd Init Win Bytes*, which capture packet transmission patterns and connection behavior characteristics. Compared to the Logistic Regression and XGBoost models, the DNN distributes feature importance more evenly across a broader set of variables, reflecting the neural network's ability to capture complex nonlinear interactions among multiple traffic characteristics simultaneously.

Across all three models, *Fwd Seg Size Min* and *Bwd Packets/s* consistently appear among the top two most influential features. This cross-model agreement is a notable finding, as it suggests that minimum forward segment size and backward packet transmission rate are robust indicators of malicious network activity, regardless of the underlying learning paradigm. Security analysts and IDS designers may therefore prioritize monitoring these two features as reliable signals of potential attack behavior.

4.3 LIME Explainability for First Data Instance

The LIME explanation for the first test instance, a benign network flow correctly classified by the LR model, is presented in Figure 9. Features shown in green

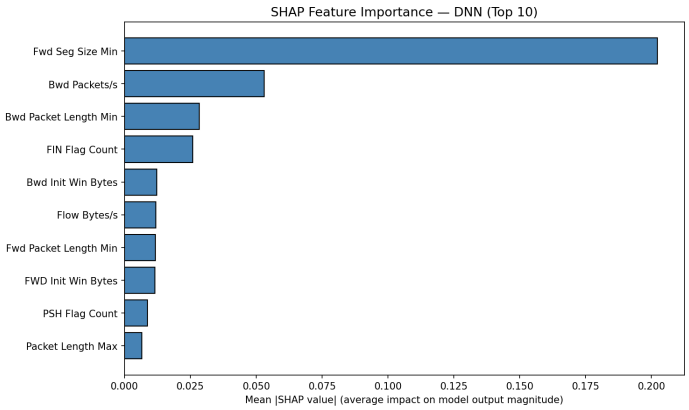


Fig. 8. SHAP feature importance for the DNN model.

push the prediction toward a malicious classification, while red features push it toward a benign classification. The strongest benign contributors are *Idle Max* and *Packet Length Min*, indicating that low idle time and small packet lengths are characteristic of normal traffic in this instance. The strongest malicious push comes from *Bwd Packets/s*, which falls within a range the model associates with suspicious backward transmission rates. However, the combined benign forces outweigh the malicious ones, resulting in a correct benign classification. The prediction reflects a balance between competing packet-level and timing-related features rather than a single dominant signal.

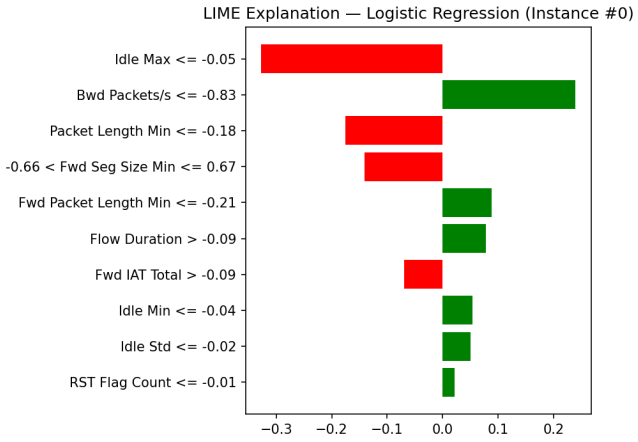


Fig. 9. LIME explanation for the first test instance using LR.

Figure 10 presents the LIME explanation for the same instance using XGBoost. A notable contrast with the LR explanation emerges: *Bwd Packets/s* is the strongest contributor in this instance but pushes toward a *malicious* classification, the opposite direction from the LR explanation, where the same feature pushed toward benign. This demonstrates that different models can assign opposite interpretations to the same feature value for the same instance, yet still arrive at the correct prediction through different reasoning paths. The benign classification is ultimately driven by activity and segment size features, including *Active Std* and *Fwd Seg Size Min*, whose combined influence outweighs the malicious push from *Bwd Packets/s*.

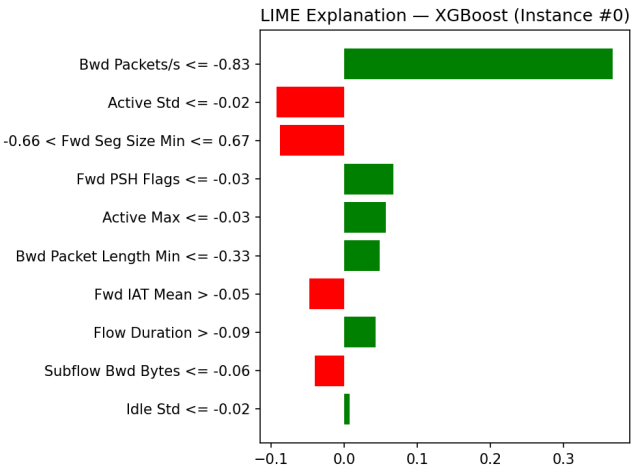


Fig. 10. LIME explanation for the first test instance using XGBoost.

The LIME explanation for the same instance using the DNN is presented in Figure 11. The model predicted a malicious probability of 0.0000, indicating complete certainty in its benign classification and the most confident prediction among the three models for this instance. *Fwd Seg Size Min* is the dominant benign contributor, consistent with its position as the top feature in the DNN's global SHAP analysis. This suggests that the feature plays a decisive role in the DNN's decision-making at both global and local levels. *Fwd PSH Flags* provides the strongest malicious push (+0.48) but is overwhelmed by the combined benign contributions of eight other features. Compared to LR and XGBoost, the DNN's explanation is more one-sided; the benign forces are substantially stronger than the malicious ones, reflecting the model's high confidence in this prediction.

Across all three LIME explanations, each model correctly classified the instance as benign through markedly different reasoning paths. LR relied on a balance of many competing features, XGBoost was dominated by a single large malicious push that was overcome by several smaller benign forces, and the

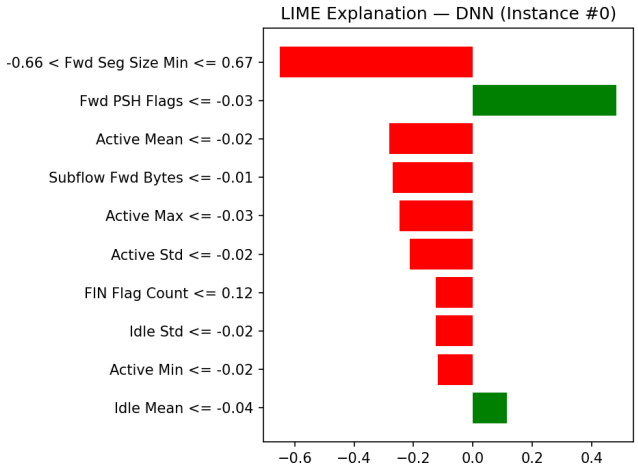


Fig. 11. LIME explanation for the first test instance using the DNN.

DNN produced an overwhelmingly one-sided benign decision. This cross-model comparison illustrates how post-hoc interpretability techniques can reveal fundamentally different internal reasoning processes, even when models reach the same correct conclusion, highlighting the value of applying complementary explainability methods when evaluating IDS models.

5 Conclusions and Future Work

This study presented a comparative analysis of LR, XGBoost, and a DNN for network intrusion detection using the CIC-UNSW-NB15 dataset. All three models achieved strong performance, with accuracy values exceeding 97%. XGBoost provided the best overall balance between detection capability and false alarm reduction, achieving an accuracy of 0.983, an F1-score of 0.959, and the highest specificity of 0.981. The DNN achieved the highest recall of 0.999, missing only 15 out of 17,917 attack instances, while LR remained competitive as a simple and effective linear baseline.

SHAP and LIME were applied as complementary interpretability techniques to analyze model behavior at both global and local levels. SHAP identified *Fwd Seg Size Min* and *Bwd Packets/s* as consistently among the most influential features across all three models, suggesting that they are robust indicators of malicious network activity.The consistency of these features across different model types supports the usefulness of SHAP for identifying robust indicators of malicious network activity. LIME demonstrated that although all three models correctly classified the examined instance, they relied on fundamentally different reasoning processes. Together, these findings highlight the value of combining high-performing ML models with complementary explainability techniques in IDS research.

As with any empirical study, this work has some limitations that should be acknowledged. Because all experiments were conducted on the CIC-UNSW-NB15 dataset, the results may not fully generalize to other network environments, traffic types, or attack distributions. In addition, the use of StandardScaler for feature normalization may suppress extreme outlier values that carry diagnostic importance for certain attack types, such as denial-of-service attacks, where anomalous traffic volumes are a primary indicator. The conversion of multi-class attack labels into a binary classification task also simplifies detection while eliminating the ability to distinguish among different attack categories.

Future research could evaluate the proposed framework on additional benchmark datasets, such as NSL-KDD, CICIDS2018, and UNSW-NB15, to strengthen the generalizability of the findings. Extending the classification task to multi-class intrusion detection would offer more granular insight into model behavior across different threat categories. Additionally, exploring inherently interpretable architectures, such as rule-based classifiers or attention-based neural networks, would complement the post-hoc approach used in this study and support a more comprehensive understanding of transparency in IDS design.

References

1. Ables, J., Kirby, T., Mittal, S., Banicescu, I., Rahimi, S., Anderson, W., Seale, M.: Explainable intrusion detection systems using competitive learning techniques. arXiv preprint arXiv:2303.17387 (2023)
2. Alabbadi, A., Bajaber, F.: An intrusion detection system over the iot data streams using explainable artificial intelligence (xai). Sensors (Basel, Switzerland) **25**(3), 847 (2025)
3. Araújo, A., Rodrigues, D., Leite, P., Gonçalves, J.: Network intrusion detection system based on multiple datasets: Machine learning approaches. In: 2025 13th International Symposium on Digital Forensics and Security (ISDFS). pp. 1–5. IEEE (2025)
4. Arreche, O., Guntur, T., Abdallah, M.: Xai-ids: Toward proposing an explainable artificial intelligence framework for enhancing network intrusion detection systems. Applied Sciences **14**(10), 4170 (2024)
5. Corea, P.M., Liu, Y., Wang, J., Niu, S., Song, H.: Explainable ai for comparative analysis of intrusion detection models. In: 2024 IEEE International Mediterranean Conference on Communications and Networking (MeditCom). pp. 585–590. IEEE (2024)
6. Fu, R.: Design and implementation of network intrusion detection system based on machine learning. In: 2025 International Conference on Intelligent Systems and Computational Networks (ICISCN). pp. 1–6. IEEE (2025)
7. Guo, F., Jiao, H., Zhang, X., Zhou, Y., Feng, H.: Information security network intrusion detection system based on machine learning. In: 2024 International Conference on Data Science and Network Security (ICDSNS). pp. 01–04. IEEE (2024)
8. Kaushik, C., Ram, T., Ritvik, C., Lakshman, T.: Network security with network intrusion detection system using machine learning deployed in a cloud infrastructure. In: 2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC). pp. 701–708. IEEE (2022)

9. Laghrissi, F., Douzi, S., Douzi, K., Hssina, B.: Ids-attention: an efficient algorithm for intrusion detection systems using attention mechanism. Journal of Big Data **8**(1), 149 (2021)
10. Long, H., Li, H., Tang, Z., Zhu, M., Yan, H., Luo, L., Yang, C., Chen, Y., Zhang, J.: Boa-acrf: An intrusion detection method for data imbalance problems. Computers and Electrical Engineering **124**, 110320 (2025)
11. Mohamed, A., Heilala, J., Madonsela, N.S.: Machine learning-based intrusion detection systems for enhancing cybersecurity. In: 2023 Second International Conference On Smart Technologies For Smart Nation (SmartTechCon). pp. 366–370. IEEE (2023)
12. Mohammadian, H., Lashkari, A.H., Ghorbani, A.A.: Poisoning and evasion: Deep learning-based nids under adversarial attacks. In: 2024 21st Annual International Conference on Privacy, Security and Trust (PST). pp. 1–9. IEEE (2024)
13. Mohammadian, H., Habibi Lashkari, A., Ghorbani, A.A.: Cic-unsw-nb15 dataset (2024), `https://www.unb.ca/cic/datasets/cic-unsw-nb15.html`, an augmented version of the UNSW-NB15 dataset generated using CICFlowMeter
14. MP, P.J., et al.: An explainable and optimized network intrusion detection model using deep learning. International Journal of Advanced Computer Science & Applications **15**(1) (2024)
15. Ogunseyi, T.B., Thiyagarajan, G.: An explainable lstm-based intrusion detection system optimized by firefly algorithm for iot networks. Sensors **25**(7), 2288 (2025)
16. Shanthi, K., Maruthi, R.: Machine learning approach for anomaly-based intrusion detection systems using isolation forest model and support vector machine. In: 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA). pp. 136–139. IEEE (2023)
17. Sharma, V., Shah, D.J.: A novel approach to intrusion detection systems using hybrid machine learning techniques. In: 2024 International Conference on Artificial Intelligence and Quantum Computation-Based Sensor Application (ICAIQSA). pp. 1–6. IEEE (2024)
18. Wardana, A.A., Kołaczek, G., Sukarno, P.: Coat-set: Transformed coordinated attack dataset for collaborative intrusion detection simulation. Data in Brief p. 111354 (2025)
19. Wei, F., Li, H., Zhao, Z., Hu, H.: {xNIDS}: Explaining deep learning-based network intrusion detection systems for active intrusion responses. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 4337–4354 (2023)
20. Yagiz, M.A., Goktas, P.: Lens-xai: Redefining lightweight and explainable network security through knowledge distillation and variational autoencoders for scalable intrusion detection in cybersecurity. arXiv preprint arXiv:2501.00790 (2025)

SPAR-HT: A PMem-Optimized Hash Table for Efficient Joins

Sudip Chatterjee[1][0009−0008−4523−1941], Suprio Ray[1][0000−0003−0681−9685], Ian Finlay[2][0009−0009−8686−2448], Calisto Zuzarte[2][0009−0006−1440−6775], and Mark Stoodley[2][0009−0005−2108−8181]

[1] University of New Brunswick, Canada
{sudip.chatterjee,sray}@unb.ca
[2] IBM Canada
{finlay,calisto,mstoodle}@ca.ibm.com

Abstract. Join processing remains a fundamental bottleneck in modern analytical database systems, particularly when hash tables must be repeatedly constructed for recurring join workloads. While most systems build hash tables on the fly, many real-world analytical workloads entail executing joins on the same columns, making them strong candidates for persistent hash indexes. The emergence of byte-addressable persistent memory (PMem) enables database systems to store hash indexes directly in non-volatile memory and reuse them across queries, eliminating repeated build costs while reducing recovery overhead.
In this paper, we investigate persistent memory–resident hash indexes for accelerating join processing in OLAP workloads operating on largely static datasets. We propose `SPAR-HT`: a ***S**calable **P**Mem **A**ware, **R**esource-optimized **H**ash **T**able*, a persistent memory–optimized hash table designed for high-concurrency analytical environments. `SPAR-HT` leverages the read-mostly nature of analytical workloads and adopts a *rebuild-on-significant-change* strategy that simplifies maintenance while achieving near O(1) lookup performance. Through a comprehensive experimental evaluation against state-of-the-art PMem-aware hashing techniques, including CCEH, Level Hashing, and Dash, we demonstrate that `SPAR-HT` achieves up to 2×, 5×, and 13× higher performance compared to Dash, CCEH, and Level Hashing, respectively, under low-selectivity workloads. These results highlight the effectiveness of `SPAR-HT` and demonstrate the potential of PMem-native indexing structures for accelerating join-intensive analytical workloads.

Keywords: Hash table · Persistence memory · Index · Hash Join.

1 Introduction

Hash tables play a central role in Online Analytical Processing (OLAP) systems due to their near constant-time lookup performance, making them particularly effective for join processing and key-based access. In typical hash-join execution, hash tables are constructed on the fly. However, many workloads repeatedly

join on the same columns. Real-world analytical workloads exhibit significant redundancy; for instance, 80% of queries in half of all Amazon Redshift clusters are exact repeats [14]. Therefore, prebuilding and persisting hash tables for these frequently used attributes can substantially reduce join latency. Emerging byte-addressable persistent memory (PMem) technology offers a promising alternative by providing near-DRAM access latency while supporting significantly larger storage capacities. This capability enables database systems to maintain larger indexing structures directly in PMem. Leveraging PMem in this way offers a promising opportunity to improve overall hash-join performance.

Several hash tables have been specifically designed to exploit the characteristics of persistent memory. However, these designs have largely been evaluated in the context of indexing transactional workloads, focusing on insert, lookup, and update performance. Their effectiveness for accelerating join processing on persistent memory has not been systematically studied. In particular, it remains unclear how PMem-optimized hash tables behave under the access patterns and concurrency characteristics typical of hash-join execution. In this work, we address this gap by conducting a comprehensive evaluation of leading persistent memory hash tables under join workloads.

Persistent-memory hash indexing approaches, such as CCEH [11], Level Hashing [19], and Dash [10], use bucket- or segment-based layouts to enable fast point lookups. CCEH relies on a directory-driven extendible hashing scheme, while Level Hashing and Dash scale via two-level buckets or PMem-optimized segments, with Dash additionally supporting NUMA-aware segmented concurrency for highly parallel joins. Beyond their structural differences, modern PMem-aware hash tables can accelerate hash join execution by persisting hash tables across queries, reducing the cost of repeated construction and probe lookups. However, these PMem-aware hash indexes suffer from several limitations.

Level Hashing's performance degrades as the load factor increases because its log-free update mechanism depends on locating vacant slots; as the table becomes full, it falls back to expensive logging, and although it supports in-place scaling, it still incurs significant NVM write overhead. Similarly, CCEH introduces structural complexity through its three-level organization—directory, segments, and buckets, where large segment sizes can lead to high latency during splits due to extensive cacheline flushes. Like other extendible hashing variants, CCEH also requires an additional directory access for each hash table lookup. Although it places buckets within a single cacheline to reduce memory accesses, it still relies on linear probing to resolve collisions. Moreover, these hash tables tend to consume considerable memory while trying to optimize performance. CCEH often exhibits the most unused space because its segment-based splitting may trigger a rehash when only a single bucket overflows, leaving many other buckets within the segment under-utilized. Level Hashing maintains a relatively balanced occupancy but still requires some reserved empty space to operate efficiently. In contrast, Dash minimizes unused space by leveraging stash buckets and fingerprinting, enabling load factors exceeding 90% while maintaining high performance on real persistent memory hardware.

To address the limitations of existing PMem-aware hash indexes we introduce `SPAR-HT`, a PMem-aware hybrid hash table designed to efficiently exploit both persistent memory and modern CPU architectures. It is built on top of CAMEL-HT [3]. Our design places lightweight metadata in DRAM to enable fast access, while storing key–value (or address of the row) pairs in PMem to ensure persistence and reuse. To efficiently support incremental updates, we incorporate a small in-memory hash table that buffers updates before integration with the persistent structure. Furthermore, `SPAR-HT` leverages software prefetching and SIMD instructions to better utilize CPU parallelism and cache locality. Through this design, `SPAR-HT` achieves up to 13.4×, 4.9×, and 2× higher probing performance than Level Hashing, CCEH, and Dash, respectively, when probing 132M records against a 50M build table using 16 threads.

In summary, this paper makes the following contributions:

- **PMem-optimized `SPAR-HT` (section 3):** We present a persistent-memory-optimized hash table, `SPAR-HT`, which follows a "*rebuild-on-significant-change*" strategy, tailored for read-mostly analytical workloads. Furthermore, we also support bulk insertion of new records after initial load.
- **Rehash-free hybrid DRAM–PMem design (section 4):** We propose a hybrid design that leverages the complementary strengths of DRAM and persistent memory (PMem). Our approach eliminates costly rehashing through an elegant design and further exploits modern hardware features, such as software prefetching and SIMD, to improve performance.
- **Comprehensive performance comparison (section 6):** We compare `SPAR-HT` with existing PMem hash tables, including CCEH, Level Hashing, and Dash for join operation on building time, probing time, cache utilization, and memory efficiency in multi-threaded environments on PMem (Intel Optane).

The remainder of this paper is organized as follows. Section 2 provides background. Sections 3 and 4 present the design principles of `SPAR-HT` and describe how it combines the benefits of DRAM and PMem within a unified architecture. Sections 5 and 6 present the experimental evaluation. Section 7 discusses the results, and Section 8 concludes the paper.

2 Background

We start by giving a summary of PMem and its features that are important for data management. Next, we give an overview of hash join and highlight cutting-edge persistent memory optimized hash tables and indexing structures. We then introduce `SPAR-HT`.

2.1 Persistent memory

Intel Optane DC Persistent Memory represents the first commercially available PMem technology that combines high capacity with byte-addressability and

near-DRAM latency. Its unique characteristics have led to increasing adoption in large-scale data management and query processing systems in recent years.

PMems bandwidth is similar to DRAM and high capacity (256/512 GB per DIMM) at a price lower than DRAM. The core of Intel's PMem modules lies the three-dimensional (3D) XPoint memory medium, which is inherently non-volatile and retains data even in the absence of power. Communication with Optane DIMMs is orchestrated through the processor's integrated memory controller (iMC), which employs the DDR-T interface, a 64-byte asynchronous extension of DDR designed for persistent memory [1,17]. The iMC manages separate Read Pending Queues (RPQs) and Write Pending Queues (WPQs) for each Optane DCPMM (Data Center Persistent Memory Module). Crucially, only the WPQs reside within the Asynchronous DRAM Refresh (ADR) domain [7]. The internal module controller on the Optane DIMM manages read and write operations to the persistent media at a 256-byte granularity. Since PMem persists data only after a cache-line flush (CLFLUSH, CLFLUSHOPT, or CLWB) [8] and provides just 8-byte failure atomicity, software must use memory fences and these flush/writeback instructions to prevent partial updates after a crash.

PMem operate on two modes: *Memory mode* and *App direct mode*. In Memory Mode, DRAM serves as a transparent cache for frequently accessed data, while DCPMMs supply a large but volatile memory capacity that is lost on power failure. In App Direct Mode, by contrast, the operating system and applications are explicitly aware of the separate DRAM and PMem regions, enabling software to issue direct load/store operations to DCPMMs and exploit their persistence [7]. Data persistence to PM is achieved when a cache line flush instruction, such as clflush, clflushopt, or clwb, is executed.

PMem has asymmetric read and write latency, with writes being slower. It has a read latency of approximately 300 ns, which is approximately four times that of DRAM. In comparison to DRAM, it has around 3×/8× slower sequential/random read bandwidth. The numbers for sequential/random writing are around 11×/14× [10].

2.2 Hash join

In a typical hash join involving tables R and S, |R| and |S| denote the cardinalities of the build and probe relations, respectively, with $|R| < |S|$. During the build phase, keys from |R| are hashed to locate target slots, where key–value pairs are inserted. In the probe phase, keys from |S| are hashed to identify candidate buckets, and matches are found via targeted search in the linked lists or arrays. Hash join methods are characterized as *hardware-oblivious* [2] or *hardware-conscious* [13], depending on their interaction with system architecture. Hardware-oblivious techniques promote portability and simplicity by using simple designs such as the build-probe hash join [2] (no partition join) that do not explicitly utilize hardware characteristics. In contrast, hardware-conscious techniques optimize architectural features including cache hierarchies, SIMD [6] support, and NUMA [6] effects. Radix partitioning is a good example for improving cache efficiency and memory locality.

2.3 Hash Tables and Indexes

A number of state-of-the-art hash tables and index structures have been specifically designed for persistent memory. In this work, we focus on Level Hash [19], CCEH [11], and Dash [10] as representative PMem-optimized hash tables. Based on their resizing strategies, persistent-memory hash tables can be broadly categorized into two groups: *Level Hashing*, which organizes data across multiple levels to enable cost-efficient incremental resizing, and *Extensible Hashing*, which supports dynamic growth and shrinkage without requiring full rehashing of the dataset [15]. These index structures are fundamentally hardware-aware, as their designs explicitly exploit persistent memory characteristics through write-efficient updates and cache-line–granularity optimizations.

Level Hashing: Level hashing [19] organizes data across two levels: a top level and a bottom level. Items are first inserted into the top level using cuckoo hashing [12], and overflow entries are placed in the bottom level. When the bottom level becomes full, it is resized to twice the size of the current top level, and only bottom entries are rehashed, reducing write amplification on persistent memory. This design, along with carefully ordered, failure-atomic updates, ensures correctness under crashes. After resizing, the expanded region becomes the new top level, while the former top level is demoted to the bottom, allowing the structure to grow incrementally without full-table rehashing.

CCEH: Cacheline-Conscious Extendible Hashing (CCEH) [11] is a PMem-optimized dynamic hash table that extends the classical extendible hashing [5] architecture. Instead of the traditional two-level design, CCEH introduces a three-tier structure consisting of a global directory, segments, and fixed-size buckets. The global directory, indexed by the least significant bits of the hash value, points to segments, which serve as the fundamental units of allocation and splitting. Each segment contains multiple buckets, and these buckets are sized to fit exactly within a single 64-byte CPU cache line. This cacheline-aware layout minimizes cache-line fetches during lookups, thereby improving performance.

Dash: Dash [10] incorporates principles from Extendible Hashing and Linear Hashing [9] while optimizing their behavior for PMem characteristics. Dash (all our experiments use the extendible-hashing variant of Dash; thus, Dash refers to Dash-EH throughout this paper) builds on several design elements from CCEH but adapts the bucket size to match the 256-byte XPLine of Intel Optane DCPMM, improving spatial locality. Each directory entry references a segment composed of 64 regular buckets and two stash buckets for overflow handling, all sharing the same internal layout. Every bucket is divided into a 224-byte record region that holds pointers to variable-length key-value items. The remaining 32-byte metadata region is used to optimize probing and sustain high load factors.

3 SPAR-HT Architecture: Design and Implementation

Many modern analytical systems operate under read-mostly workloads, where datasets remain largely static for long periods. In such scenarios, traditional

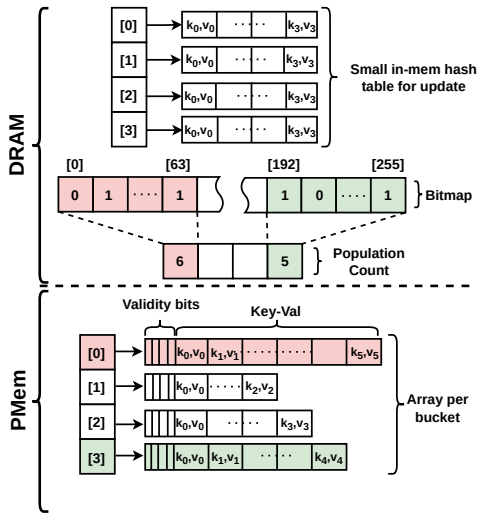


Fig. 1: Detailed Architecture of the SPAR-HT Framework

indexing structures designed for frequent updates often introduce unnecessary overhead. To address the limitations of traditional and dynamic hashing under certain workloads, we propose SPAR-HT, inspired by our previously proposed CAMEL-HT [3]. CAMEL-HT was originally designed as an in-memory hash table for hash join operations without update functionality and is not PMem-aware. In this work, we redesigned the system to leverage persistent memory (PMem) and added support for updates. Our architecture adopts a hybrid layout, storing metadata in DRAM and placing the main hash table in PMem. The proposed design mitigates several drawbacks commonly observed in dynamic PMem hash tables, including excessive pointer chasing during insert and probe operations, frequent memory allocations, and costly rehashing. Notably, it is inherently rehash-free, which simplifies the design and improves predictability for static workloads.

As shown in Figure 1, our design follows a hybrid architecture in which a compact bitmap is maintained in DRAM, while the actual hash table storing the key–value pairs resides in PMem. Since the bitmap is small, keeping it in DRAM enables fast existence checks and reduces expensive PMem accesses during probing.

To construct the hash table, the dimension table is scanned twice during the build phase. The first pass constructs the bitmap and determines the exact number of elements per bucket (*population count*), enabling precise memory allocation. In the second pass, the tuples are inserted into the hash table and persisted to PMem. Since Intel Optane persistent memory operates at a 256-byte access granularity, we adopt a persistence strategy similar to prior designs such as Dash. Specifically, data is persisted in 256-byte aligned segments, which reduces write amplification and improves write efficiency on PMem. Despite performing two iterations over the build data, the elimination of costly rehashing and PMem rewrite operations results in superior performance during the build phase. Additionally, we set validity bits for each item during insertion into PMem to ensure durability and facilitate recovery when needed.

During the probe phase, we further optimize performance by incorporating software prefetching and SIMD-based probing. These hardware-conscious optimizations improve cache utilization and parallelism during lookup operations. Together, these design choices significantly improve both build and probe performance compared to state-of-the-art persistent memory hash tables.

To support updates, we maintain a simpler in-memory data structure that is approximately one-quarter the size of the PMem-resident hash table. Newly arriving records are first inserted into this DRAM-resident structure, which acts as a delta store for small batches of updates. During query processing, probes are performed on both the in-memory delta store and the PMem-resident table to ensure correctness. When system resources permit, the accumulated updates in DRAM are persisted to PMem. This design enables efficient handling of small incremental updates without degrading query performance.

During persistence, the bitmap and the corresponding population count are updated to reflect the newly inserted entries. If a bucket receives additional records, a new array is allocated to accommodate the updated contents. The existing entries are copied to the new array, the new records are inserted, and the bucket pointer is updated to reference the new array. This process preserves the compact layout of the bucket while ensuring consistency and efficient access.

4 SPAR-HT Operational Workflow: Building the Index and Executing Probes

This section outlines the detailed build and probe steps for SPAR-HT. Our design optimizes both stages through structured multi-step processes, unlike the general hash join technique discussed previously. Three essential processes make up the build phase, which effectively builds the internal hash table and guarantees memory and cache efficiency. In a similar vein, the probe phase consists of two separate processes designed to find and confirm matching entries fast and with the least amount of overhead. In order to improve join performance on large-scale datasets, these methods are intended to take use of hardware features and data access patterns. The following SQL query, which involves a join between two tables R and S, is used in all ensuing discussions.

```
SELECT COUNT(r.val) FROM R r, S s WHERE r.key = s.key;
```

4.1 Build Phase

Many hash tables suffer from high resizing costs and over-allocation of memory in order to maintain constant-time lookups. To improve persistent-memory efficiency, structures such as CCEH and Level Hashing address this challenge by trading off memory utilization. In contrast, our approach completely avoids rehashing by scanning the dimension data twice during the build phase. The first pass determines the exact storage requirements, enabling precise memory allocation without overflow and allowing us to use a contiguous array instead of

Algorithm 1: Build the hash table

```
   input  : (K_1, V_1), ... (K_N, V_N)
   output: Hash table
 1 bits[] ← 0;
 2 prefixCount[] ← 0;
 3 atomic_index ← 0;
 4 flush_bytes ← 256;
 5 flush_elems ← flush_bytes/(sizeof(key) + sizeof(value));
   /* Step 1: Calculate the prefix population count                    */
 6 for each key K[i] in thread chunk do
 7 |   bit_pos = hash(K[i]);
 8 |   bucket = bit_pos/64;
 9 |   Automic(updatebits[bucket]);
10 |   ++LocalCount_t[bucket];
11 After all threads finish, merge counts to compute prefixCount[bucket];
   /* Step 2: Precise memory allocation                                */
12 Partition buckets among threads;
13 for each bucket b in thread range do
14 |   Allocate(VariableArray) ← size(prefixCount[b]);
   /* Step 3: Insert the data into SPAR-HT and persist                 */
15 Partition keys among threads;
16 for each key K[i] in thread chunk do
17 |   bucket = hash(K[i])/64;
18 |   pos = atomic_index[bucket].fetch_add(1);
19 |   keyArr[pos] = K[i];
20 |   valArr[pos] = V[i];
21 |   Update validity bits;
22 |   if pos completes a 256B chunk then
23 |   |   pmemobj_persist(keyArr + chunk_start, flush_bytes);
24 After thread finishes, flush any remaining partial chunk;
```

linked structures. The second pass inserts the data and persists it into persistent memory. Accordingly, the build phase of `SPAR-HT` consists of three distinct steps as shown in Figure 2, described below.

Step 1: Bitmap creation and popularity count — To minimize synchronization overhead, the input keys are partitioned across multiple threads. Each thread maintains a thread-local counter array (algorithm 1, lines 6–10), eliminating contention on shared counters during the counting phase. The hash function calculates a bitmap position for each key; the thread updates its local bucket counter while the matching bit is atomically set in the shared bitmap. The final bucket cardinalities (*population_ count* or *prefix_ count*) are obtained by aggregating the local counts once all threads have finished.

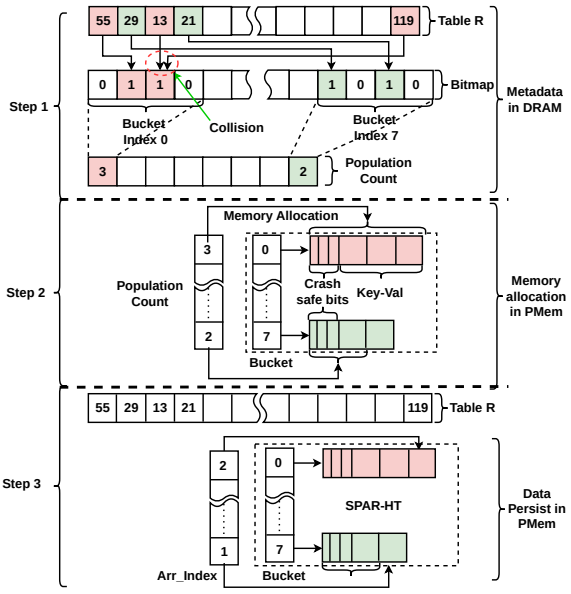


Fig. 2: Build phases of the SPAR-HT Framework

Step 2: Memory allocation — As shown in algorithm 1 (lines 12–14), storage allocation is parallelized by partitioning buckets among threads. Each thread allocates storage for its assigned range independently. The previously calculated prefix_count is used to establish the allocation size for each bucket. The variable-sized bucket arrays are allocated concurrently in this step to eliminate thread contention.

Step 3: Insert & persist — In the second iteration of the dimension table, the input keys are divided among the threads, and each thread simultaneously puts its designated keys into the appropriate buckets. The hash function selects the target bucket for each key, and an atomic counter designates a distinct location in the bucket's array. The corresponding validity bit is set to indicate that the entry is active, and the key–value pair is written to the designated contiguous array, as shown in algorithm 1 (lines 15–21). To optimize persistence and reduce write amplification, the implementation flushes each 256B chunk of a bucket incrementally into persistent memory as it is filled, aligning with the underlying Intel Optane XPLine granularity [16], described in algorithm 1 (lines 22-23). Any remaining partial chunks are flushed at the end, ensuring all inserted data is fully durable without requiring a single large flush after the entire insertion phase.

4.2 Probe phase

The probe phase comprises two steps. First, the in-memory hash table is checked; if a match is found, the search returns true. Otherwise, a bitmap-based prefiltering step is applied to prune probe keys before accessing the PMem hash table. Specifically, if the corresponding bit in the bitmap for a hashed probe key is not set, the key can be immediately discarded without probing the hash table. This early filtering step significantly reduces unnecessary memory accesses.

In the second step, when the bitmap indicates a potential match (i.e., the corresponding bit is set), the algorithm proceeds to search the corresponding bucket

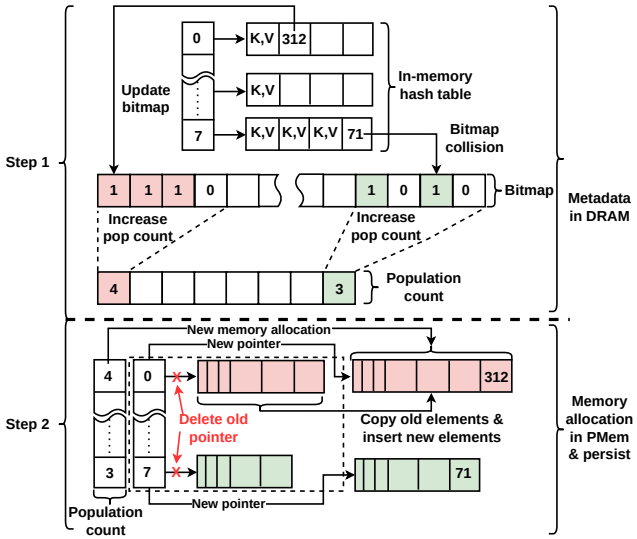


Fig. 3: Second phase of incremental insert of the SPAR-HT

in the hash table. This lookup is accelerated using SIMD-based comparisons, enabling multiple keys within the bucket to be checked in parallel. Furthermore, to reduce memory access latency, software prefetching is employed to load the next set of probe elements while the current SIMD operations execute.

We describe these steps in detail below.

Step 1: Bitmap search. — The approach first searches for the probe key in a small in-memory hash table. If the key is found, the lookup returns immediately. Otherwise, the method computes the hash of the key to identify the correspond-

Algorithm 2: Bitmap filtered lookup operation

input : $Key(k)$
output: True if key exists, otherwise False

```
if find_dramHT(Key) == true then
    return true;
pos ← hash(k);
bucket_index(b) ← pos/64;
bit_position(i) ← pos%64;
if bits[b][i] != 1 then
    return false;
else
    return ScanBucket(b, k);
```

Algorithm 3: Bucket search operation: **ScanBucket(b, k)**

```
   input  : Bucket(b), Key(k)
   output: True if key exists, otherwise False
 1 Let A be the key array of size n stored in bucket b;
 2 i ← 0;
 3 w ← 4;
   // number of keys processed per SIMD vector
   /* Vectorized Scan (SIMD):                                        */
 4 while (i + w) < n do
 5     __builtin_prefetch(&keys[i + w], 0, 3)
 6     Load w keys starting at position i;
 7     Generate a bitmask of candidate matches;
 8     while bitmask! = empty do
 9         Extract the next candidate position p;
10         if validity bit of slot[p] == 1 then
11             return true;
12         Remove p from the mask;
13     i ← i + w;
   /* Scalar Scan (Remaining Entries):                               */
14 while i < n do
15     if validity bit of slot[i] == 1 && A[i] == k then
16         return true;
17     i ← i + 1;
18 return false;
```

ing bucket and bit position in a bitmap, and proceeds with the probe. As outlined in Algorithm 2, the bitmap is examined to determine whether the target bucket may contain the key quickly. If the corresponding bit is not set (Line 6), the probe terminates early. Otherwise, the search continues in persistent memory `SPAR-HT` (line 9).

Step 2: SIMD-accelerated bucket search. — As shown in Algorithm 3, the key array within the variable-length structure of the bucket is examined to determine whether the target key exists. Each entry includes a valid bit that indicates whether a valid key is present in the slot.

The approach leverages SIMD instructions to compare four keys in parallel when AVX2 is available. Specifically, given 64-bit keys and the use of 256-bit SIMD registers (e.g., __m256i), the method performs simultaneous comparisons to improve probe efficiency. The comparison produces a bitmask indicating possible matches as depicted in Algorithm 3 (lines 6-8). Before reporting success, the algorithm checks the validity of the matching bit for each candidate match to ensure the entry is legitimate. Software prefetching is employed to mitigate memory access latency (Line 5), enabling earlier retrieval of target data and

improving overall probe performance. Any remaining entries that the vectorized scan cannot process are handled using a scalar scan (lines 14-17).

If SIMD support is unavailable, the algorithm falls back to a scalar scan that checks each key sequentially while validating the corresponding bit. The function returns true if a valid matching key is found; otherwise, it returns false.

4.3 Incremental insert phase

Incremental insertion consists of two phases as shown in Figure 3. In the first phase, incoming records are inserted into an in-memory hash table, which occupies approximately one quarter of the space of the PMem-resident structure. When a new chunk of data arrives, the records are immediately inserted into this DRAM-resident structure, enabling significantly faster updates than directly modifying PMem. In the second phase, when no queries are running and system resources are available, the system iterates over the in-memory hash table and merges the updates into the PMem-resident structure. The second phase consists of two steps. In the first step, the corresponding bitmap and population count are updated, as outlined in Algorithm 4 (lines 1–6). In the second step, the updated population count is then compared with the current array capacity of the corresponding bucket cell. If the population exceeds the existing capacity, a larger array is allocated, existing entries are copied to the new array, and newly inserted elements are appended as described in lines 7–23.

This approach preserves the conciseness of the PMem hash table while enabling efficient incremental updates and maintaining high query performance.

5 Experiments

This section evaluates the performance of the `SPAR-HT`. We compare `SPAR-HT` to three modern concurrent PMem hash tables, Dash [10], CCEH [11], and Level hashing [19] from the Dash library [4]. The Dash-Extendible Hashing (Dash-EH) variation from the Dash-enabled hash tables was employed in our assessment. When compared to other Dash versions, Dash-EH provides speedier performance. Moreover, we describe the workload, datasets, experimental and hardware settings.

This section's objective is to respond to the following research questions:

1. How does `SPAR-HT`'s performance in join operations compare to that of other hash tables on PMem (section 6)?
2. In comparison to other hash tables, how does `SPAR-HT` scale as the number of threads increases (section 6)?
3. How does `SPAR-HT` compare with existing hash tables in terms of space efficiency (section 6.4)?
4. What is the cache efficiency of `SPAR-HT` in comparison to other hash tables (section 6.5)?

Workload and datasets: Our workload involves an equi-join between two relations, R and S, where R has unique primary keys and S has corresponding

Algorithm 4: Batch Update from DRAM Hash Table

input : $DRAM\_HT(dramHT), Thread\_count(T)$
output: Insert incremental records in the HT

```
/* Phase 1: Count incoming entries                                   */
foreach DRAM bucket d ∈ dramHT do
    foreach key k ∈ d do
        p ← hash(k, bitmap_size) ≫ 6;
        bit ← bit_pos & 63;
        bits[b] ← bits[b] | (1 ≪ bit);
        newCount[p]++;

/* Phase 2.1: Resize PMem buckets                                    */
foreach bucket b where newCount[b] > 0 do
    prefixCount[b] ← prefixCount[b] + newCount[b];
    oldSize ← size(b);
    newSize ← prefixCount[b];
    A ← allocate(newSize);
    copy existing entries of b to A;
    pmemBuckets[b] ← A;
    persist(A);
    atomic_index[b] ← newSize;

/* Phase 2.2: Parallel insertion & persist                           */
foreach DRAM bucket d using T threads do
    foreach (k, v) ∈ d do
        p ← hash(k, bitmap_size) ≫ 6;
        pos ← atomic_index[p].fetch_add(1);
        insert (k, v) into bucket p at pos;
        set validity bit(pos);
        if 256B boundary reached then
            persist inserted chunk;

foreach PMem bucket b do
    persist remaining unflushed entries;
```

foreign keys referencing R. Although our experiments focus on PK-FK joins, SPAR-HT can support M:N joins as well. We run the following query:

```
SELECT COUNT(r.val) FROM R r, S s WHERE r.key = s.key;
```

In our experimental setup, we perform equi-join operations between two relations, denoted as R (build) and S (probe), where each tuple consists of an 8-byte key and an 8-byte value/address of the row. At a scale factor of 10, relation R contains 10 million tuples, while relation S contains 26 million tuples. To examine the impact of larger data volumes on persistent memory behavior, we further increase the scale factor to 50, resulting in 50 million tuples in R and 132 million

tuples in S. This scaling allows us to evaluate the performance of the evaluated join algorithms under higher memory pressure.

Prior experimental studies have largely overlooked the impact of data selectivity in the context of join operations. However, real-world workloads often exhibit varying selectivity levels in the build relation during probing. In this work, we systematically evaluate join performance across two input relations under different selectivity levels, ranging from 20% to 100%. In addition, we consider skewed access patterns by generating probe workloads following a Zipfian distribution. We also account for the effect of key order on join performance by fully shuffling the build and probe data, thereby eliminating any bias arising from presorted or clustered keys.

Experimental setting: The software was compiled using GCC 11.4.0 with the –O3 optimization level enabled. We vary the number of threads from 1 to 16 to evaluate the scalability of PMem-based hash tables. All experiments were conducted on a machine equipped with an Intel Xeon Gold 6248 processor running at 2.50 GHz, featuring 20 physical cores with two hardware threads per core (hyper-threading enabled). The system is configured with 370 GB of DDR4 main memory. Each core has private 32 KB L1 instruction and 32 KB L1 data caches, along with a 1 MB private L2 cache, while all cores share a 55 MB last-level (L3) cache. In addition, the system is equipped with 126 GB of Intel Optane persistent memory. PMem is accessed via fsdax on an ext4 filesystem and configured to use AppDirect mode. In fsdax mode, an Intel Optane namespace allows Linux file systems to bypass the kernel page cache, allowing for direct byte-addressable mapping with *mmap()* [18].

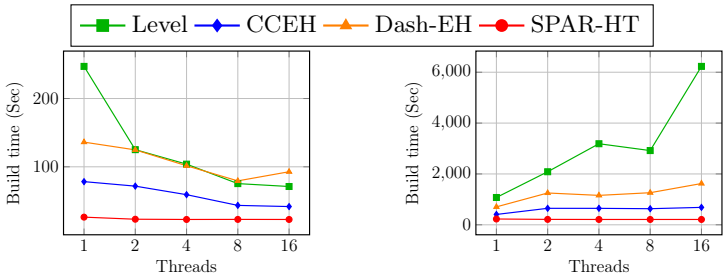
6 PMem benchmarks

In this section, we evaluate the total execution time of the join operation across different hash indexes while varying the number of threads from 1 to 16. Across both data volumes, the build phase dominates the overall execution time. When the build size is 10M, the probe size is 26M, and the selectivity is 20%, the build phase accounts for 95–98% of the total runtime, whereas the probe phase contributes only 2–5%. As selectivity increases to 100%, the probe time increases accordingly. We also observe that a substantial portion of the build time is spent on resizing operations. This trend remains consistent when the data volume is increased by a factor of five.

6.1 Smaller dataset

Build - `SPAR-HT` consistently achieves the lowest cost across all thread counts, as seen in Figure 4(a), suggesting noticeably improved efficiency when compared to the state-of-the-art PMem hash tables. During single-threaded hash table construction, `SPAR-HT` significantly outperforms existing persistent memory indexing structures, achieving a 2.98×, 5.18×, and 9.39× speedup over CCEH, Dash, and Level Hash, respectively. This performance indicates that the underlying architecture is highly optimized for modern hardware constraints, allowing

it to complete the same workload in a fraction of the time required by traditional persistent memory hash tables.



(a) Build time (R-10M) (b) Build time (R-50M)

Fig. 4: Overall build time for 10M and 50M records across varying thread counts

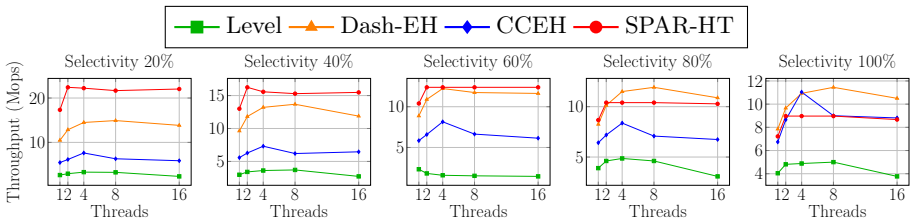


Fig. 5: Probe throughput of Level, CCEH, Dash-EH, and `SPAR-HT` over 26M records under varying thread counts and selectivity levels (higher is better)

Probe - `SPAR-HT` consistently achieves the maximum throughput throughout selectivity settings and thread counts, as illustrated in Figure 5. At 20%–60% selectivity, `SPAR-HT` significantly outperforms the other hash tables, achieving up to 3–4× higher throughput than CCEH and 6–8× higher throughput than Level Hashing as depicted in Table 1a, while maintaining strong scalability with increasing threads. Dash becomes more competitive at high selectivities (80%–100%), where many probes succeed, and can surpass `SPAR-HT` at higher thread counts (4 or more threads). Nonetheless, `SPAR-HT` still maintains superior or comparable throughput in most configurations, consistently demonstrating efficient lookup paths and better utilization of memory and CPU resources. In comparison, CCEH delivers moderate throughput, largely insensitive to selectivity, and Level Hashing experiences declining performance under high concurrency, reinforcing that `SPAR-HT` is the most robust and high-performing hash table overall.

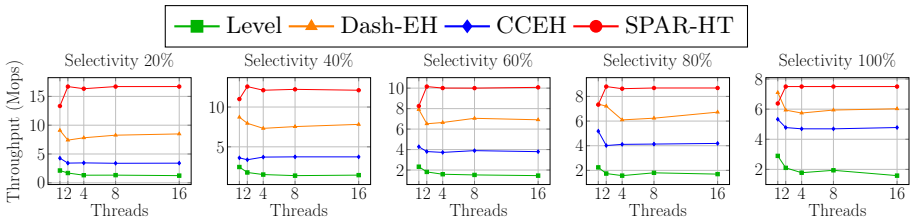


Fig. 6: Probe throughput of Level, CCEH, Dash-EH, and SPAR-HT over 132M records under varying thread counts and selectivity levels (higher is better)

6.2 Larger dataset

Build - The build results reveal (Figure 4(b)) significant differences across the evaluated hash tables as thread count increases. SPAR-HT consistently achieves the best performance, maintaining nearly constant insert time (228–210) across all threads. Compared to SPAR-HT, CCEH is about 2–3× slower, while Dash is roughly 4–8× slower depending on the thread count. Level Hashing performs the worst, becoming up to 30× slower at 16 threads due to severe scalability degradation. This behavior is mainly caused by PMem write-bandwidth saturation and persistence overheads, such as cache-line flushes and memory fences, which increasingly dominate large-scale build workloads for Level and Dash, while SPAR-HT remains largely unaffected due to its PMem efficient design.

Probe - We evaluated the probe throughput of four hash tables, Level, CCEH, Dash, and SPAR-HT, on a 132M-record probe dataset across thread counts (1–16) and selectivities from 20% to 100% as depicted in Figure 6. SPAR-HT consistently achieves the highest throughput, particularly at low and medium selectivities, delivering roughly 2× the throughput of Dash, 5× that of CCEH, and over 13× that of Level at 20% selectivity with 16 threads as shown in Table 1b. Dash provides moderate throughput and approaches SPAR-HT at higher selectivities, while CCEH shows stable but moderate performance, and Level remains the lowest due to cache inefficiencies. Overall, throughput decreases slightly as selectivity increases, and thread scalability is limited across all hash tables.

6.3 Update operation

To evaluate the efficiency of handling dynamic updates, Figure 7 compares the execution latency of SPAR-HT against Level Hashing, CCEH, and Dash for an incremental load of 5 million records (10% of the total dataset). CCEH was the most performant solution, executing the update operation in 43 sec. Level Hashing (66 sec) and Dash (84.42 sec) showed moderate performance loss, with 1.53× and 1.96× slower than the CCEH, respectively. SPAR-HT had the highest latency, requiring 117.48 sec to execute the same volume of records, resulting in a 2.73× increase in execution time versus CCEH. This discrepancy in performance suggests that CCEH gains from more cache-aware design and less element

(a) Probe size 26M

	SPAR-HT vs Level					SPAR-HT vs CCEH					SPAR-HT vs Dash-EH				
Sel.	1	2	4	8	16	1	2	4	8	16	1	2	4	8	16
20%	6.6	7.6	6.7	6.7	9.4	3.2	3.7	2.9	3.4	3.8	1.7	1.7	1.5	1.5	1.6
40%	4.4	4.8	4.3	4.1	5.7	2.3	2.6	2.1	2.5	2.4	1.4	1.4	1.2	1.1	1.3
60%	3.3	3.0	3.2	3.1	4.6	1.8	1.9	1.5	1.9	2.0	1.2	1.1	1.0	1.1	1.1
80%	2.2	2.3	2.1	2.3	3.3	1.4	1.5	1.2	1.5	1.5	1.1	1.0	0.9	0.9	0.9
100%	1.8	1.9	1.8	1.8	2.3	1.1	1.0	0.8	1.0	1.0	0.9	0.9	0.8	0.8	0.8

(b) Probe size 132M

	SPAR-HT vs Level					SPAR-HT vs CCEH					SPAR-HT vs Dash-EH				
Sel.	1	2	4	8	16	1	2	4	8	16	1	2	4	8	16
20%	6.3	9.8	12	13	13.4	3.1	4.9	4.7	4.9	4.9	1.5	2.3	2.1	2.0	2.0
40%	4.5	7.1	8.0	8.9	8.4	3.0	3.7	3.3	3.3	3.2	1.3	1.6	1.7	1.6	1.5
60%	3.5	5.6	6.3	6.6	6.9	1.9	2.7	2.7	2.6	2.7	1.0	1.6	1.5	1.4	1.5
80%	3.3	5.1	5.5	4.8	5.1	1.4	2.2	2.1	2.1	2.1	1.0	1.2	1.4	1.4	1.3
100%	2.2	3.6	4.2	3.9	4.7	1.2	1.6	1.6	1.6	1.6	0.9	1.3	1.3	1.3	1.2

Table 1: Relative speedup of SPAR-HT across varying selectivity and thread counts. Values > 1 indicate SPAR-HT is faster.

shifting. In contrast, in order to maintain conciseness, SPAR-HT must copy the old array to the new array and add the new members; nevertheless, this is an expensive PMem operation. Since the overhead of incremental updates is higher, the target application of SPAR-HT is more suitable for read-mostly workloads, which is typical in analytical applications.

6.4 Memory usage

Memory efficiency varies dramatically among the four models tested, with requirements ranging from 1.18 to 2.12 GB. CCEH exhibits the largest memory footprint (2.12 GB), primarily due to its multi-level directory structure and segment-based allocation. Although this design is cache-conscious, it can incur higher memory overhead under dynamic workloads. Dash and Level, on the other hand, use less data, with 1.24 GB and 1.95 GB, respectively. In particular, SPAR-HT delivers the highest optimized performance, using only 1.18 GB — a 44% reduction over CCEH. This implies that SPAR-HT uses a more compact data structure or has a higher load factor, allowing it to maintain competitive indexing capabilities while staying the most resource-efficient alternative for memory-constrained systems.

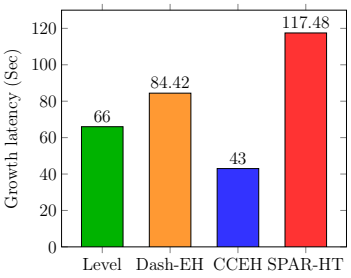


Fig. 7: Execution time for update of 5M records in the existing `SPAR-HT`

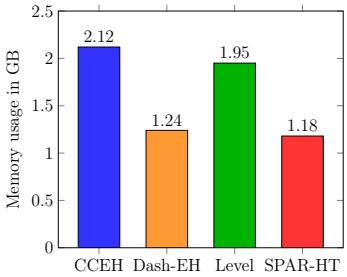


Fig. 8: Memory usage of four hash tables in PMem

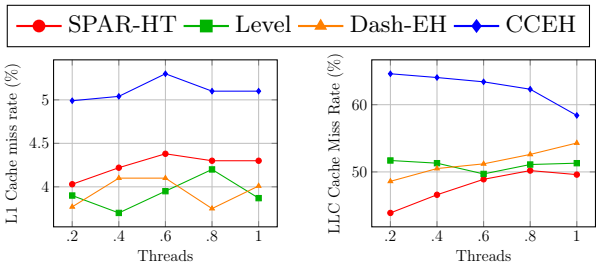


Fig. 9: Cache Performance of Hash Tables During Join Execution with 50M Build and 132M Probe Records (Lower is better)

6.5 Cache performance

Figure 9 reports the L1 and LLC cache miss rates across different workload settings. Overall, `SPAR-HT` consistently achieves the lowest LLC miss rate, ranging between 43.9% and 50.2%, indicating better last-level cache locality compared to the other approaches. While Level Hashing and Dash exhibit slightly lower L1 miss rates (around 3.7–4.1%), their LLC miss rates remain noticeably higher, reaching up to 54.3%. In contrast, CCEH suffers from the poorest cache behavior, with L1 miss rates exceeding 5% and LLC misses as high as 64.6%, suggesting inefficient memory access patterns and reduced spatial locality. Across all workload configurations, `SPAR-HT` maintains stable cache performance, demonstrating improved memory locality and more efficient cache utilization compared to the baselines.

7 Discussion

In our experiments, for the smaller dataset, CCEH, Level Hash, and Dash initially benefit from increased parallelism, with throughput rising as thread count grows, but performance starts to drop at high thread counts due to contention on shared metadata, synchronization overheads, and memory system pressure.

However, SPAR-HT shows linear scaling and does not degrade its performance. For the larger dataset, however, these hash tables fail to scale effectively except SPAR-HT, as the combined impact of random memory accesses, cache-line invalidations, and persistent memory durability enforcement (flushes and fences) overwhelms the benefits of parallelism, leading to limited throughput gains even at moderate concurrency levels. Even though SPAR-HT consistently maintains high throughput for build and probe operations, its performance is worse than others in incremental updates. Since our target workloads are read-mostly analytical workloads, we mainly highlight SPAR–HT's superior build and probe performance, emphasizing its robustness and optimization over existing hash tables. Furthermore, of all the PMem hash tables covered here, SPAR-HT has the most memory-efficient architecture.

8 Conclusion

We have introduced a cache-aware, and lightweight hybrid DRAM-PMem optimized hash table that is specifically designed to be reused while performing hash join operations in database query processing. We compare SPAR-HT against state-of-the-art PMem-aware hash tables, and the experiment results demonstrate that it consistently outperforms the build operation across all load factors and threads. While Dash performs slightly better than SPAR-HT under low data volume and high selectivity, SPAR-HT outperforms it in all other scenarios. Moreover, SPAR-HT consistently outperforms both Level Hashing and CCEH across all workloads and thread counts. Although SPAR-HT incurs higher growth latency, it is amortized in read-mostly analytical workloads with repeated join operations.

References

1. Akel, A., Caulfield, A.M., Mollov, T.I., Gupta, R.K., Swanson, S.: Onyx: A prototype phase change memory storage array. In: 3rd Workshop on Hot Topics in Storage and File Systems (HotStorage 11) (2011)
2. Blanas, S., Li, Y., Patel, J.M.: Design and evaluation of main memory hash join algorithms for multi-core cpus. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. pp. 37–48 (2011)
3. Chatterjee, S., Zhang, X., Ray, S., Finlay, I., Zuzarte, C., Stoodley, M.: Camel hash table: striking a balance between cpu and memory efficiency in main-memory hash join. In: EDBT (2026)
4. Dash Library. https://github.com/baotonglu/dash
5. Fagin, R., Nievergelt, J., Pippenger, N., Strong, H.R.: Extendible hashing — a fast access method for dynamic files. ACM Transactions on Database Systems (TODS) **4**(3), 315–344 (1979)
6. Hennessy, J.L., Patterson, D.A.: Computer architecture: a quantitative approach. Elsevier (2011)
7. Hu, D., Chen, Z., Wu, J., Sun, J., Chen, H.: Persistent memory hash indexes: An experimental evaluation. Proceedings of the VLDB Endowment **14**(5), 785–798 (2021)

8. Izraelevitz, J., Yang, J., Zhang, L., Kim, J., Liu, X., Memaripour, A., Soh, Y.J., Wang, Z., Xu, Y., Dulloor, S.R., et al.: Basic performance measurements of the intel optane dc persistent memory module. arXiv preprint arXiv:1903.05714 (2019)
9. Litwin, W.: Linear hashing: a new tool for file and table addressing. In: VLDB. vol. 80, pp. 1–3 (1980)
10. Lu, B., Hao, X., Wang, T., Lo, E.: Dash: Scalable hashing on persistent memory. arXiv preprint arXiv:2003.07302 (2020)
11. Nam, M., Cha, H., Choi, Y.r., Noh, S.H., Nam, B.: {Write-Optimized} dynamic hashing for persistent memory. In: 17th USENIX Conference on File and Storage Technologies (FAST 19). pp. 31–44 (2019)
12. Pagh, R., Rodler, F.F.: Cuckoo hashing. Journal of Algorithms **51**(2), 122–144 (2004)
13. Shatdal, A., Kant, C., Naughton, J.F.: Cache conscious algorithms for relational query processing. Tech. rep., University of Wisconsin-Madison Department of Computer Sciences (1994)
14. Van Renen, A., Horn, D., Pfeil, P., Vaidya, K.E., Dong, W., Narayanaswamy, M., Liu, Z., Saxena, G., Kipf, A., Kraska, T.: Why tpc is not enough: An analysis of the amazon redshift fleet (2024)
15. Wang, C., Hu, J., Yang, T.Y., Liang, Y., Yang, M.C.: {SEPH}: Scalable, efficient, and predictable hashing on persistent memory. In: 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23). pp. 479–495 (2023)
16. Xiang, L., Zhao, X., Rao, J., Jiang, S., Jiang, H.: Characterizing the performance of intel optane persistent memory: A close look at its on-dimm buffering. In: Proceedings of the Seventeenth European Conference on Computer Systems. pp. 488–505 (2022)
17. Yang, J., Kim, J., Hoseinzadeh, M., Izraelevitz, J., Swanson, S.: An empirical guide to the behavior and use of scalable persistent memory. In: 18th USENIX Conference on File and Storage Technologies (FAST 20). pp. 169–182 (2020)
18. Zhuge, Q., Zhang, H., Sha, E.H.M., Xu, R., Liu, J., Zhang, S.: Exploring efficient architectures on remote in-memory nvm over rdma. ACM Transactions on Embedded Computing Systems (TECS) **20**(5s), 1–20 (2021)
19. Zuo, P., Hua, Y., Wu, J.: {Write-Optimized} and {High-Performance} hashing index scheme for persistent memory. In: 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). pp. 461–476 (2018)

Multi-Modal RAG Search in Vector-Relational Databases: an Experimental Analysis

Suchitra Roy and Suprio Ray

University of New Brunswick, Canada
{sroy10,sray}@unb.ca

Abstract. Efficiently processing the ever-growing volumes of unstructured data is a pressing need. Traditional relational databases are not well-suited for managing and querying unstructured data. Advances in Large Language Model (LLM) have ushered in a new paradigm that enables natural language processing tasks on unstructured text. Since LLMs are susceptible to 'hallucination', Retrieval Augmented Generation (RAG) guided by vector indexes has emerged as a key enabler for LLM oriented data analysis. Consequently, several specialized vector database systems have been proposed. However, maintaining different data systems for different data tasks is fraught with challenges. A unified vector-relational database can offer the best of both worlds and hence may be preferable over custom solutions. Although vector-relational databases, based on vector extensions, have been introduced recently, their functionalities and performance vary widely.
Recent innovations in LLMs technology have led to Vision-Language Models (VLMs) that can incorporate visual inputs. This has opened up new possibilities for analyzing multimedia data, besides text, through multi-modal retrieval. In this paper, we present a multi-faceted and systematic experimental evaluation of VLM supported multi-modal RAG search with two open-source vector-relational databases. For our experimental analysis, we utilize an AI-assisted personalized nutrition application that we have developed, as well as a synthetic dataset with different data volumes to evaluate the scalability. Our study reveals several interesting findings and a few issues with existing systems, and it points to some future research directions.

Keywords: Vector-relational database · vision-language model (VLM) · Retrieval Augmented Generation (RAG) · multi-modal search.

1 Introduction

With the rapidly rising volumes of unstructured data, including text, image, audio and video, efficient processing of these data has become imperative. In recent years, pre-trained Large Language Model (LLM) [42] emerged as a transformative technology that has enabled the analysis of unstructured data and perform many tasks, such as, clustering, classification and information retrieval. LLMs

like ChatGPT [3] and Llama [11] have become quite popular within a short period of time. LLMs are typically Deep Neural Network (DNN) models that are pre-trained on vast bodies of natural language text corpus. A key idea behind LLMs is to represent data as vectors using neural embedding techniques.

Although LLMs have demonstrated remarkable feats with Natural Language Processing (NLP) applications, they suffer from 'hallucination', a term used to refer to the fact that LLMs may often provide responses that are fabricated and factually incorrect. This is caused by LLMs' inability to include domain-specific knowledge that is not part of the pre-training corpus. To address this issue, Retrieval-Augmented Generation (RAG) was introduced [30]. It can incorporate domain knowledge by integrating a pretrained retriever model with an external knowledge base, which is typically indexed by techniques, such as vector indexing [10, 31].

Recently, the advent of Vision-Language Models (VLMs) has made multi-modal analysis of text and multimedia data practical. There are several VLM training paradigms [26]. Contrastive training is a common approach that uses pairs of positive and negative examples, whereas the masking strategy involves reconstructing the missing patches of an image given an unmasked text caption. VLMs can leverage open-source LLMs, like Llama, as pretrained backbones. Generative VLMs are trained to generate complete images or long texts. By connecting visual inputs with language inputs, VLMs can enable applications, such as AI-assisted Personalized Nutrition (AIPN). Several projects [32,38,40,43] leveraged LLMs for tasks such as ingredient detection and nutrition assessment. In this paper, we explore how to extend the language model based approaches for ingredient recognition and nutrition assessment by utilizing multi-modal RAG search on vector-relational databases.

While traditional relational database systems are designed to manage well-structured data, they are not as suitable for querying unstructured data. Recently, a few vector databases have been introduced, such as Pinecone [17] and Milvus [13]. They are custom engines designed to efficiently process similarity searches on high-dimensional vectors. Previously, a few papers [24, 36] focused on evaluating Approximate Nearest Neighbor (ANN) queries. Recently, Kang et al. [28] developed a benchmark to evaluate the performance of three vector databases, Milvus, Weaviate [22] and Qdrant [18]. In contrast, a vector-relational database extends a traditional relational database to support both SQL queries and vector search. PostgreSQL with pgvector [16] extension, is an example of such a system. A vector-relational database is attractive because it enables users to leverage existing transactional and analytical query processing capabilities of a relational database, along with semantic search capabilities within one database system. Even a few vector databases use a relational database as the backend. For instance, ChromaDB [4] uses SQLite as the underlying relational database. However, to our knowledge no previous studies evaluated vector-relational databases in the context of VLM-based multi-modal RAG search.

To address the aforementioned research gap, we evaluate multi-modal RAG search with two open-source vector-relational databases: PostgreSQL (with pgvec-

tor extension) and SQLite (with vector extension) [20]. To that end, we develop a full-fledged real-world application. Specifically, we implement a prototype AI-assisted personalized nutrition (AIPN) application, demonstrating multi-modal RAG search empowered by a VLM and guided by a vector-relational database. We also evaluate the scalability of vector relational databases with a synthetic dataset consisting of vector data tables of varying sizes. We conduct a systematic experimental evaluation of different aspects of VLMs and vector-relational databases, as mentioned next.

1. Aspect 1: Vision Language Models (VLM) with multi-modal embedding and prompt response
2. Aspect 2: Multi-modal vector search strategies
3. Aspect 3: Vector-relational databases
4. Aspect 4: Vector data indexing strategies

Our evaluation reveals several interesting findings, which are summarized in the Conclusion section. These findings can provide some guidelines to VLM+RAG application developers, as well as researchers in the community.

The remainder of this paper is organized as follows. Section 2 outlines a motivating use-case and describes an application that we have developed based on this. In Section 3 we discuss our methodology, and details of the different aspects that we have analyzed. Then we present the experimental settings, datasets and evaluation results in Section 4. Finally, we conclude the paper in Section 5.

2 Motivating use-case

In this section, we describe a motivating use-case and prototype system that we have implemented, along with its workflows.

2.1 AI-assisted personalized nutrition (AIPN)

Food is a fundamental human need. Human body requires a number of basic nutritional elements to sustain life. Dietary practices and food choices depend on many factors, including, lifestyle, cultural background, socio-economic standing and location. Balanced nutritional intake is crucial for health and longevity. Many chronic diseases, such as type 2 diabetes, obesity, hypertension, stroke, and neuro-degenerative diseases, are related to dietary habits [37]. According to a report from the World Health Organization (WHO) [2], unhealthy diet along with sedentary lifestyle was the second-most leading contributing factor behind the 17 million deaths from cardiovascular and brain diseases. To reduce the risk of mortality from these chronic conditions, a healthy diet with adequate nutritional content is essential. An unhealthy diet is often characterized by foods with added sugars, and saturated fats, high sodium and low dietary fiber content. However, due to our busy urban life-style it is not easy to do advance meal planning and preparation. Moreover, the assessment of the nutritional quality of a meal that is

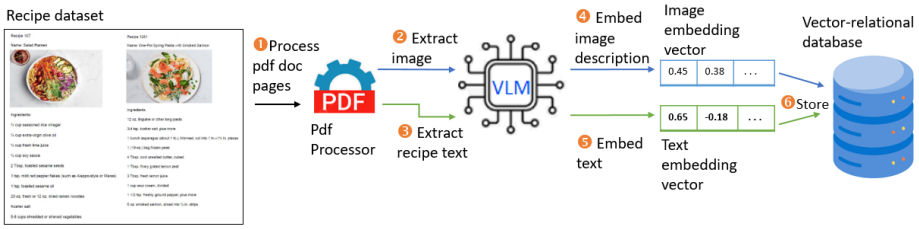


Fig. 1: Data ingestion workflow

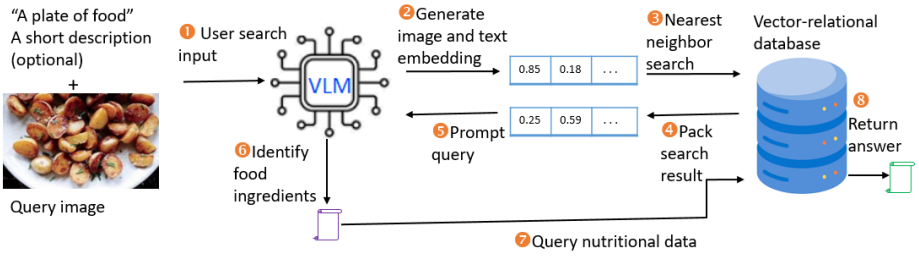


Fig. 2: User search workflow

appropriate for the health of a person requires domain expertise. In this context, AI-assisted personalized nutrition can play an important role to promote healthy eating habits.

With the growth of available food data and the advent of LLMs, several recent studies explored the application of LLMs for tasks such as ingredient detection and nutrition assessment. They include FoodLMM [40], FoodSky [43], CalorieLLaVA [38], and LLaVA-Chef [32]. However, these approaches mainly focused on developing domain specific customized LLM based solutions.

2.2 AIPN prototype implementation and workflows

We have developed a prototype application for AI-assisted personalized nutrition support. The primary use-case involves a user taking an image of her meal and posting this to our application, which generates a list of ingredients detected in the food and a list of nutritional components. This application can help a user determine whether her meal is healthy or not, based on the nutritional content.

There are two main workflows within our system that enable the prototype application. The first workflow involving data ingestion is shown in Figure 1. Step 1 entails processing each page of a pdf document, a Recipe Book, consisting of many food recipes, each with a picture and a text description. The details of the recipe dataset are discussed in Section 4.2. From each page of the pdf document, the image of the dish (if any) and a description of the ingredients in the recipe are extracted, which are then sent to the VLM as a prompt query (steps 2 and 3). The VLM generates a text description of the image, from which a

Table 1: VLM embedding capability

Comparison criteria	LayoutLMv3 [27]	Moondream [14]	CLIP-ViT-B-32 [5]	LLaVA-Llama3 [12]	Granite Vision3.2 [39]
Number of parameters	100M	1.4B	86M	8.0B	2.5B
Context length	512	2048	77	8192	16384
Embedding dimension (length)	768	2048	512	4096	2048

vector embedding is created (step 4). Similarly, the description of the recipe ingredients is also used to create another vector embedding (step 5). Then both embeddings are stored in a table (or tables) within the vector-relational database (step 6).

The second workflow involving user search is depicted in Figure 2. In step 1, the user submits a query image, such as a plate of food, a small text description (optional) and a short search query. The text description and query image are used to generate corresponding text and image embedding vectors using a text embedding model and a VLM based image embedding model respectively (step 2). These embedding vectors may be combined depending on vector search strategy (described in Section 3.2) and then submitted as part of nearest neighbor query/queries to the vector-relational database (step 3). All the matching query results are packed in step 4 and then sent as a prompt query back to the VLM (step 5). In step 6, the VLM generates a list of food ingredients for the user's query image. With this list of ingredients a query is issued to the vector-relational database (step 7) to determine the nutritional components. Based on this query, the result is returned back to the query user (step 8).

3 Methodology

In this section we describe our methodology and discuss the different aspects that we have considered in our analysis.

3.1 Aspect 1: Vision Language Models (VLM) with multi-modal embedding and prompt response

Large Language Models (LLMs) have revolutionized natural language processing tasks, including translation, text summarization, and question answering [35]. Vision-Language Models (VLMs) are the next step in this progression. VLMs are capable of integrating both text and visual inputs, such as image and videos and perform tasks involving those inputs. Several VLMs have already been developed. Based on the two workflows described in Section 2.2, two different capabilities are considered: i) generation of vector embeddings from (texts and) images, and

Table 2: VLM prompt (chat) capability

Comparison criteria	**Moondream** [14]	**GraniteVision3.2** [39]	**LLaVA-Llama3** [12]	**Qwen2.5-VL** [19]
Number of parameters	1.4B	2.5B	8.0B	8.3B
Context length	2048	16384	8192	128000
Embedding dimension (length)	2048	2048	4096	3584
Prompt response quality	○	◐	◐	●

●: excellent; ◐: good; ○: fair.

ii) generation of prompt (chat) responses based on user inputs and provided contexts.

To determine a suitable model for the first task of embedding generation from images, we considered several VLM embedding models. They are LayoutLMv3 [27], Moondream [14], CLIP-ViT-B-32 [5], LLaVA-Llama3 [12] and GraniteVision3.2 [39]. Their key features are summarized in Table 1, including, the number of model parameters, context length and embedding dimension (length). The *number of parameters* refers to the adjustable weights and biases within a neural network. The larger the number of parameters of a model, the more complex it is. As can be seen in the table, this value ranges from 86M with CLIP-ViT-B-32 to 8.0B with LLaVA-Llama3. The *context length* represents the maximum number of tokens a model can process at once i.e. in a single prompt [23]. Among these models, CLIP-ViT-B-32 has the smalled context length of 77, while GraniteVision3.2 has the longest context length of 16384. *Embedding dimension (length)* identifies the number of floating-point numerals used to represent a piece of data as a vector of numbers. This is the length of the embedding vector generated by a particular embedding model. It can be expected that a more complex model will usually take longer to generate embedding than a relatively simpler model.

For the second task of prompt (chat) response generation, several VLM models are considered. They are (as shown in Table 2): Moondream [14], GraniteVision3.2 [39], LLaVA-Llama3 [12] and Qwen2.5-VL [19]. Note that, not all VLM models support embedding creation, whereas not all embedding models support prompt response generation. Therefore, not all the entries in Table 1 and Table 2 are the same. Among the prompt generation capable VLMs, Qwen2.5-VL has the highest number of models parameters (8.3B) and the longest context length of 128000. In contrast, Moondream has the fewest model parameters and the shortest context length. VLM models GraniteVision3.2 and LLaVA-Llama3 sit in the middle of the pack. Hence, Qwen2.5-VL is expected to be the most complex of these models. Finally, the criteria *Prompt response quality* refers to the degree to which a VLM model's prompt response matches the ground truth or the expected response. We discuss about this further in Section 4.6.

3.2 Aspect 2: Multi-modal vector search strategies

Multi-modal data can take different forms, such as text, image, audio and video. Due to their unstructured nature, these data formats cannot be directly queried. Instead, the data must fist be converted into a multi-dimensional feature vector representation by using different embedding techniques, and stored in the database storage. During query time, the feature vectors are retrieved and queries are evaluated. Depending on how vector embedding is created for each data modality and utilized during query processing, there can be different search strategies. We classify them into three categories.

1. Separate embedding based search and re-ranking (SEBSR). In this approach, for an object a separate embedding vector is created for each modality. For instance, a text embedding vector can be created with nomic-embed-text [33] and an image embedding vector is created with LayoutLMv3. The embedding vectors for each modality corresponding to all objects are search independently. These search results are combined and re-ranked to produce the final result-set.

2. Unified-domain embedding based search and ranking (UEBSR). If different modalities can be mapped into a shared unified high dimensional vector space, then a single embedding vector is created per object. These vectors can then be directly compared to evaluate their similarity. For instance, CLIP (Contrastive Language-Image Pretraining) [1] embeddings provide high-dimensional unified representations of text and images. An example CLIP model is CLIP-ViT-B-32.

3. Ensemble embedding based search and ranking (EEBSR). As with the SEBSR approach, in this case separate embedding vectors are generated for every object, with one vector for each modality. Then the different embedding vectors for a particular object are fused to create a single ensemble embedding vector per object. There are different techniques to perform this embedding fusion. In our study, we utilize three different techniques: ConcatMLP [6], MLB [29] and MFH [41].

3.3 Aspect 3: Vector-relational databases

As previously mentioned, we consider two open-source vector-relational databases, namely, PostgreSQL (with pgvector extension) and SQLite (with vector extension). The primary datatype supported by PostgreSQL is `vector` that can be used to represent the data type of the vector embeddings generated by a language model. It also supports data type `halfvec` to store half-precision vectors. SQLite does not have a vector data type, but it supports vector data through the BLOB data type. Further details regarding the vector-relational features of PostgreSQL and SQLite are provided in Section 4.3.

3.4 Aspect 4: Vector data indexing strategies

Vector data indexing is integral to efficiently supporting vector similarity queries in a vector-relational database. For instance, PostgreSQL (with pgvector) sup-

Recipe 107

Name: Salad Ramen

Ingredients:

½ cup seasoned rice vinegar

¼ cup extra-virgin olive oil

¼ cup fresh lime juice

¼ cup soy sauce

2 Tbsp. toasted sesame seeds

1 tsp. mild red pepper flakes (such as Aleppo-style or Maras)

1 tsp. toasted sesame oil

20 oz. fresh or 12 oz. dried ramen noodles

Kosher salt

5-6 cups shredded or shaved vegetables

Recipe 1261

Name: One-Pot Spring Pasta with Smoked Salmon

Ingredients:

12 oz. linguine or other long pasta

3/4 tsp. kosher salt, plus more

1 bunch asparagus (about 1 lb.), trimmed, cut into 1 in.–1½ in. pieces

1 (10-oz.) bag frozen peas

4 Tbsp. cold unsalted butter, cubed

1 Tbsp. finely grated lemon zest

3 Tbsp. fresh lemon juice

1 cup sour cream, divided

1 1/2 tsp. freshly ground pepper, plus more

6 oz. smoked salmon, sliced into ½ in. strips

Fig. 3: A few example (recipe) entries from the dataset

ports two different index types: Inverted File with Flat Compression (IVF-Flat) [10] and Hierarchical Navigable Small World (HNSW) [31]. An IVFFlat index operates by partitioning data vectors into a number of lists of vectors, which is a configurable parameter, during index construction time. Then while executing a vector search, it improves query efficiency over an exhaustive search approach by needing to search only a subset of those lists that are closest to the query vector. An HNSW index constructs a multilayer graph during index building time. In an HNSW index, proximity graphs are constructed with input vector data points. In a proximity graph nodes represent vector embeddings, whereas edges are the connections between nodes. The length or weight of each edge is determined by the distance between the corresponding nodes, calculated using a distance function, such as L2 (Euclidean) or cosine. Multiple layers of the proximity graph are organized such that the top layers contain graphs with fewer nodes for long-range navigation, and lower layers have graphs with more nodes for detailed search.

4 Experimental evaluation and analysis

In this section, we discuss our experimental evaluation of multi-modal RAG search, with an emphasis on vector-relational database operations. In Section 4.1 we describe the experimental settings, and in Section 4.2 we describe the datasets. The vector-relational databases that we used and workload queries are discussed in Section 4.3, along with some details of the vector indexing techniques in Section 4.4. Experimental evaluation of the embedding capability and prompt (chat) capability of VLMs are presented in Sections 4.5 and 4.6 respectively. Evaluation of data ingestion and multi-modal search with our AIPN application are presented in Sections 4.7 and 4.8 respectively. Then we evaluate the scalability of vector-relational databases, with the vector index build time evaluated in Section 4.9 and query execution time in Section 4.10.

4.1 Experimental settings

We used a machine, with an Intel Xeon w3-2423 processor having 6 cores and 2 threads per core, NVIDIA RTX A1000 GPU, and 32 GB of main memory. The machine runs Ubuntu 24.04 LTS. The code was implemented in Python and Ollama [15] was used as the framework to run VLMs locally on our machine.

Table 3: Details of the datasets

Dataset	**Type**	**Cardinality**	**Vector dimension (embedding length)**
Real-world dataset			
recipe	Real-world	13,582	Varies, depends on embedding model (see Table 1)
Synthetic datasets with different scale factors (sizes)			
items10K	Synthetic	10,000	512
items100K	Synthetic	100,000	512
items1M	Synthetic	1,000,000	512
items10M	Synthetic	10,000,000	512

4.2 Datasets

We used two different datasets: a real-world dataset and a synthetic dataset. The real-world dataset consists of a `recipe` table and three tables related to food nutrition. The `recipe` data table, as shown in Table 3, is based on the Food Ingredients and Recipes Dataset [8] created from Epicurious [7]. It consists of 13,582 images of different food dishes and a csv file containing text description of the images. For this experimental study, a Recipe Book was created as a pdf file, containing 13,582 recipes, with an image along with the associated

text description for each recipe. Figure 3 shows a few example recipes from the Recipe Book. The food nutrition data tables are based on The Food Database (FOODB) [21]. It is considered as the world's most comprehensive resource on food nutritional content and their biochemistry information. There are three tables to capture and maintain information based on the csv files obtained from FOODB. These tables are: `food`, `content` and `compound`. They are not shown in Table 3, as they do not involve any vector embedding or vector-relational operation and we do not evaluate any query execution on these three tables.

The synthetic dataset were generated to evaluate vector-relational database scalability. It consists of four tables, each with a column *embedding*. These tables are `items10K`, `items100K`, `items1M` and `items10M`, with each table containing 10× more records than the previous table respectively. The smallest table `items10K` contains 10 thousand records, whereas the largest table `items10M` consists of 10 million records.

4.3 Vector-relational database and queries

To evaluate vector-relational database features, we have selected two open-source relational databases with vector extensions. We have chosen PostgreSQL 18.1 with pgvector 0.8.1 extension and SQLite with vector extension as the vector-relational databases.

PostgreSQL. PostgreSQL with pgvector extension supports a vector data type, which can be declared with the keyword `vector`. It supports nearest neighbor queries based vector distance metrics, such as L2 distance (<->) and cosine distance (<=>). We illustrate the creation of a table with a vector column in PostgreSQL with the `items10K` table from the synthetic dataset.

Create table (PostgreSQL)

```
-- Create a table with a column embedding of type vector.
CREATE TABLE items10K (
    id serial PRIMARY KEY,
    embedding vector(512)
);
```

A cosine distance based nearest neighbor query on `items10K` is shown next. Note that the *`$qry_vector$`* symbolizes the query vector, which is used to compute cosine similarity against the `embedding` column entries of all rows in the `items10K` table.

Query (PostgreSQL)

```
-- Nearest neighbor query based on cosine distance.
SELECT id,  embedding <=> $qry_vector$::vector AS score
FROM items10k
ORDER BY  score limit 5;
```

SQLite. SQLite with vector extension does not support a dedicated vector data type, but rather it stores vector data in a BLOB field. This is shown in the illustration below, with `items10K` table. To initialize the BLOB field as a vector column, a User-Defined Function (UDF) `vector_init` is used. Then, the UDF `vector_quantize` is invoked to quantize the initialized vector column entries.

Create table (SQLite)

```
-- Create a table with a column embedding of type BLOB.
CREATE TABLE items10K (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    embedding BLOB
);

-- Initialize the vector. By default, the distance
-- function is L2. Specify a cosine distance metric.
SELECT vector_init('items10K', 'embedding',
'type=FLOAT32,dimension=512,distance=COSINE');

-- Quantize vector.
SELECT vector_quantize('items10K', 'embedding');
```

SQLite supports cosine distance based nearest neighbor query, an example of which is shown below. SQLite utilizes the UDF `vector_quantize_scan` to compute cosine distance of the query vector against the vectors in a table, where *`$qry_vector$`* is the query vector.

Query (SQLite)

```
-- Nearest neighbor query based on cosine distance
-- on the quantized version.
SELECT d.id,  v.*
FROM items10K AS d
JOIN vector_quantize_scan('items10K', 'embedding',
vector_as_f32($qry_vector$), 5) AS v
ON d.id = v.rowid;
```

4.4 Vector indexing

Vector indexing is not supported by SQLite (vector extension), whereas PostgreSQL (pgvector) support two techniques: IVFFlat and HNSW. These techniques can be used to construct indexes on the embedding vectors stored in a table with a `vector` column.

Note that, as indicated in Table 1, the dimensions (lengths) of the embeddings generated by different VLMs range from 512 to 4096. For instance, Moondream generates embeddings of dimension (length) 2048. When we attempted to create a vector index (IVFFlat or HNSW) on embedding vectors generated by Moondream, we received the following error message:

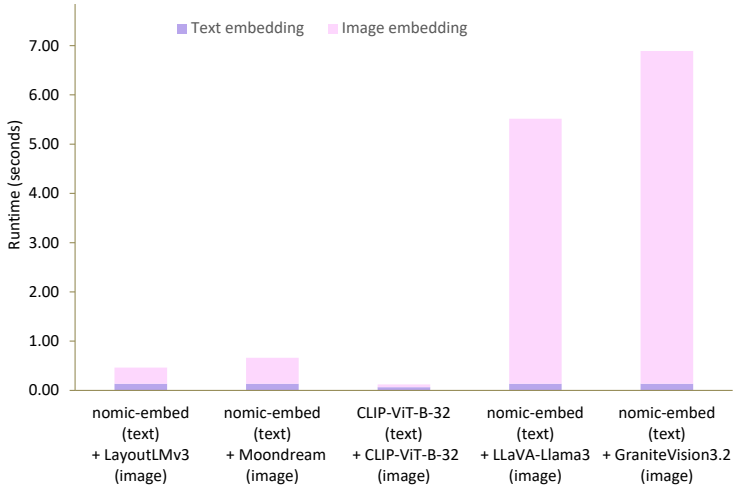


Fig. 4: Total embedding generation (text and image) time

ERROR: column cannot have more than 2000 dimensions for ivfflat index

It suggests that the maximum length of a vector-indexed column with `vector` data type is 2000 in PostgreSQL. It appears that [9] this is due to the default size of the PostgreSQL page, which is 8KB and is not adjustable. This restricts the number of 4-byte floats that can be stored on a page in PostgreSQL. Based on this observation and VLM evaluation results from the next section, for our subsequent experimental evaluation in sections from Section 4.7 to 4.10, we utilize two VLMs, LayoutLMv3 and CLIP-ViT-B-32, which generate embeddings of dimension less than 2000.

4.5 Evaluation: VLM/LLM embedding capability

During a multi-modal RAG-guided search, the user specifies a text and an image as inputs. These inputs need to be converted into embedding vectors, so that a vector-relational database can be queried with them to find the top-K matches using nearest neighbor queries.

In this section, we evaluate the capability of embedding models, particularly that of VLM models to generate embeddings from images. We evaluate 5 different VLM models: LayoutLMv3, Moondream, CLIP-ViT-B-32, LLaVA-Llama3 and GraniteVision3.2. For text embedding, nomic-embed-text is used along with all VLM models, except for CLIP-ViT-B-32. Since, CLIP-ViT-B-32 utilizes a unified embedding domain, we use it for both text and image embedding. The performance of these approaches are shown in Figure 4, and as can be seen CLIP-ViT-B-32 performs the best in terms of the lowest image embedding time. LLaVA-Llama3 and GraniteVision3.2 take the longest times, with

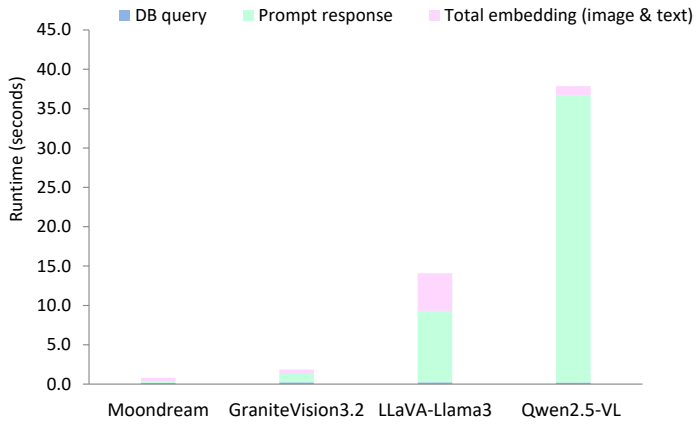
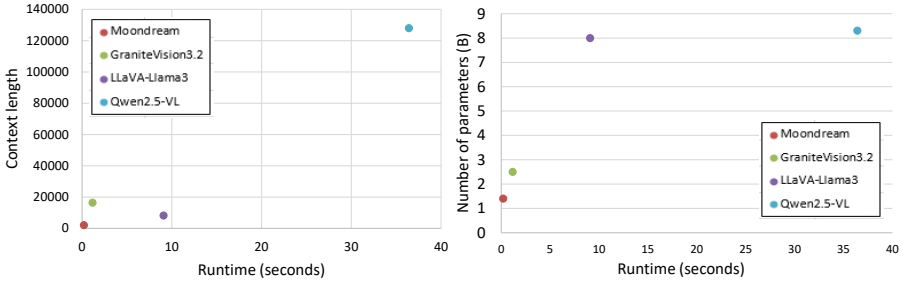


Fig. 5: Overall response time



(a) Context length vs. prompt response time

(b) Num. params (B) vs. prompt response time

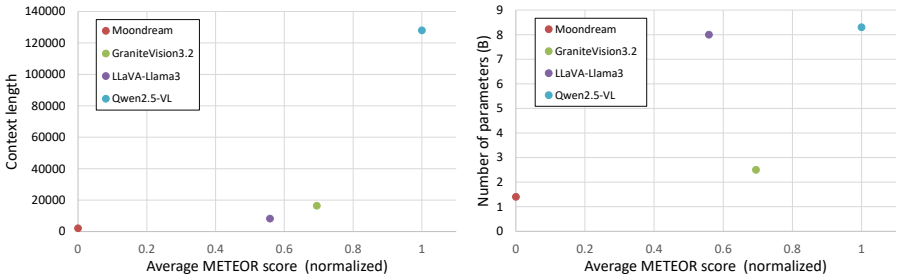
Fig. 6: Prompt response time: context length vs. number of parameters (Billion)

GraniteVision3.2 performing the worst. To explain these results, we observe the features of these VLM models in Table 1. GraniteVision3.2 has the longest context length of 6384, whereas CLIP-ViT-B-32 has the shortest context length of 77. Also, CLIP-ViT-B-32 has the fewest number of model parameters of 86M.

Based on these findings, LayoutLMv3 is utilized as the default image embedding model and nomic-embed-text as the default text embedding model in the subsequent experiments, except for cases where CLIP-ViT-B-32 is applicable.

4.6 Evaluation: VLM prompt (chat) capability

After retrieving the top-k matching nearest neighbors based on the text and image embedding, a context is created using the best matches. Then a prompt query along with the context is issued to a VLM chat model. To provide an ap-



(a) Context length vs. prompt quality (b) Num. params (B) vs. prompt quality

Fig. 7: Prompt quality: context length vs. number of parameters (Billion)

pripriate response to the user's prompt query, the VLM model uses the provided context.

In this section, we evaluate the performance of the prompt capability of several VLM models. These models are: Moondream, GraniteVision3.2, LLaVA-Llama3 and Qwen2.5-VL. In terms of response generation latency, as shown in Figure 5, Moondream has the lowest response latency, whereas Qwen2.5-VL has the longest latency. To further analyze this, we plot the prompt response time and the corresponding context length of the VLM models in Figure 6a. We observe that they are not always correlated, for instance, LLaVA-Llama3 has a smaller context length than GraniteVision3.2 but higher runtime than that of GraniteVision3.2. Figure 6b shows the number of model parameters (Billion) vs. the runtime of the VLM models. The figure indicates that a higher number of model parameters generally imply a longer runtime. Qwen2.5-VL has the most number of model parameters (8.3B, see Table 2) and the highest runtime of prompt response generation among the VLM models.

A lower runtime does not imply a better performance for a particular VLM model with regards to prompt response, since the quality of the response matters significantly. To evaluate the prompt response quality, we compute the METEOR score [25]. We plot of the average METEOR score of the VLM models normalized over the best observed score against (a) the context length in Figure 7a, and (b) the number of model parameters (Billion) in Figure 7b. The figures suggest that a longer context length typically implies a better prompt response quality. In contrast, this is not necessarily the case with the number of model parameters. Overall, the prompt response quality of Moondream was the worst, whereas the response quality of Qwen2.5-VL was the best. These observations are noted with (partially-)colored circles in Table 2. In future, we intend to conduct a more thorough evaluation of prompt response quality.

4.7 Evaluation: data ingestion

In this section, we evaluate the performance of the data ingestion pipeline of our AIPN application, which involves generating embedding vectors from the

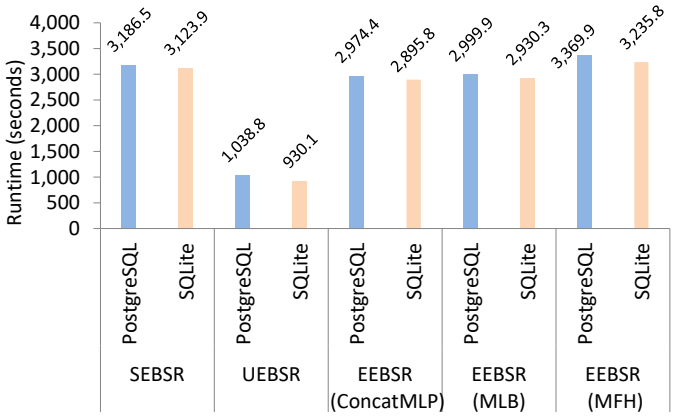


Fig. 8: Data ingestion performance (PostgreSQL vs. SQLite) with different multi-modal vector search strategies

input datasets, and then storing them in the `recipe` table in a vector-relational database. We compare the performance of two open-source vector-relational databases: PostgreSQL (with pgvector) extension and SQLite (with vector extension). Note that, in these experiments no vector index is created for PostgreSQL and the dataset used is `recipe` (see Table 3). We vary the vector search strategies, which determine how the text embedding vector and image embedding vector are combined. These strategies are: SEBSR, UEBSR, EEBSR (ConcatMLP), EEBSR (MLB) and EEBSR (MFH). The results are shown in Figure 8. As can be seen, UEBSR approach performs the best, and with both databases. The reason is that UEBSR utilizes CLIP-ViT-B-32 for both text and image embedding generation. With all other vector search strategies, nomic-embed-text is used as the text embedding model, whereas LayoutLMv3 is used as the image embedding model. Between the two databases, SQLite performs slightly better than PostgreSQL.

4.8 Evaluation: Search performance with different multi-modal vector search strategies

We evaluate the overall multi-modal RAG search performance in our AIPN application. In this search process, the user specifies a query image, a small text description of the image and a brief search text. From these inputs corresponding text and image embeddings are generated, and they are combined using one of the vector search strategies: SEBSR, UEBSR, EEBSR (ConcatMLP), EEBSR (MLB) and EEBSR (MFH). The combined vector is searched in the vector-relational database table `recipe` to find the nearest neighbor matches on cosine distance and then to create a context using the query results. This context and

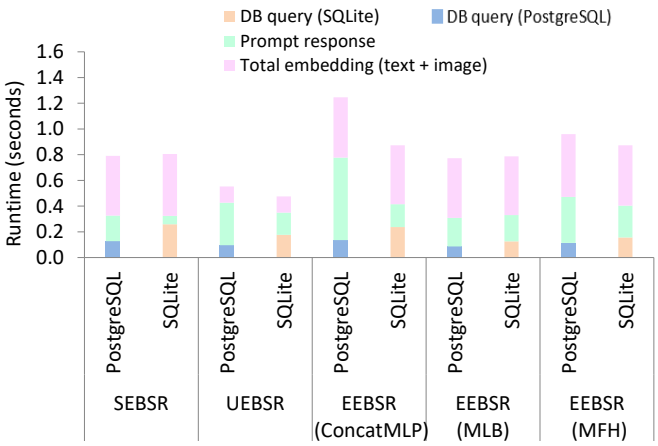


Fig. 9: User search performance (PostgreSQL vs. SQLite) with different multimodal vector search strategies

the user's search text is then sent to the VLM prompt model. We use Moondream as the VLM for prompt response and the dataset used is the `recipe` dataset. Figure 9 shows the overall user search performance, with a breakdown of times spent in each step. As can be seen, the prompt response time constitutes a significant portion of each bar in the stacked bar chart. The total embedding generation time (text and image) varies significantly from approach to approach, taking the least amount of time for UEBSR and significantly longer times for EEBSR approaches. Database query execution times remain steady across different scenarios for both the databases. As for the overall time, USBSR shows the best performance with the lowest overall runtime, as it requires the lowest prompt response time owing to CLIP-ViT-B-32.

4.9 Evaluation: vector-relational database scalability - vector index build time (PostgreSQL)

In this section and the next section, we focus on the scalability of vector-relational databases. For this purpose, we use the synthetic datasets, which differ in data volumes (see Table 3).

In this section, we evaluate the different vector indexing techniques. Specifically, we evaluate the construction time of two vector indexes supported by PostgreSQL (with pgvector extension), namely, IVFFlat and HNSW. SQLite (with vector extension) does not support any vector index. Note that, we deployed the databases as "out-of-the-box" and did not change any configuration parameters to tune them.

Figure 10 shows the time to build index on the tables in the synthetic datasets: items10K, items100K, items1M and items10M. For all the four ta-

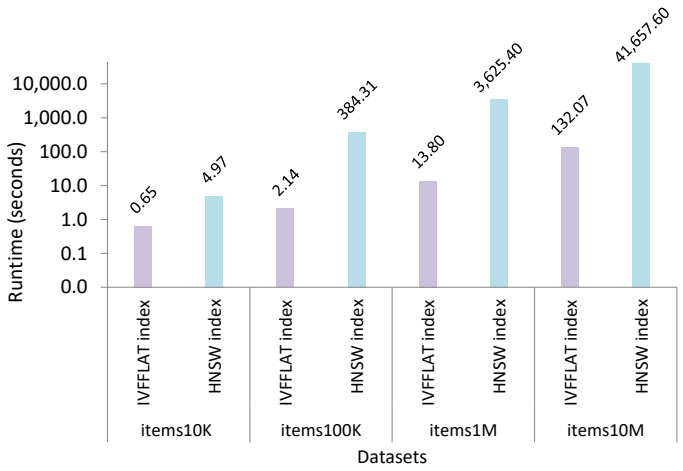


Fig. 10: PostgreSQL vector index construction time

bles, IVFFlat was faster than HNSW in terms of index building time. This is expected, as HNSW is a more complex index structure than IVFFlat, and HNSW construction involves building multilayer proximity graphs based on the embedding vectors. However, what is somewhat surprising is that as the table size increases, the performance gap (index building time) between the two indexes grow. For instance, with items10K table the index building time of IVFFlat is 7.6× faster than that of HNSW, while with items10M table, IVFFlat is 315.4× faster. Although index construction is usually done once and *a priori*, these results may provide some guidance for vector index selection, particularly if the *time-to-query* is an important consideration.

4.10 Evaluation: vector-relational database scalability with query execution time (PostgreSQL vs. SQLite)

Next we present the results based on the query execution time, by executing the vector search query directly on the corresponding database tables in the synthetic dataset. To evaluate the impact of data volume and indexing on the query performance, our evaluation involves three different versions of PostgreSQL setup: PostgreSQL without any index, and PostgreSQL with the two vector indexes, IVFFlat and HNSW. Note, we only evaluate SQLite without any index due to a lack of support for a vector index. The performance results are plotted in Figure 11. As can be observed, PostgreSQL generally performs better than SQLite. Regarding the impact of using vector indexes on query performance, PostgreSQL with either of the indexes, IVFFlat or HNSW, performs better than PostgreSQL without any index. Between, the two indexes, there is no appreciable difference in performance at lower scale factors. However, at the highest scale factor, i.e.

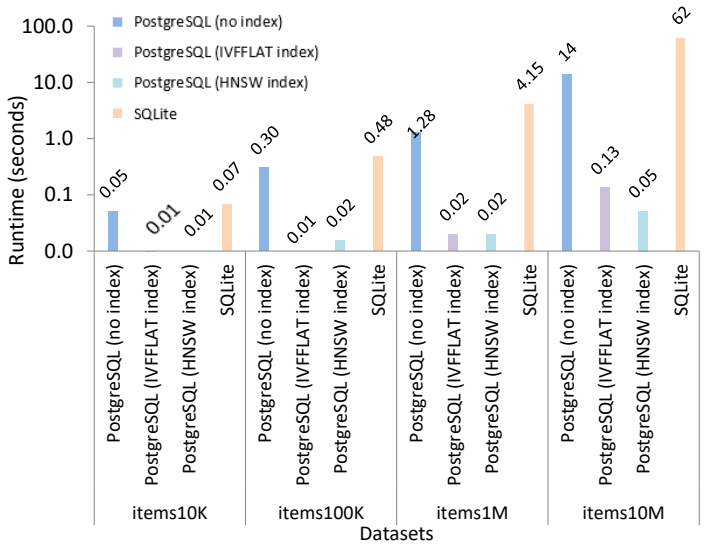


Fig. 11: Query performance (PostgreSQL vs. SQLite) with different data sizes

with the items10M table, PostgreSQL (HNSW) is slightly faster than that of PostgreSQL (IVFFlat).

5 Conclusion and future research directions

We developed an AI-assisted personalized nutrition application, as a novel use-case of multi-modal RAG search with VLM and vector-relational database. We have conducted a systematic evaluation of several aspects of current VLM systems and two vector-relational database PostgreSQL (with pgvector) and SQLite (with vector). Our experimental study reveals several issues that are summarized next. (1) The performance of embedding models, in terms of embedding generation latency, varies widely and it depends on factors such as context length and the number of model parameters. (2) The latency of VLM models' prompt (chat) response varies widely as well, depending on the number of model parameters and other factors. Moreover, a longer context length generally implies a better prompt response quality. (3) The supported vector features in the evaluated vector-relational databases differ significantly, in terms of dedicated vector data type, query syntax, storage constraint and query performance. If vector indexing is required, PostgreSQL limits the vector embedding dimension length to a maximum of 2000. Therefore, it can only index vector embeddings with dimension length less than 2001. (4) Vector indexes can significantly improve (reduce)

query latency. However, index construction time, such as with HNSW, can be substantially higher than that of IVFFLAT. (5) The vector embedding generation and high quality prompt response processes can be significantly time-consuming with the more advanced VLMs, even with consumer-grade GPUs. It is hoped that these findings will provide some guidance to the practitioners.

Future work will involve a more detailed evaluation of the prompt response quality and retrieval effectiveness with appropriate metrics. Another future work is to evaluate more complex queries with RAGs, including multi-hop queries [34]. Exploring other modalities, besides text and image, is another area of future investigation. The evaluation of the energy efficiency of different approaches, particularly of embedding generation and prompt response, is also a potential direction of future work.

References

1. Learning transferable visual models from natural language supervision. In: ICML. vol. 139, pp. 8748–8763 (2021)
2. World health organization (2022), WHO Manual on Sugar-Sweetened Beverage Taxation Policies to Promote Healthy Diets; World Health Organization, Geneva
3. ChatGPT (2026), https://chatgpt.com
4. ChromaDB (2026), https://www.trychroma.com/
5. CLIP-ViT-B-32 (2026), https://huggingface.co/sentence-transformers/clip-ViT-B-32
6. ConcatMLP (2026), https://github.com/Cadene/block.bootstrap.pytorch?tab=readme-ov-fileconcatMLP
7. epicurious (2026), https://www.epicurious.com/
8. Food Ingredients and Recipes Dataset with Images (2026), https://www.kaggle.com/datasets/pes12017000148/food-ingredients-and-recipe-dataset-with-images/
9. Increase max vectors dimension limit for index (2026), https://github.com/pgvector/pgvector/issues/461
10. IVFFlat (2026), https://github.com/pgvector/pgvector#ivfflat
11. Llama (2026), https://huggingface.co/meta-llama
12. Llava-llama3: Large language and vision assistant with llama3 (2026), https://github.com/Michel-liu/LLava-LLama3
13. Milvus (2026), https://milvus.io/
14. moondream (2026), https://github.com/vikhyat/moondream
15. Ollama (2026), https://ollama.com/
16. pgvector (2026), https://github.com/pgvector/pgvector
17. Pinecone (2026), https://www.pinecone.io/
18. Qdrant (2026), https://qdrant.tech/
19. Qwen2.5-vl vision-language model series based on qwen2.5 (2026), https://huggingface.co/collections/Qwen/qwen25-vl
20. SQLite Vector (2026), https://github.com/sqliteai/sqlite-vector
21. The Food Database (FOODB) (2026), https://foodb.ca/
22. Weaviate (2026), https://weaviate.io/platform
23. What is a context window? (2026), https://www.ibm.com/think/topics/context-window

24. Aumüller, M., Bernhardsson, E., Faithfull, A.: Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Information Systems **87** (2020)
25. Banerjee, S., Lavie, A.: METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In: ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. pp. 65–72 (Jun 2005)
26. Bordes, F., et al.: An introduction to vision-language modeling (2024), https://arxiv.org/abs/2405.17247
27. Huang, Y., Lv, T., Cui, L., Lu, Y., Wei, F.: Layoutlmv3: Pre-training for document ai with unified text and image masking (2022), https://arxiv.org/abs/2204.08387
28. Kang, G., Ge, Z., Hu, J., Zhang, X., Wang, L., Zhan, J.: Bigvectorbench: Heterogeneous data embedding and compound queries are essential in evaluating vector databases. Proceedings of the VLDB Endowment **18**(5), 1536–1550 (2025)
29. Kim, J., On, K.W., Lim, W., Kim, J., Ha, J., Zhang, B.: Hadamard product for low-rank bilinear pooling. CoRR **abs/1610.04325** (2016)
30. Lewis, P., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. In: NeurIPS (2020)
31. Malkov, Y.A., Yashunin, D.A.: Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs . IEEE TPAMI **42**(04) (2020)
32. Mohbat, F., Zaki, M.J.: Llava-chef: A multi-modal generative model for food recipes. In: CIKM. p. 1711–1721 (2024)
33. Nussbaum, Z., Morris, J.X., Duderstadt, B., Mulyar, A.: Nomic embed: Training a reproducible long context text embedder (2025), https://arxiv.org/abs/2402.01613
34. Osipjan, A., Khorashadizadeh, H., Kessel, A.L., Groppe, S., Groppe, J.: Graphtrace: A modular retrieval framework combining knowledge graphs and large language models for multi-hop question answering. Computers **14**(9) (2025)
35. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research **21**(140), 1–67 (2020)
36. Simhadri, H.V.: big-ann-benchmarks: Framework for evaluating ANNS algorithms on billion scale datasets
37. Tafuri, D., Latino, F.: Association of dietary intake with chronic disease and human health. Nutrients **13**(3) (2025)
38. Tanabe, H., Yanai, K.: Caloriellava: Image-based calorie estimation with multimodal large language models. In: ICPR. p. 63–75 (2025)
39. Team, G.V., Karlinsky, L., et al.: Granite vision: a lightweight, open-source multimodal model for enterprise intelligence (2025), https://arxiv.org/abs/2502.09927
40. Yin, Y., et al.: Foodlmm: A versatile food assistant using large multi-modal model. IEEE Transactions on Multimedia (2025)
41. Yu, Z., Yu, J., Xiang, C., Fan, J., Tao, D.: Beyond Bilinear: Generalized Multimodal Factorized High-order Pooling for Visual Question Answering (2017), https://arxiv.org/abs/1708.03619
42. Zhao, W.X., et al.: A survey of large language models (2023), https://arxiv.org/abs/2303.18223
43. Zhou, P., et al.: Foodsky: A food-oriented large language model that can pass the chef and dietetic examinations. Patterns **6**(5) (2025)

Measuring the Sensitivity of Classification Models with the Error Sensitivity Profile

Andrea Maurino[1 [0000−0001−9803−3668]]

Università degli studi di Milano Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Viale Sarca 336,20126 Milano, Italy
andrea.maurino@unimib.it

Abstract. The quality of training data is critical to the performance of machine learning models. In this paper, the Error Sensitivity Profile (ESP) is proposed. It quantifies the sensitivity of model performance to errors in a single feature or in multiple features. By leveraging ESP, data-cleaning efforts can be prioritized based on error types and features most likely to affect model performance. To support the computation of this metric, an integrated suite of tools, called Dirtify , is created. We conduct an extensive experimental study on two widely used datasets using 14 classification models, revealing that performance degradation is not always predictable from simple correlations with the target variable.

Keywords: Data Quality · Data Centric AI · Machine Learning performance.

1 Introduction

Multiple studies have explored the influence of data errors on the efficacy of machine learning (ML) models [10, 13, 9]. These studies suggest that, due to the pronounced nonlinearity of datasets and models, it is challenging to generalize about how data quality affects model performance. Consequently, a dataset with certain errors might perform well with one machine learning model but poorly with another. Similarly, two datasets with the same error types, when used to train the same ML model, may yield significantly different results. In this paper, we propose the Error Sensitivity Profile (ESP), a new method for assessing and comparing the impact of errors on ML performance. ESP is able to 1) detect for a given model trained on a dataset contaminated by a type of error applied to one or more features if there is a linear relationship between the percentage of errors and the performance degradation (Error Performance Correlation EPC), 2) calculate the performance that globally the model miss or gain with the increase of the error level (Area under Curve Error-Performance AEPC), and 3) the behavior of the model according to the different level of errors. With ESP, both practitioners and researchers can compare the effects of training data quality across models and datasets. To fully exploit the ESP, we developed **Dirtify** a suite of Python-based tools to aid specialists in examining the impact of specific

dataset errors on the performance of specific Machine Learning models. This is facilitated through a Python library named PuckTrick[2], which is capable of injecting predefined percentages of specific errors. The analysis of two widely used datasets across 14 ML models demonstrates the effectiveness of ESP as a tool for assessing the impact of data quality on machine learning models. The study confirms the Dirtify suite's ability to support this type of research. This paper is structured as follows: Section 2 examines the current state of the art. Section 3 introduces a formal definition of ESP, while the tool suite and its key architectural components are outlined in Section 4. Section 5 displays and analyzes the principal results from the empirical analysis. Lastly, Section 6 concludes the discussion and outlines potential future works

2 State of the Art

Prior research has explored the impact of different types of dirty data on ML models, including missing values [10, 1], inconsistent data [10], outliers [3], label noise [7], and categorical duplicates [13]. Frameworks for sensitivity analysis have been proposed in [6], but are limited to a specific domain. While these works addressed data quality issues in various ways, their analyses focused on a single error type; additionally, none have developed a flexible tool for dataset contamination and assessment comparable to Dirtify . Recently, [9] investigated the impact of six data quality dimensions across multiple ML tasks, evaluating five classification algorithms (Logistic Regression, MLP, SVM, TabNet, and KNN) on a fixed collection of datasets. Although complementary in spirit, the two approaches differ along several dimensions. First, [9] operates at the dataset level, producing aggregate scalar indicators of quality impact, whereas ESP operates at the feature level and provides a multi-dimensional profile that jointly captures the direction, cumulative magnitude, and local structure of the error–performance relationship. Second, the quality dimensions adopted in [9] conflate structurally distinct error types, for instance, outliers and noisy values are subsumed under the single dimension feature accuracy, whereas ESP treats each error type as a separate, independently controllable corruption operator, enabling finer diagnostic resolution. Third, ESP is evaluated across 14 classification models, providing broader coverage of the algorithmic landscape. These differences make ESP and [9] complementary rather than competing: [9] offers a dataset-level view of quality impact across a curated benchmark, while ESP provides a fine-grained, model-specific sensitivity profile designed to guide feature-level data cleaning decisions.

CleanML [8] evaluates the impact of cleaning algorithms applied to real-world dirty datasets, using repeated runs and FDR control to ensure statistical rigour. The ESP profile can also be computed on top of experimental protocols such as CleanML, provided that the underlying methodology generates error–performance curves at controlled corruption levels.

Among tools for data corruption, BART [4] allows users to specify error types and distributions via denial constraints, but operates only on relational tables

and does not evaluate ML tasks. Jenga [12] is a Python library that introduces data corruptions into datasets to evaluate model robustness, but tests a single model at a time and lacks a configuration interface for error strategies. The PuckTrick library [2], on which Dirtify relies, provides a broader range of error types than Jenga, including label perturbations and duplicated data.

3 Error Sensitivity Profile

Let D_0 be a cleaned training dataset (our baseline) and let $\mathcal{T}_\theta : D \to D$ be the family of corruption operators, where θ is a vector of parameters (e.g., error rate, error type, affected features). For each error rate $k = 1, \ldots, K$, we define $D_k = \mathcal{T}_{\theta_k}(D_0)$ as a corrupted training dataset. Let $f_k = A(D_k; s_k)$ be a model based on algorithm A trained on D_k with random seed s_k, and let $p_k = \mathrm{Perf}(f_k, D_{\text{test}})$ be a performance metric evaluated on a clean test set D_{test}. We define a scalar measure of the training data degradation induced by $\mathcal{T}_{\theta_k}$ as $e_k = \mathrm{Err}_\tau(D_k, D_0)$, where τ denotes the error type (e.g., label noise, missing values, outliers), and $\mathrm{Err}_\tau : D \times D \to \mathbb{R}^+$ is an error-specific degradation function that quantifies the severity of the corruption with respect to the clean baseline D_0. In the experimental evaluation presented in this paper, $\mathrm{Err}_\tau(D_k, D_0)$ is instantiated as the percentage of corrupted values injected into the training set, i.e. $e_k \in \{0, 20, 40, 60, 80\}\%$, which provides a transparent and reproducible operationalisation of the corruption severity. Given a learning algorithm A, a baseline training dataset D_0, a corruption type τ, and an associated family of corrupted datasets $\{D_k\}_{k=1}^K$, we define the *Error Sensitivity Profile* of A with respect to τ as the tuple:

$$\mathrm{ESP}(A, D_0, \tau) = \Big\langle \mathrm{EPC},\ \mathrm{AEPC},\ \{\hat{\beta}_j\}_{j=1}^J \Big\rangle, \tag{1}$$

where each component isolates a specific, complementary facet of the error–performance relationship, as detailed below.

3.1 Error Performance Correlation

The *Error Performance Correlation* (EPC) provides a global, first-order summary of the relationship between data corruption and model performance. Specifically, EPC is defined as the negated Pearson correlation coefficient between the error severity sequence $\{e_k\}$ and the corresponding performance sequence $\{p_k\}$:

$$\mathrm{EPC} = -\,\rho(\{e_k\},\, \{p_k\})\,, \tag{2}$$

where $\rho(\cdot, \cdot)$ denotes the Pearson correlation coefficient. The negation is introduced so that a *positive* EPC value indicates the expected inverse relationship between error level and model performance: as corruption increases, performance decreases. Conversely, a negative EPC signals that performance improves as errors accumulate, a counterintuitive but empirically observed phenomenon (see

Section 5). An EPC value close to zero indicates the absence of a globally linear relationship between error and performance. It is important to note that EPC is intentionally designed as a lightweight first-order diagnostic: it efficiently detects whether a linear trend dominates the error–performance curve, but it is not intended to provide a complete characterisation on its own. When EPC is close to zero, the relationship is non-linear or non-monotonic, and the remaining components of the ESP, AEPC and the piecewise slope vector, become the primary diagnostic tools.

3.2 Area under the Error–Performance Curve

The *Area between the Error–Performance curves* (AEPC) quantifies the cumulative magnitude of the deviation from baseline performance across all error levels. To ensure comparability across models and datasets with different baseline performance levels, AEPC is defined as the normalised integral of the signed vertical distance between the baseline performance p_0 and the observed performance curve:

$$\mathrm{AEPC} = \frac{\displaystyle\int_0^{e_{\max}} \big(p(e) - p_0\big)\,\mathrm{d}e}{p_0 \cdot e_{\max}}, \tag{3}$$

where $p(e)$ is the performance observed at error level e, p_0 is the baseline performance on the uncorrupted dataset, and $e_{\max}$ is the maximum corruption level considered. The denominator $p_0 \cdot e_{\max}$ represents the area subtended by the baseline performance over the full corruption range, so that AEPC expresses the cumulative deviation as a fraction of the baseline contribution. A value of $\mathrm{AEPC} = -0.10$, for instance, indicates that the model lost, on average, 10% of its baseline performance across all corruption levels. It is worth noting that AEPC is meaningful only when computed with respect to a consistent performance metric: comparisons across tasks or metrics (e.g., F1 vs. accuracy) are not defined, and in the canonical representation, the metric adopted is always reported explicitly. Regarding numerical stability, the normalisation in Equation 3 is well-defined provided $p_0 > 0$. In the degenerate case $p_0 \approx 0$, the normalisation becomes numerically unstable; however, this scenario implies that the model achieves negligible baseline performance on the uncorrupted dataset, a condition under which the model itself is not a viable candidate for deployment, and a sensitivity analysis with respect to data quality would be of no practical relevance. We therefore consider this case outside the intended scope of ESP. A positive AEPC indicates that, on average, the model performs better than the baseline across the corruption range, whereas a negative AEPC reflects a net degradation. Notice that, as with EPC, AEPC provides a global summary and may obscure local non-monotonic behaviour; the piecewise slope vector is therefore essential for a complete interpretation. While EPC captures the direction and consistency of the error–performance response, AEPC quantifies its overall relative extent, providing complementary information especially in non-monotonic regimes where EPC loses descriptive power.

3.3 Piecewise Slope Vector

To capture the local structure of the error–performance curve, we partition the error axis into J locally monotonic regions and estimate a slope $\hat{\beta}_j$ within each region $j = 1, \ldots, J$. The resulting *piecewise slope vector* $\{\hat{\beta}_j\}_{j=1}^{J}$ reveals local regimes, critical thresholds, and saturation phenomena that neither EPC nor AEPC can express. Regime boundaries are identified by detecting sign changes in the first-order differences of the performance sequence $\{p_k\}_{k=1}^{K}$: a boundary is placed at position k^* whenever $\text{sign}(p_{k^*} - p_{k^*-1}) \neq \text{sign}(p_{k^*+1} - p_{k^*})$. Within each resulting monotonic region $\mathcal{R}_j = \{k : k_j \leq k \leq k_{j+1}\}$, the slope $\hat{\beta}_j$ is estimated by ordinary least squares (OLS) regression of p_k on e_k:

$$\hat{\beta}_j = \frac{\sum_{k \in \mathcal{R}_j} (e_k - \bar{e}_j)(p_k - \bar{p}_j)}{\sum_{k \in \mathcal{R}_j} (e_k - \bar{e}_j)^2}, \tag{4}$$

where $\bar{e}_j$ and $\bar{p}_j$ denote the mean error level and mean performance within region $\mathcal{R}_j$ respectively. A negative $\hat{\beta}_j$ indicates performance degradation within that regime, while a positive value signals improvement.

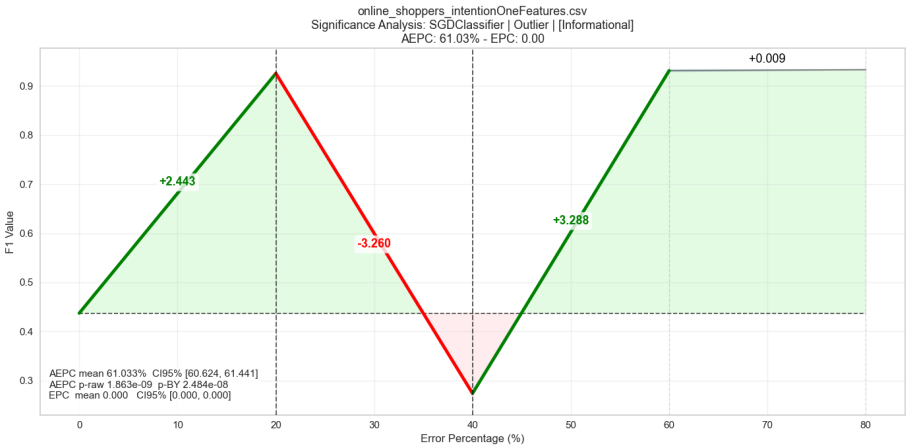


Fig. 1: Canonical representation of ESP

It should be noted that, given the five corruption levels used in this study, slope estimates in short monotonic regions may be computed on as few as two observations, yielding exact fits with no residual degrees of freedom; such estimates should therefore be interpreted as directional indicators rather than statistically reliable regression coefficients, and increasing the number of corruption levels remains a direction for future work

3.4 Canonical Representation

To graphically represent and compare different ESPs, we adopt the canonical representation illustrated in Figure 1. The performance–error curve is annotated with the global dependence (EPC), the cumulative degradation (AEPC), and the piecewise slopes identifying local regimes.

This unified visual form allows practitioners to assess, at a glance, both the overall sensitivity of a model to data quality issues and the structural features of its error–performance response. Since each experiment is repeated $N = 30$ times with different random seeds (see Section 5), in practice, we compute an ESP for each independent run $n = 1, \ldots, N$ and report the aggregate ESP, whose components are defined as:

$$\overline{\mathrm{EPC}} = \frac{1}{N}\sum_{n=1}^{N} \mathrm{EPC}^{(n)}, \qquad \overline{\mathrm{AEPC}} = \frac{1}{N}\sum_{n=1}^{N} \mathrm{AEPC}^{(n)}, \qquad \bar{\hat{\beta}}_j = \frac{1}{N}\sum_{n=1}^{N} \hat{\beta}_j^{(n)}. \tag{5}$$

Confidence intervals at 95% are reported alongside each aggregate component to quantify estimation uncertainty across runs, as shown in the canonical representation of Figure 1.

4 Dirtify

To support the computation of the ESP, we developed Dirtify, a suite of three loosely coupled Python-based tools. The *Configurator* is a wizard-style application that guides the user in defining an error strategy and serialises it as a JSON configuration file. To support users in designing a strategy, the Configurator provides two predefined error strategies and allows users to define their own. The first predefined strategy, referred to as *one-feature-at-a-time*, applies each error type to each user-selected feature independently, enabling the user to isolate the contribution of each feature to model performance degradation. The second predefined strategy, referred to as *correlated-features*, applies each error type to groups of features whose pairwise correlation exceeds a user-defined threshold, modelling the realistic scenario in which corrupting a single feature has limited impact due to redundancy with correlated features. Users may also define custom strategies by specifying arbitrary error functions parametrised by error type, target features, corruption mode, injection distribution, and a predicate for selecting the subset of D_0 to be modified, enabling fine-grained control over which instances are subject to corruption.

The *Trainer* reads the JSON configuration produced by the Configurator and executes as follows: for each combination of error type, corrupted feature, and error level e_k, it corrupts the baseline dataset D_0 via the PuckTrick library [2] and trains each selected ML model using PyCaret[1], storing the resulting performance metrics. The precise operational definition of each error type, including injection distributions, parametrization, and categorical handling, is documented in the

[1] `https://pycaret.org/`

Label	model Name	Label	model Name
SVM	SVM - Linear Kernel	NB	Naive Bayes
ET	Extra Trees Classifier	DT	Decision Tree Classifier
RF	Random Forest Classifier	QDA	Quadratic Discriminant Analysis
KN	K Neighbors Classifier	SGD	Stochastic Gradient Descent Classifier
LDA	Linear Discriminant Analysis	RC	Ridge Classifiers
MLP	Multilayer Perception Classifier	ADA	Ada Boost
LR	Logistic Regression	XG	XGBoost

Table 1: Machine learning algorithms used by the Dirtify suite.

PuckTrick library [2]. Each experimental configuration is repeated $N = 30$ times with different random seeds. For each repetition, the training/test split is generated once using an 80–20% ratio with Stratified K-Folds cross-validation and held fixed across all corruption levels within that repetition. Crucially, corruption is applied exclusively to the training set: D_{test} is never corrupted, ensuring that observed variations in p_k are attributable solely to the degradation of training data quality and not to test-set resampling variance. The list of supported ML algorithms is shown in Table 1. Hyperparameters are set to PyCaret defaults (see Table 2) and held fixed across all corruption levels and repetitions, ensuring that observed performance variations are attributable solely to training data degradation and not to model recalibration.

Imputation type	simple
Numeric imputation	means
Categorical imputation	mode
Fold Generator	Stratified K-Folds
Fold Number	10

Table 2: Hyperparameter and Configuration setting

While fixing hyperparameters to their defaults ensures that observed performance variations are attributable solely to training data degradation, it also implies that models sensitive to configuration, such as MLP and XGBoost, may exhibit ESP profiles that partially reflect suboptimal parametrisation rather than intrinsic sensitivity to data quality, a confound that future work should address through hyperparameter-controlled experimental designs.

5 Case study

To illustrate how data quality impacts model performance and to demonstrate the insights provided by the proposed sensitivity profile and the Dirtify suite[2], we

[2] `https://anonymous.4open.science/r/dirtify-4730`

consider the Online Shoppers Purchasing Intention dataset[11], publicly available through the UCI Machine Learning Repository. The dataset comprises 12,330 sessions from distinct users, spanning 17 numerical and categorical features, including web analytics metrics such as bounce rates, exit rates, and page values, as well as contextual variables such as visit timing and visitor category, with no missing values and a well-defined binary target variable (*Revenue*), making it well-suited for classification without extensive preprocessing. Feature importance scores, computed using a Random Forest classifier (accuracy: 89.7%), reveal a strongly skewed distribution: *PageValues* dominates (0.383), followed by *ProductRelated_Duration* (0.088), *ExitRates* (0.086), *ProductRelated* (0.073), and *BounceRates* (0.059), while demographic and technical attributes contribute negligibly. Pearson correlation analysis identifies two highly correlated feature pairs — *BounceRates*/*ExitRates* (r = 0.913) and *ProductRelated*/*ProductRelated_Duration* (r = 0.861) — suggesting multicollinearity, while *PageValues* shows the strongest association with *Revenue* (|r| = 0.493). Given the significant class imbalance (84.5% non-purchase vs. 15.5% purchase), the F1-score was adopted as the primary evaluation metric. The one-feature-at-a-time strategy was applied, contaminating each feature independently with noisy values and outliers. The same error types were applied to correlated feature groups ($r \geq 0.5$), following the correlated-features strategy. Mislabeling, duplication, and a custom oversampling strategy, duplicating only rows where the target equals 0, were also investigated. In all strategies, corruption was applied incrementally from 0 to 80% in steps of 20%, with 30 independent runs per combination of model, error type, feature, and corruption level. Overall, 21840 models were trained across 728 distinct scenarios, where a scenario is defined as a specific combination of error type, corrupted feature(s), and ML model.

Before analysing the ESP components, we applied a two-stage filtering procedure to identify scenarios that are both statistically and practically relevant, and to make the analysis tractable, given the large number of experimental configurations. In the first stage, statistical significance was assessed for each scenario independently. For each scenario, we compared the distribution of performance values obtained across the 30 independent runs at the maximum corruption level against the corresponding baseline distribution using a Wilcoxon signed-rank test, with a significance level of $\alpha = 0.05$. This non-parametric test was preferred over a standard t-test as it does not assume normality of the performance distributions, which cannot be guaranteed across all models and corruption types.To address the multiple testing problem arising from the simultaneous evaluation of all scenarios, raw p-values were corrected using the Benjamini-Yekutieli (BY) procedure [5] at FDR level $\alpha = 0.05$, which provides FDR control under arbitrary dependence among tests. In the second stage, among the scenarios that passed the statistical significance criterion, we retained only those for which the absolute value of the normalised $\overline{\text{AEPC}}$ exceeded a practical relevance threshold: $|\overline{\text{AEPC}}| > \delta, \quad \delta = 0.05$. This threshold corresponds to a cumulative deviation greater than 5% of the baseline performance across the full corruption range. Of the 728 scenarios, only 44, approximately 6% of

the total, were significant. Deviations below this value were deemed negligible for practical data-cleaning purposes, as they would not meaningfully influence the decision about whether to invest resources in cleaning a given feature. To verify the robustness of our findings with respect to this choice, we repeated the analysis with $\delta \in \{0.03, 0.10\}$, obtaining 115 and 36 significant scenarios respectively The qualitative findings reported below remain stable across all three thresholds, confirming that the choice of $\delta = 0.05$ does not introduce a material bias in the conclusions. Among these scenarios, the model that appears most frequently, i.e., the one exhibiting the most significant changes, is the **SGD Classifier**, with 34 occurrences. Only two scenarios involve the correlated feature pair *[ExitRates, BounceRates]*, and in both cases the model affected is the **Quadratic Discriminant Analysis**. Among the top four features by importance, the first two appear in 6 scenarios each; *ExitRates* appears only in one scenario, while *ProductRelated* and *BounceRates* each appear individually in 2 scenarios. It is worth noting that contamination of certain variables, such as `Month`, `Weekend`, and `Administrative_Duration`, can significantly affect model performance, even though these variables are not among the most important features. Considering $\overline{\text{AEPC}}$, we observe that in 91% of significant scenarios, the value is positive, meaning that in 40 out of 44 significant scenarios, corrupting the data increases model performance. Among others, in the scenario shown in Figure 2, we notices that the SGD trained on a dataset where the `Administrative_Duration` is affected by outliers produces an F1 score that improves sharply and stabilizes up to 60% error rate, as confirmed by the strongly positive $\overline{AEPC}$ (86.87%), indicating an overall beneficial effect of the outlier corruption on this model. However, beyond this threshold, the performance collapses dramatically. In the scenario of Figure 3, where the MLP Classifier was trained on the same dataset used in the previous scenario, we observe that F1 exhibits behavior that is virtually specular to that of the previous one, even though the dataset is the same. In fact, a sharp performance drop up to 40% error rate followed by a substantial recovery beyond the 60% threshold, where the model progressively compensates for the introduced noise, almost fully restoring its baseline F1 score.

To provide preliminary evidence of generalisability, we apply ESP to the South German Credit dataset[3]: 1,000 credit records with 20 features and a binary target (`kredit`), moderate class imbalance (70/30%), and dominant features `hoehe` (0.131) and `laufkont` (0.116). Using the same experimental protocol, we obtained 980 scenarios, of which 77 (≈8%) were significant; **SGD** again dominates with 47 occurrences, and 86% of significant scenarios exhibit a positive $\overline{\text{AEPC}}$. Table 3 summarises findings across both datasets: the vast majority of models exhibit remarkable robustness, while SGD accounts for 81 of 121 significant scenarios overall, a vulnerability attributable to its incremental update mechanism, which lacks the averaging effect of ensemble methods.

Notably, low-importance features, such as `Administrative_Duration` and `TrafficType` in the first dataset can significantly destabilize SGD when heavily

[3] `https://archive.ics.uci.edu/dataset/522/south+german+credit`

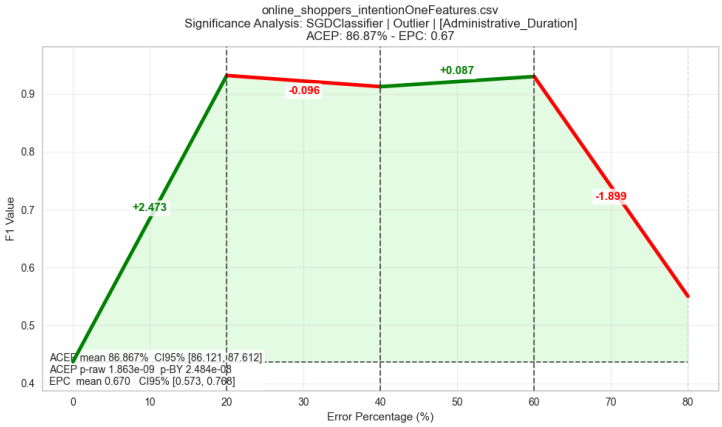


Fig. 2: Relevant scenario where $\overline{\text{AEPC}}$ is positive

Table 3: Comparative summary of ESP analysis across datasets.

	Online Shoppers	South German Credit
Total scenarios	728	980
Significant scenarios	44 (6%)	77 (8%)
Most sensitive model	SGD (34/44)	SGD (47/77)
Positive $\overline{\text{AEPC}}$ (%)	91%	86%
Top feature (importance)	PageValues (0.383)	hoehe (0.131)
Correlated pairs ($r \geq 0.5$)	2	1

corrupted, suggesting that feature importance scores computed on clean data may underestimate the disruptive potential of marginal features under quality degradation. The error–performance relationship is rarely monotonic, as illustrated in Figures 2 and 3, motivating the joint use of $\overline{\text{AEPC}}$ and $\overline{\text{EPC}}$ to capture both the magnitude and directionality of performance changes that neither metric alone could fully characterise. Furthermore, the counterintuitive prevalence of positive $\overline{\text{AEPC}}$ values is confirmed on both datasets, in 91% and 86% of significant scenarios, respectively, suggesting that this phenomenon is not dataset-specific. We conjecture that this phenomenon is partially attributable to class imbalance. Since the error types that produce significant scenarios involve corruptions applied exclusively to feature values rather than to class labels, the class distribution in the training set remains unchanged by construction. Nevertheless, injected outliers may alter the geometry of the feature space in ways that inadvertently improve the separability of minority-class instances, effectively acting as an implicit resampling mechanism. This hypothesis is consistent with the observed difference between the two datasets (91% vs. 86% of positive $\overline{\text{AEPC}}$ scenarios), where the less imbalanced South German Credit dataset exhibits a

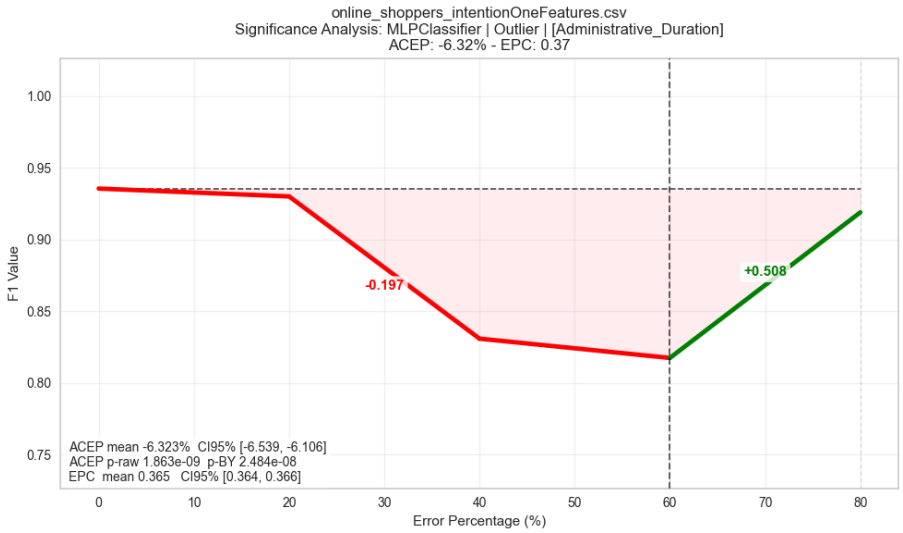


Fig. 3: Relevant scenario where $\overline{\text{AEPC}}$ is negative

moderately lower prevalence of the effect, though a formal verification is left to future work.

6 Conclusion

It is widely recognized that the quality of training data is crucial to the success of machine learning models. In this paper, we introduce the Error Sensitivity Profile and present Dirtify, an all-encompassing tool suite for systematically evaluating the impact of data quality on classification tasks. While Dirtify facilitates a structured analysis of how various types of data degradation affect model performance, the Error Sensitivity Profile aids ML practitioners and researchers in determining the most robust model for a specific dataset and the resilience of ML models against data quality issues. We also present two significant applications of our approach by analysing well-known and complex datasets. The extensive experimental evaluation of about 51200 trained models (approximately 21800 for the first dataset and almost 29400 for the second) demonstrates the importance of empirically assessing the impact of data quality, rather than assuming that corruption invariably degrades performance. While the experimental evaluation presented in this paper focuses on classification tasks, the ESP framework is designed to be task-agnostic: the performance function $\text{Perf}(f_k, D_{\text{test}})$ can be instantiated with any suitable metric, such as RMSE or MAE for regression, or silhouette score for clustering. The extension of Dirtify to these settings is facilitated by the native support for regression and clustering pipelines in the underlying PyCaret library. Future work will extend the evaluation to a broader

set of datasets; moreover we intend to expand Dirtify along four directions: introducing new error types, such as data drift, incorporating deep learning models, optimizing PuckTrick to scale to high-dimensional datasets, and investigating intelligent feature pre-screening strategies, based on importance scores, variance, or target correlation, to make the exhaustive ESP analysis tractable when hundreds of features are involved.

References

1. Adnan, F.A., et al.: A review of the current publication trends on missing data imputation over three decades: direction and future research. Neural Comput. Appl. **34**(21), 18325–18340 (Nov 2022)
2. Agostini, A., Sphaiu, B., Maurino, A.: Pucktrick: A library for making synthetic data more realistic. In: SEBD. `https://arxiv.org/abs/2506.18499`
3. Ansari, S., et al.: Impact of outliers on regression and classification models: An empirical analysis. In: DeSE. pp. 211–218 (2024)
4. Arocena, P.C., et al: Messing up with bart: error generation for evaluating data-cleaning algorithms. Proc. VLDB Endow. **9**(2), 36–47 (Oct 2015)
5. Benjamini, Y., Yekutieli, D.: The control of the false discovery rate in multiple testing under dependency. The Annals of Statistics **29**(4), 1165–1188 (2001)
6. Dix, M., et al.: Measuring the robustness of ML models against data quality issues in industrial time series data. In: INDIN. pp. 1–8. IEEE (2023)
7. Frenay, B., Verleysen, M.: Classification in the presence of label noise: A survey. IEEE Trans. on Neural Networks and Learning Systems **25**(5), 845–869 (2014)
8. Li, P., et al: Cleanml: A benchmark for joint data cleaning and machine learning [experiments and analysis]. arXiv preprint arXiv:1904.09483 **75** (2019)
9. Mohammed, S., et al: The effects of data quality on machine learning performance on tabular data. Information Systems **132**, 102549 (2025)
10. Qi, Z., Wang, H., Dong, Z.: Impacts of Dirty Data on Classification and Clustering Models, pp. 7–37. Springer Nature Singapore, Singapore (2024)
11. Sakar, C., Kastro, Y.: Online shoppers purchasing intention dataset. UCI Machine Learning Repository (2018), licensed under CC BY 4.0
12. Schelter, S., Rukat, T., Biessmann, F.: Jenga: a framework to study the impact of data errors on the predictions of machine learning models (2021)
13. Shah, V., et al.: How do categorical duplicates affect ml? a new benchmark and empirical analyses. Proc. VLDB Endow. **17**(6), 1391–1404 (Feb 2024)

Multi-Perspective Credibility Adjustment Using Aspect-Based Sentiment Analysis for Robust Retrieval-Augmented Generation

Masahide Okochi[0009−0001−4604−985X], Israel Mendonça[0000−0001−6819−4305], Thanda Shwe[0000−0002−8052−2321], and Masayoshi Aritsugi[0000−0003−0861−849X]

Kumamoto University, Japan

Abstract. Large Language Models (LLMs) augmented with Retrieval-Augmented Generation (RAG) can ground their responses in external documents, reducing hallucinations. However, when the external corpus itself contains misinformation, RAG can amplify rather than suppress factual errors. Existing countermeasures assign credibility scores to retrieved documents based on the LLM's internal knowledge or query-document relevance, which is insufficient when the LLM lacks domain expertise or when misinformation documents exhibit high relevance to the query. We propose a multi-perspective credibility adjustment method that enriches conventional credibility estimation with Aspect-Based Sentiment Analysis (ABSA). Our approach exploits an empirical observation: misinformation within documents tends to be expressed in an optimistic, confident tone, whereas accurate information is described with cautious, hedging language. By combining three complementary signals: LLM-assigned credibility, query-document relevance, and aspect-level sentiment; our method adjusts document credibility to more precisely suppress the influence of misinformation while preserving access to valuable information within partially unreliable documents. Experiments on open-domain question-answering tasks across multiple LLMs and datasets demonstrate that our method consistently outperforms conventional credibility adjustment approaches and maintains robust performance even as the proportion of misinformation increases.

Keywords: Retrieval-Augmented Generation · Aspect-Based Sentiment Analysis · Misinformation handling.

1 Introduction

Large Language Models (LLMs) are prone to hallucinations, generating plausible but factually incorrect outputs [8,15,21]. Retrieval-Augmented Generation (RAG) mitigates this problem by grounding LLM responses in documents retrieved from external corpora [4,25,27]. However, a critical but underexplored assumption in most RAG research is that the external corpus itself is trustworthy. In practice, external corpora may contain documents with misinformation, and it has been shown that even a single such document can substantially degrade LLM performance [17,19].

Recent work has addressed this vulnerability by incorporating document credibility into RAG. Pan et al. [18] proposed fine-tuning LLMs on credibility-annotated datasets, while Hong et al. [6] trained a classifier to estimate the probability that a document contains misinformation. More recently, CrAM [3] introduced a training-free approach that modifies the LLM's attention weights during inference based on credibility scores assigned by the LLM itself. While these methods represent important progress, they share a fundamental limitation: credibility is estimated from a single perspective. Specifically, existing approaches rely primarily on either the LLM's internal knowledge or the relevance between the document and the query. Both perspectives are insufficient on their own. LLM-based credibility assessment fails when the model lacks knowledge about the topic, potentially assigning high credibility to misinformation [3]. Relevance-based assessment is equally problematic because documents containing misinformation—particularly those generated by LLMs—tend to exhibit high relevance to the query, making them difficult to distinguish from reliable documents [12,7,19]. Furthermore, assigning uniformly low credibility to an entire document simply because it contains some misinformation is counterproductive: as described in [9], LLMs can resolve contradictions between correct and incorrect information, meaning that valuable information within such documents may be discarded unnecessarily [3].

In this paper, we propose a multi-perspective credibility adjustment method that addresses these limitations by incorporating Aspect-Based Sentiment Analysis (ABSA) as an additional signal. Our approach is grounded in an empirical observation: within documents containing misinformation, the misinformation itself tends to be described in an optimistic and confident tone (e.g., "X is definitely the case"), whereas accurate information is typically expressed with more cautious, hedging language (e.g., "X is likely to be the case"). This tonal asymmetry, which we validate experimentally in Section 3, provides a complementary perspective for credibility assessment that is independent of both LLM internal knowledge and query-document relevance.

Concretely, given a question and its retrieved documents, we first extract aspects from each document using an LLM and classify the sentiment toward each aspect as optimistic, pessimistic, or neutral. We then combine these sentiment signals with the query-document relevance to adjust the initial credibility scores assigned by the LLM. The adjusted credibility is used to modify the LLM's attention weights during inference, following the CrAM framework [3]. By integrating three complementary perspectives—LLM-assigned credibility, query-document relevance, and aspect-level sentiment—our method achieves a more nuanced evaluation of document credibility that not only suppresses the influence of misinformation but also preserves access to valuable information within partially unreliable documents.

The main contributions of this paper are as follows:

– We empirically demonstrate that documents containing misinformation exhibit systematic tonal differences: misinformation is described optimistically while accurate information is described pessimistically.

- We propose a credibility adjustment method that combines aspect-level sentiment with query-document relevance to refine LLM-assigned credibility scores, enabling more precise suppression of misinformation while retaining useful information.
- We conduct extensive experiments across multiple datasets and LLMs, demonstrating that our method consistently outperforms conventional credibility adjustment approaches and exhibits robust performance even as the proportion of misinformation increases.

2 Related Work

2.1 Misinformation Detection

Research on misinformation detection spans fake news detection [11], rumor detection [5,26], and factuality assessment [16]. Early approaches relied on feature engineering and classical machine learning classifiers, but pre-trained language models such as BERT and RoBERTa substantially improved detection performance. For instance, [11] demonstrated the effectiveness of BERT for fake news detection on social media.

The emergence of LLMs has reframed misinformation detection as an instruction-following or zero-shot task. Pelrine et al. [20] showed that GPT-4 can numerically evaluate the truthfulness of sentences, achieving higher accuracy than BERT- or RoBERTa-based methods. Min et al. [16] proposed FActScore, which decomposes generated text into atomic facts and verifies each individually, enabling fine-grained factuality assessment.

Our work draws on advances by using LLM-based credibility scoring [20,16] as the starting point for document-level credibility estimation. However, rather than treating misinformation detection as a binary classification task, we use it as one component of a multi-perspective credibility adjustment framework.

2.2 Countermeasures against Misinformation in RAG

RAG is effective at suppressing hallucinations by grounding LLM responses in retrieved documents [4,25], but this benefit is undermined when the external corpus itself contains misinformation [19]. Existing countermeasures can be broadly categorized into training-based and inference-time approaches.

Among training-based methods, Pan et al. [18] proposed fine-tuning LLMs on credibility-annotated datasets so that generated responses incorporate credibility-based reasoning. Hong et al. [6] trained a separate classifier to estimate the probability that a document contains misinformation, and conditioned GPT-3.5's generation on these probabilities. While effective, both approaches require constructing specialized training data and incur additional computational cost, limiting their flexibility when the corpus or domain changes.

To avoid these costs, Deng et al. [3] proposed CrAM, a training-free method that modifies the LLM's attention weights during inference based on credibility

scores. CrAM assigns credibility using the LLM's internal knowledge and suppresses attention to low-credibility documents without requiring any fine-tuning.

Our method is built directly on CrAM as its inference mechanism. However, CrAM relies on a single perspective for credibility estimation, the LLM's own judgment, which can be unreliable when the LLM lacks domain knowledge or when misinformation documents exhibit high query relevance. In contrast, prior work estimates document credibility by combining multiple signals, including query-document relevance, document timestamp, and source reliability [18]. While effective, such an approach often relies on external metadata or curated resources, which may not always be readily available or may require additional preprocessing. In comparison, our approach leverages aspect-level sentiment, which can be obtained directly from a pretrained LLM without requiring explicit annotations or external metadata. This enables a lightweight and easily deployable alternative for enriching credibility estimation. Our contribution is to enrich this credibility signal with aspect-level sentiment and query-document relevance before it is used to modify attention weights.

2.3 Aspect-Based Sentiment Analysis

Aspect-Based Sentiment Analysis (ABSA) aims to identify the sentiment expressed toward specific aspects within a text. Recent work has demonstrated that LLMs perform competitively on ABSA tasks. Zhang et al. [28] showed that LLMs outperform smaller models like BERT in few-shot sentiment classification and ABSA settings. Simmering and Huoviala [23] evaluated GPT-4 and GPT-3.5 under zero-shot, few-shot, and fine-tuned configurations, confirming ABSA performance across settings. In a practical application, Alkhnbashi et al. [1] employed LLM-based ABSA to analyze patient feedback, extracting sentiments toward aspects such as medication and communication with medical staff.

These results establish that LLMs are reliable ABSA tools, which motivates our use of LLM-based ABSA as a component of credibility assessment. To our knowledge, our work is the first to apply ABSA not for understanding opinions or feedback, but as a signal for detecting tonal patterns associated in misinformation in retrieved documents. This novel application of ABSA connects two previously separate lines of research (misinformation handling in RAG and aspect-level sentiment analysis) into a unified framework for credibility adjustment.

3 Preliminary Experiment

Before presenting our method, we empirically verify the key assumption underlying our approach: that misinformation and accurate information within the same document exhibit systematically different tonal characteristics that can be detected through ABSA.

Setup. We analyze misinformation documents from [3], which were generated by prompting an LLM with a question, its correct answer, and a target misinformation term. Each resulting document therefore contains both accurate information and misinformation with respect to the question. We perform ABSA on these documents using three aspect categories: the accurate answer to the question, the misinformation term, and "Not applicable" (used when no relevant aspect can be extracted, as detailed in Section 5.6 and Table 4). For each document-aspect pair, an LLM classifies the sentiment as optimistic, pessimistic, or neutral.

Results. Figure 1 shows the sentiment distributions across the Natural Questions (NQ) [13] and TriviaQA [10] datasets. Two clear patterns emerge. First, when the aspect corresponds to misinformation, optimistic sentiment is overwhelmingly dominant, whereas when the aspect corresponds to accurate information, pessimistic sentiment dominates. This confirms our hypothesis: misinformation tends to be expressed with confident, assertive language, while accurate information is described with more cautious, hedging expressions. Second, when the aspect is labeled "Not applicable" (i.e., the aspect is absent from the document), the LLM most frequently predicts neutral sentiment, though optimistic predictions occur more often than in the accurate-information case. This suggests that when no specific aspect is present, the LLM falls back on analyzing the overall tone of the document rather than performing aspect-level evaluation.

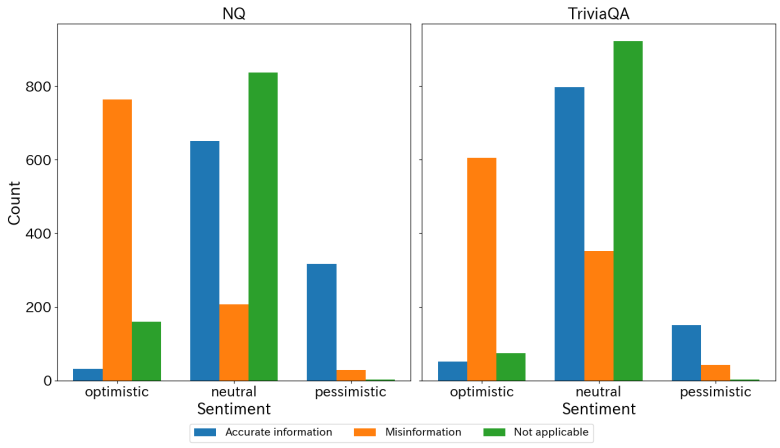


Fig. 1. Sentiment tendencies toward misinformation and accurate information within documents containing misinformation. NQ means Natural Questions [13]. TriviaQA means TriviaQA [10].

Implications for our method. These findings provide the empirical foundation for credibility adjustment proposed in Section 4. The tonal asymmetry between misinformation and accurate information shows that sentiment polarity is a signal

for distinguishing between the two, complementing the LLM-assigned credibility and query-document relevance used in prior work. The fallback behavior observed for "Not applicable" aspects also motivates our error analysis in Section 5.6, where we examine how this limitation affects downstream performance.

4 Method

In this section, we describe our proposed method. Let x be a question and $D = \{d_1, d_2, \ldots, d_n\}$ be the documents searched using x as a query in a dataset. Let $T(x, D) = \{t_1, t_2, \ldots, t_m\}$ be the concatenation of x and D tokenized, and let it be the input. The credibility s of each document is provided by an LLM, and based on the results of ABSA for the document, the credibility s is processed using the relevance r between x and d_i. Subsequently, the processed credibility s' is used to directly manipulate the attention weight of the LLM during the inference of the answer to the question x. Figure 2a shows the overall structure of the proposed method. Figure 2b shows a flowchart from extracting the aspect from a document to analyzing sentiment for the aspect. Figure 2c shows a flowchart up to calculating the relevance between a question and a document. Given a question and its document, sentiment and relevance are calculated. Then, the proposed method adjusts the credibility assigned by an LLM to the document based on sentiment and relevance. Section 4.1 explains the background of ABSA and how it is used in this method. Section 4.2 explains how to adjust the credibility based on the results of ABSA, taking into account the relevance to the question. Section 4.3 explains how to utilize the obtained credibility to reduce the influence of misinformation documents while leveraging information relevant to the question in an LLM.

4.1 Aspect Based Sentiment Analysis

ABSA can be divided into two tasks: aspect extraction and aspect sentiment analysis. In ABSA, an aspect refers to a specific subject or attribute mentioned within a document. In this study, given a question x and a set of related documents D, an aspect is extracted as the answer obtained when the LLM is given question x and document d_i, denoted as a_i. This is to prevent an excessive number of aspects while extracting aspects related to the question. This processing flow is shown in Figure 2b. The set of aspects for the document collection D is represented as $A = \{a_1, a_2, \ldots, a_n\}$.

For aspect sentiment analysis, using the corresponding pairs (d_i, a_i) for the document set D and the aspect set A obtained from it, the sentiment toward a_i is classified into one of the three categories, {optimistic, neutral, pessimistic}. However, in practice, many pairs are classified as neutral, providing limited information for credibility adjustment. To address this issue, for pairs initially classified as neutral, we re-evaluate the sentiment of a_i using a different document d_j $(i \neq j)$, as shown in Figure 2b. This design is to capture variations in how the same aspect is expressed across different documents. While a document

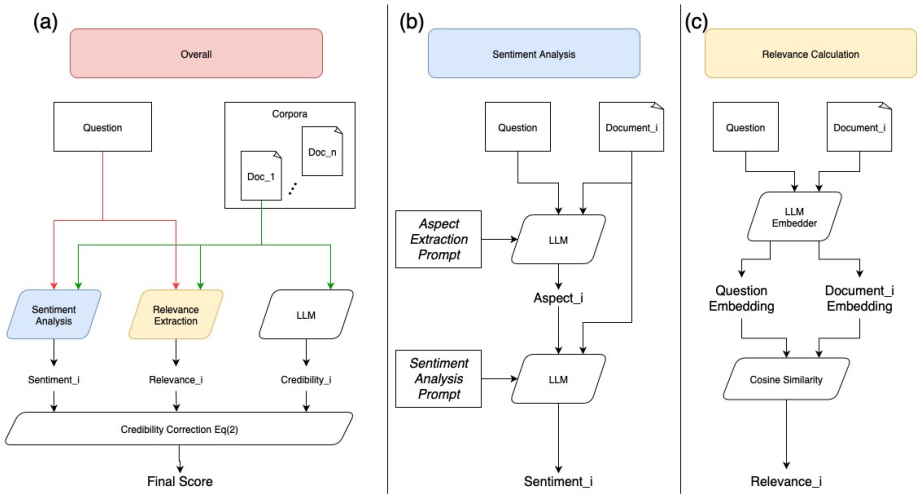


Fig. 2. (a) Overall structure of the proposed method. (b) Flowchart for document aspect extraction and sentiment analysis of aspects. (c) Flowchart for calculating the relevance between questions and documents. If Sentiment_i is neutral, Sentiment Analysis is performed again using a different Document_j for Aspect_i.

may describe an aspect in a neutral manner, other documents discussing the same aspect may exhibit more distinctive sentiment. This procedure depends on cross-document context and may introduce inconsistencies. It should therefore be regarded as a heuristic, and more principled alternatives are left for future work. Figures 3 and 4 show the prompts we used in the experiment for the ABSA tasks.

4.2 Credibility Adjustment Considering the Relevance between Questions and Documents with Sentiment

In this section, we explain how to adjust the credibility provided by the LLM using the relevance r between question x and document d_i. First, question x and document d_i are converted into embedded representations $emb(x)$ and $emb(d_i)$, respectively, using a language model. Then, as shown in Eq. 1, r is expressed by the cosine similarity between the two. This processing flow is shown in Figure 2c.

$$r = cosine\_similarity(emb(x), emb(d_i)) \tag{1}$$

Next, we adjust the credibility s as s' based on the sentiment analysis results for each (d_i, a_i) pair obtained in Section 4.1 using Eq. 2 as shown in the bottom of Figure 2a.

$$s' = \begin{cases} \dfrac{s}{1+r} & \text{(optimistic)} \\ s & \text{(neutral)} \\ s \times (1+r) & \text{(pessimistic)} \end{cases} \tag{2}$$

Aspect Extraction Prompt

Given the following information: {TEXT}

Answer the following question based on the given information or your internal knowledge with one or few words, without the source.

Do not answer with "Not mentioned", "Not applicable", or similar phrases.If the information is incomplete, infer the most reasonable and accurate answer based on your knowledge.

Question: {QUESTION}

Answer:

Fig. 3. Prompt template used for open-domain question answering as the aspect extraction. The template instructs the model to generate concise answers to a given question based on the provided context.

The branching condition in Eq. 2 is motivated by the empirical observation validated in Section 3: documents containing misinformation tend to describe that misinformation in an optimistic, confident tone (e.g., "something is absolutely good"), whereas accurate information within the same documents is typically expressed with cautious, hedging language (e.g., "something is probably considered to be good"). Thus, optimistic sentiment toward aspect a_i signals that document d_i likely contains misinformation and should receive reduced credibility, while pessimistic sentiment suggests that the document is reliable.

The role of the relevance score r in the adjustment reflects a second empirical finding, which we detail in Section 5.5: documents containing misinformation, particularly those generated by LLMs, tend to cluster in the high-relevance region of the score distribution. This means that r carries different implications depending on the sentiment. While cosine similarity does not capture logical entailment between a query and a document, our goal here is not to model entailment but to obtain a lightweight estimate of semantic relevance. Such similarity-based measures are commonly used in RAG pipelines for document ranking and filtering [12]. More expressive alternatives, such as entailment-based scoring, could potentially improve this component, and we leave this as future work. When sentiment is optimistic, a high r provides additional evidence that the document was crafted to closely match the query, warranting a stronger credibility penalty; dividing by $(1 + r)$ achieves this effect. When sentiment is pessimistic, a high r indicates that the document is both topically relevant and expressed with appropriate caution, so multiplying by $(1 + r)$ rewards this combination. For neutral cases, where sentiment analysis provides no discriminative signal, we retain the original LLM-assigned credibility s without modification.

We use $(1+r)$ rather than r alone to ensure numerical stability, since $r \in [0, 1]$ could otherwise reduce credibility to zero or leave it unchanged. We note that

Sentiment Analysis Prompt

You are an assistant for aspect-based sentiment analysis (ABSA). Your task is to classify the sentiment for the given aspect in the provided text as either "optimistic", "pessimistic", or "neutral".

- If the aspect is mentioned and discussed in a positive manner, return "optimistic".
- If the aspect is mentioned and discussed in a negative manner, return "pessimistic".
- If the aspect is not clearly mentioned, return "neutral".
- Do NOT infer sentiment just from unrelated positive or negative language in the text.

Respond with **only** one of the three labels: "optimistic", "pessimistic", or "neutral".

Text:{TEXT}

Aspect:{ASPECT}

Sentiment:

Fig. 4. Prompt template for the aspect sentiment analysis. The model is instructed to classify the sentiment toward a specified aspect as optimistic, pessimistic, or neutral.

alternative formulations, such as exponential scaling or learned weighting functions, are also possible. We chose the form in Eq. 2 because it is parameter-free and requires no additional training, consistent with our goal of a lightweight method that builds on CrAM [3] without introducing extra computational overhead. Despite its simplicity, this formulation yields consistent improvements across all tested models and datasets, as shown in Section 5.

4.3 Suppression of the Influence of Misinformation Using Credibility

In this section, we describe a method for suppressing the influence of documents containing misinformation using credibility. In this paper, following [3], we normalize the modified credibility $S' = [s'_1, s'_2, \ldots, s'_n]$ using min-max normalization to obtain $\overline{S'} = [\overline{s'_1}, \overline{s'_2}, \ldots, \overline{s'_n}]$. We then use $\overline{S'}$ to modify the attention mask as Eq. 3.

$$\mathrm{Attn}(w'_{t_i}) = \mathrm{Attn}(\overline{s'_i} \times w_{t_i}) \tag{3}$$

Here, w_{t_i} corresponds to t_i in the token sequence $T(x, D) = \{t_1, t_2, \ldots, t_m\}$. We use the processed credibility to suppress the impact of misinformation through this method.

5 Experiments

5.1 Experimental Setup

We evaluate our proposed method on Open-domain Question Answering (ODQA) using three LLMs: Llama3-8B [14], Llama2-13B [24], and qwen1.5-7B [2]. Relevance scores are computed with paraphrase-MiniLM-L6-v2 [22], and all ABSA steps use Llama3-8B. We use the Natural Questions (NQ) [13] and TriviaQA [10] datasets. Retrieved documents, misinformation documents, and their credibility scores are taken from [3]. The external corpus is constructed from the English Wikipedia dump (December 20, 2018), following the preprocessing procedure of [12]. Following [3], we evaluate under two settings: 4acc (4 accurate documents) and 4acc + 1misinfo (adding 1 misinformation document). Misinformation documents used in our experiments are generated by an LLM. This design choice allows us to control the degree of misinformation in a reproducible manner. Although synthetic misinformation may introduce certain biases, it provides a controlled setting to isolate the effect of misinformation, which is difficult to achieve with fully natural data. Moreover, question answering models remain vulnerable to misinformation attacks regardless of whether the misleading content is manually crafted or generated by an LLM [17]. Therefore, using LLM-generated documents does not fundamentally limit the generality of our findings, while enabling scalable and reproducible experimentation. All experiments were conducted on a machine equipped with an Intel Core i7-7700 CPU, an NVIDIA GeForce RTX 3090 GPU (24GB memory), and 32GB RAM, using CUDA 12.8.

The evaluation metrics used are Exact Match (EM) and token-level F1 score, both standard in ODQA [3,12,18].

We compare against three inference-time, training-free methods: (1) **Naive RAG**, standard RAG without misinformation countermeasures; (2) **Pseudo Cred**, which binarizes LLM-assigned credibility scores using an optimal ROC-AUC threshold (AUC = 0.780, threshold = 5), assigning credibility 0 or 10 in CrAM [3]; and (3) **LLM Cred**, which directly applies CrAM using the raw LLM-assigned credibility scores.

We focus on training-free baselines because fine-tuning-based approaches [18,6] occupy a fundamentally different point in the cost-accuracy trade-off, requiring credibility-annotated training data and retraining when the corpus or domain changes. Our method requires no additional training and can be applied to any LLM that supports attention weight modification, making it complementary to rather than competing with fine-tuning-based approaches.

5.2 Main Results

Table 1. Performance comparison of different methods on open-domain QA tasks. The best results in each column are shown in **bold**, and the second-best results are underlined. "acc" indicates an accurate document. "misinfo" indicates a document containing misinformation.

Model	In-context corpus	Method	NQ		TriviaQA	
			EM	F1	EM	F1
Llama3-8B	4acc	Naive RAG	33.30	**45.58**	**64.30**	**73.72**
	4acc + 1misinfo	Naive RAG	16.30	26.39	37.10	47.36
	4acc + 1misinfo	Pseudo Cred	27.50	38.61	58.20	67.38
	4acc + 1misinfo	LLM Cred	30.80	41.81	58.40	67.08
	4acc + 1misinfo	Ours	**34.30**	45.11	61.00	69.72
Llama2-13B	4acc	Naive RAG	**28.90**	**40.01**	**62.50**	**71.02**
	4acc + 1misinfo	Naive RAG	12.00	20.10	28.01	36.72
	4acc + 1misinfo	Pseudo Cred	24.20	35.11	52.30	59.59
	4acc + 1misinfo	LLM Cred	25.20	36.44	51.80	59.30
	4acc + 1misinfo	Ours	27.60	38.08	54.70	61.89
qwen1.5-7B	4acc	Naive RAG	**27.70**	**39.20**	**57.00**	**67.50**
	4acc + 1misinfo	Naive RAG	9.80	19.51	23.10	34.08
	4acc + 1misinfo	Pseudo Cred	22.50	34.26	50.40	61.67
	4acc + 1misinfo	LLM Cred	23.70	34.88	52.60	62.93
	4acc + 1misinfo	Ours	25.90	37.05	53.04	64.40

The experimental results are shown in Table 1. Under the 4acc + 1misinfo setting, our method consistently achieves the best performance across all models and datasets. On NQ with Llama3-8B, our method improves EM by 3.50 points

over LLM Cred (34.30 vs. 30.80), and on TriviaQA with qwen1.5-7B it improves EM by 0.44 points (53.04 vs. 52.60), representing our smallest gain. Notably, on NQ with Llama3-8B, our method under 4acc + 1misinfo (EM 34.30) nearly matches Naive RAG under 4acc with no misinformation present (EM 33.30), demonstrating that the multi-perspective credibility adjustment can almost fully recover the performance lost to misinformation contamination.

Table 2 reports p-values from Wilcoxon signed-rank tests comparing F1 scores between our method and LLM Cred under 4acc + 1misinfo. Our method significantly outperforms LLM Cred in four of six cases ($p < 0.01$). For Llama2-13B on NQ ($p = 0.06$) and qwen1.5-7B on TriviaQA ($p = 0.04$), the improvements are consistent in direction but do not all reach the $p < 0.01$ threshold, likely due to the relatively small effect sizes on these particular model-dataset combinations.

However, our method under 4acc + 1misinfo does not consistently surpass Naive RAG under 4acc, particularly on TriviaQA. This gap indicates that while our credibility adjustment substantially mitigates the impact of misinformation, it does not yet fully compensate for its presence. We analyze this further in Section 6.

We also observe that Pseudo Cred generally underperforms LLM Cred (with the exception of Llama2-13B on TriviaQA). This can be attributed to its binary credibility assignment: setting scores to either 0 or 10 based on a threshold can excessively suppress or amplify the influence of documents, discarding useful information in the process. This result reinforces a key insight of our work: effective misinformation handling requires nuanced credibility adjustment, not simply binary detection.

Table 2. P-values using Wilcoxon signed-rank test to F1 score comparing our method with the LLM Cred under the setting of 4acc + 1misinfo.

Model	NQ	TriviaQA
Llama3-8B	< 0.01	< 0.01
Llama2-13B	0.06	< 0.01
qwen1.5-7B	< 0.01	0.04

5.3 Ablation Study

We analyze the impact of the two components in the proposed method's credibility adjustment process on performance: (1) sentiment analysis on document aspects (ABSA) and (2) document-question relevance evaluation (Relevance). Evaluations were conducted on the NQ and TriviaQA datasets using two models: Llama3-8B and Llama2-13B. When using only the ABSA component, the credibility score is determined solely based on aspect-level sentiment. Specifically, if the sentiment is classified as optimistic, we assign a credibility score of

0; if it is pessimistic, the score is set to 10; and if it is neutral, the credibility score provided by the LLM is retained without modification.

As shown in Table 3, the full version of our proposed method (ABSA + Relevance) achieved the highest performance on both datasets. Compared to methods relying solely on individual components, consistent superiority is observed. These results suggest that combining both relevance and sentiment complements each other in evaluating documents containing misinformation. Specifically, relying solely on relevance is highly likely to assign high credibility to documents containing misinformation, while sentiment alone has weak ties to the query. Integrating both improves the accuracy of the final credibility estimation.

Table 3. Performance comparison of ablation variants for each model. "ABSA" indicates cases where ABSA results are only used for credibility adjustment, while "Relevance" indicates cases where relevance values are only used for credibility adjustment. "ABSA + Relevance" indicates the proposed method's full components. The best results in each column are shown in **bold**.

Model	Method	NQ		TriviaQA	
		EM	F1	EM	F1
Llama3-8B	ABSA	30.50	41.42	60.40	69.28
	Relevance	29.10	41.05	54.30	63.38
	ABSA + Relevance	**34.30**	**45.11**	**61.00**	**69.72**
Llama2-13B	ABSA	26.70	36.46	53.70	61.18
	Relevance	26.20	35.81	48.30	55.13
	ABSA + Relevance	**27.60**	**38.08**	**54.70**	**61.89**

5.4 Robustness against the Number of Documents with Misinformation

This section examines the impact of the number of documents containing misinformation on answer accuracy. Based on the NQ dataset, we compared the performance of our method with LLM Cred, which relies solely on LLM credibility, by adding 1, 2, or 3 documents containing misinformation to the query while keeping the four reliable documents unchanged. The results are shown in Figure 5. Compared to CrAM, our method showed no significant performance degradation even as the number of documents containing misinformation increased, achieving EM scores of 34.3 (1 misinformation document), 34.0 (2 misinformation documents), and 32.2 (3 misinformation documents). This demonstrates our method's robustness against misinformation. In contrast, using only LLM's credibility performance deteriorated sharply as misinformation increased, with EM scores decreasing from 30.8 (1 misinformation document) to 28.4 (2 misinformation documents) and further to 27.1 (3 misinformation documents). The performance drop of our method was 5.5%, while that of LLM Cred was

11.4%, approximately twice as large. Our method consistently maintained its performance even as the number of misinformation documents increased. This highlights that conventional methods are vulnerable to misinformation contamination because they directly reflect misinformation in their credibility.

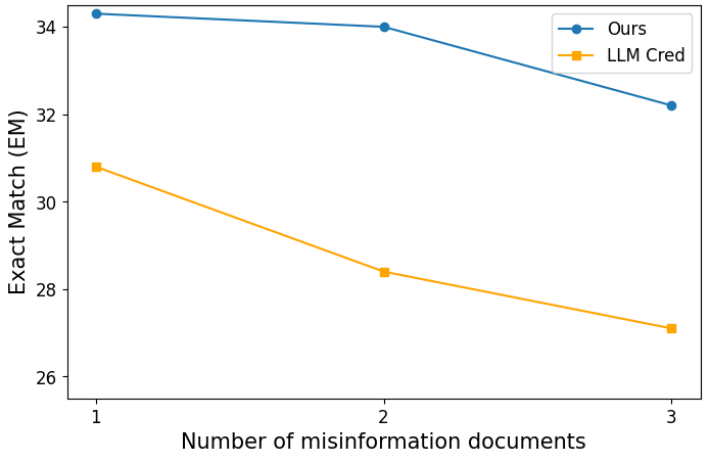


Fig. 5. Performance changes on the ODQA task as the number of documents containing misinformation increases.

5.5 Distribution of Relevance between Questions and Documents

This section investigates the relevance score between documents and questions. We used 10,000 retrieved documents without misinformation and 3,000 created documents with misinformation [3]. We used paraphrase-MiniLM-L6-v2 [22] for calculating the relevance scores as well as the main experiment.

Figure 6 shows the relevance distribution between documents containing misinformation and those without. The results confirmed that the number of documents containing misinformation was greater than that of those without in areas highly relevant to the questions. This is likely because documents containing misinformation were generated by the LLM based on the question, leading to higher relevance to the question. Furthermore, the tendency for documents containing misinformation to have higher relevance indicates that misinformation documents are more likely to be prioritized by the retriever. This aligns with the vulnerability of the retriever to misinformation [19].

5.6 Error Analysis in Aspect-Based Sentiment Analysis

Although our proposed method improves robustness against misinformation in RAG, we observed several recurring limitations related to ABSA.

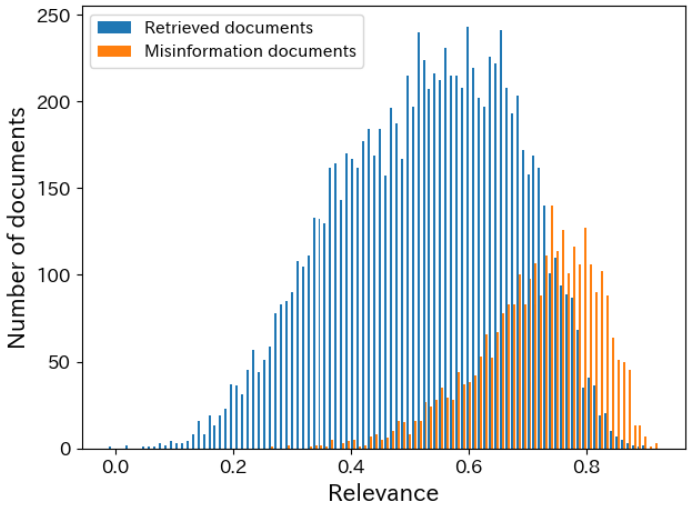


Fig. 6. This graph shows the distribution of relevance scores between documents containing misinformation and those without. The horizontal axis represents the relevance score between a query and a document, while the vertical axis shows the number of documents.

Table 4. Representative limitations observed in our method. Limitations include low-quality LLM-generated aspects and uncertainty of aspect presence across documents. The last row is attempting to have the LLM directly output aspects. Since many aspects were irrelevant to the question, aspect sentiment analysis was not conducted in this example.

Question	Document	Aspect	Sentiment	Error Type
who plays scrooge mcduck in the new ducktales	Ed Helms Edward Parker Helms (born January 24, 1974) is an American actor, comedian, and ...	Not applicable	neutral	If no answer is found for a question, the LLM will output "Not applicable"
who plays scrooge mcduck in the new ducktales	"CNN news has confirmed that Benedict Cumberbatch has been cast as Scrooge McDuck in the ...	Not applicable	optimistic	An LLM outputs document-level sentiment rather than aspect-level
who was the actor that played ben stone on law and order	as Captain Donald Cragen. On the prosecutor's side, Michael Moriarty was Dick Wolf's choice to play Executive Assistant District Attorney Benjamin Ben ...	Ben Stone, Michael Moriarty, James Naughton, Richard Brooks, Eriq La Salle, Roy Thinnes, District Attorney, Executive ...	-	LLM outputs more aspect, some are not relevant to question

Table 4 illustrates representative limitations. We categorize the observed limitations as follows. First, relying on LLM outputs as aspects can introduce low-quality aspects, such as "Not applicable" or "unknown", which may degrade sentiment analysis results. A potential mitigation is to use document titles (e.g., from Wikipedia) as provisional aspects.

Second, it is not guaranteed that an aspect appears across multiple documents. In some cases, the LLM outputs sentiment corresponding to the entire document rather than the specific aspect, due to the absence of aspects in the document. Addressing these issues remains an open challenge for future work.

Third, we show an example output when instructing the LLM to extract aspects from the document without treating the answer to the question as an aspect at the last raw of Table 4. Attempting to have the LLM directly output aspects resulted in the extraction of terms unrelated to the question. Therefore, in our method, by treating answers to the question as aspects, we prevent the extraction of aspects unrelated to the question, enabling the extraction of only those aspects relevant to the question.

We discuss broader limitations of our experimental setup in Section 6, and outline concrete strategies for addressing these ABSA-specific issues in Section 7.

6 Limitations

While our method demonstrates consistent improvements across multiple models and datasets, we acknowledge several limitations that contextualize our results and suggest directions for future work.

First, the documents containing misinformation used in our experiments were generated by an LLM following the protocol of [3]. This controlled setup ensures reproducibility and allows us to isolate the effect of misinformation on RAG performance, but it does not fully capture the diversity of real-world misinformation. In practice, misinformation may originate from edited encyclopedia articles, fabricated news reports, or social media posts, which may not exhibit the same optimistic tonal patterns that LLM-generated misinformation displays. In particular, adversarially crafted misinformation could deliberately mimic the cautious, hedging language that our method associates with reliable information. Evaluating our approach on naturally occurring misinformation corpora is an important direction for future work.

Second, all ABSA steps in our experiments were performed using a single model (Llama3-8B), regardless of which LLM was used for answer generation. While this design keeps the computational overhead low and already yields significant improvements, we have not yet investigated how sensitive the method is to the choice of ABSA model. A weaker or domain-mismatched model may produce less reliable sentiment classifications, potentially degrading the quality of credibility adjustment. Exploring this sensitivity and the impact of different sentiment analysis methods, including non-LLM-based approaches is left to future work.

Third, as shown in Table 1, our method under the 4acc + 1misinfo setting does not fully recover to the performance of Naive RAG under 4acc (i.e., the setting with no misinformation). This suggests that while our credibility adjustment substantially mitigates the impact of misinformation, it does not yet eliminate it entirely. A promising direction would be to develop an adaptive mechanism that detects whether misinformation is likely present in the retrieved documents and applies credibility adjustment only when needed, thereby avoiding unnecessary modification of attention weights in clean retrieval scenarios.

7 Conclusion

In this study, we proposed a multi-perspective credibility adjustment method for RAG that incorporates Aspect-Based Sentiment Analysis (ABSA) to suppress the influence of misinformation while preserving access to valuable information within documents. Unlike conventional approaches that estimate credibility from a single perspective, our method integrates three complementary signals—LLM-assigned credibility, query-document relevance, and aspect-level sentiment—to achieve a more nuanced evaluation of document reliability. Experiments on open-domain question-answering tasks using multiple LLMs and datasets confirmed that our method consistently outperforms existing credibility adjustment approaches and maintains robust performance even as the number of documents containing misinformation increases.

Beyond the directions discussed in Section 5.6, we identify two specific avenues for improving the ABSA component of our method. First, when the LLM cannot extract a meaningful aspect from a document (i.e., the aspect is "Not applicable"), the current method retains this uninformative aspect as-is. A natural improvement would be to substitute the document's title or topic heading as a proxy aspect, providing a more meaningful target for sentiment analysis. Second, aspects extracted from one document are not guaranteed to appear in other documents, which can cause the LLM to fall back on document-level sentiment rather than aspect-level analysis. Extracting multiple candidate aspects per document and selecting only those that are shared across the document set could mitigate this issue and improve the consistency of sentiment-based credibility adjustment.

References

1. Alkhnbashi, O.S., Mohammad, R., Hammoudeh, M.: Aspect-based sentiment analysis of patient feedback using large language models. Big Data and Cognitive Computing **8**(12) (2024). https://doi.org/10.3390/bdcc8120167, https://www.mdpi.com/2504-2289/8/12/167
2. Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S.,

Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., Zhu, T.: Qwen technical report (2023), https://arxiv.org/abs/2309.16609
3. Deng, B., Wang, W., Zhu, F., Wang, Q., Feng, F.: Cram: Credibility-aware attention modification in llms for combating misinformation in rag. Proceedings of the AAAI Conference on Artificial Intelligence **39**(22), 23760–23768 (Apr 2025). https://doi.org/10.1609/aaai.v39i22.34547, https://ojs.aaai.org/index.php/AAAI/article/view/34547
4. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-augmented generation for large language models: A survey (2024), https://arxiv.org/abs/2312.10997
5. Guo, B., Ding, Y., Yao, L., Liang, Y., Yu, Z.: The future of misinformation detection: New perspectives and trends (2019), https://arxiv.org/abs/1909.03654
6. Hong, G., Kim, J., Kang, J., Myaeng, S.H., Whang, J.J.: Why so gullible? enhancing the robustness of retrieval-augmented models against counterfactual noise. In: Duh, K., Gomez, H., Bethard, S. (eds.) Findings of the Association for Computational Linguistics: NAACL 2024. pp. 2474–2495. Association for Computational Linguistics, Mexico City, Mexico (Jun 2024). https://doi.org/10.18653/v1/2024.findings-naacl.159, https://aclanthology.org/2024.findings-naacl.159/
7. Izacard, G., Grave, E.: Leveraging passage retrieval with generative models for open domain question answering. In: Merlo, P., Tiedemann, J., Tsarfaty, R. (eds.) Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. pp. 874–880. Association for Computational Linguistics, Online (Apr 2021). https://doi.org/10.18653/v1/2021.eacl-main.74, https://aclanthology.org/2021.eacl-main.74/
8. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. ACM Comput. Surv. **55**(12) (Mar 2023). https://doi.org/10.1145/3571730
9. Jiayang, C., Chan, C., Zhuang, Q., Qiu, L., Zhang, T., Liu, T., Song, Y., Zhang, Y., Liu, P., Zhang, Z.: ECON: On the detection and resolution of evidence conflicts. In: Al-Onaizan, Y., Bansal, M., Chen, Y.N. (eds.) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 7816–7844. Association for Computational Linguistics, Miami, Florida, USA (Nov 2024). https://doi.org/10.18653/v1/2024.emnlp-main.447, https://aclanthology.org/2024.emnlp-main.447/
10. Joshi, M., Choi, E., Weld, D., Zettlemoyer, L.: TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In: Barzilay, R., Kan, M.Y. (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1601–1611. Association for Computational Linguistics, Vancouver, Canada (Jul 2017). https://doi.org/10.18653/v1/P17-1147, https://aclanthology.org/P17-1147/
11. Kaliyar, R.K., Goswami, A., Narang, P.: Fakebert: Fake news detection in social media with a bert-based deep learning approach. Multimedia Tools Appl. **80**(8), 11765–11788 (Mar 2021). https://doi.org/10.1007/s11042-020-10183-2
12. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.t.: Dense passage retrieval for open-domain question answering. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 6769–6781. Association for Computational Linguistics, Online (Nov 2020). https://doi.org/10.18653/v1/2020.emnlp-main.550, https://aclanthology.org/2020.emnlp-main.550/

13. Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.W., Dai, A.M., Uszkoreit, J., Le, Q., Petrov, S.: Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics **7**, 452–466 (2019). https://doi.org/10.1162/tacl_a_00276, https://aclanthology.org/Q19-1026/
14. LlamaTeam, AI@Meta: The llama 3 herd of models (2024), https://arxiv.org/abs/2407.21783
15. Manakul, P., Liusie, A., Gales, M.: SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 9004–9017. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.emnlp-main.557, https://aclanthology.org/2023.emnlp-main.557/
16. Min, S., Krishna, K., Lyu, X., Lewis, M., Yih, W.t., Koh, P., Iyyer, M., Zettlemoyer, L., Hajishirzi, H.: FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 12076–12100. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.emnlp-main.741, https://aclanthology.org/2023.emnlp-main.741/
17. Pan, L., Chen, W., Kan, M.Y., Wang, W.Y.: Attacking open-domain question answering by injecting misinformation. In: Park, J.C., Arase, Y., Hu, B., Lu, W., Wijaya, D., Purwarianti, A., Krisnadhi, A.A. (eds.) Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 525–539. Association for Computational Linguistics, Nusa Dua, Bali (Nov 2023). https://doi.org/10.18653/v1/2023.ijcnlp-main.35, https://aclanthology.org/2023.ijcnlp-main.35/
18. Pan, R., Cao, B., Lin, H., Han, X., Zheng, J., Wang, S., Cai, X., Sun, L.: Not all contexts are equal: Teaching LLMs credibility-aware generation. In: Al-Onaizan, Y., Bansal, M., Chen, Y.N. (eds.) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 19844–19863. Association for Computational Linguistics, Miami, Florida, USA (Nov 2024). https://doi.org/10.18653/v1/2024.emnlp-main.1109, https://aclanthology.org/2024.emnlp-main.1109/
19. Pan, Y., Pan, L., Chen, W., Nakov, P., Kan, M.Y., Wang, W.: On the risk of misinformation pollution with large language models. In: Bouamor, H., Pino, J., Bali, K. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2023. pp. 1389–1403. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.findings-emnlp.97, https://aclanthology.org/2023.findings-emnlp.97/
20. Pelrine, K., Imouza, A., Thibault, C., Reksoprodjo, M., Gupta, C., Christoph, J., Godbout, J.F., Rabbany, R.: Towards reliable misinformation mitigation: Generalization, uncertainty, and GPT-4. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 6399–6429. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.emnlp-main.395, https://aclanthology.org/2023.emnlp-main.395/

21. Quevedo, E., Yero, J., Koerner, R., Rivas, P., Cerny, T.: Detecting hallucinations in large language model generation: A token probability approach (2024), https://arxiv.org/abs/2405.19648
22. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1410, https://aclanthology.org/D19-1410/
23. Simmering, P.F., Huoviala, P.: Large language models for aspect-based sentiment analysis (2023), https://arxiv.org/abs/2310.18025
24. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C.C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P.S., Lachaux, M.A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E.M., Subramanian, R., Tan, X.E., Tang, B., Taylor, R., Williams, A., Kuan, J.X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., Scialom, T.: Llama 2: Open foundation and fine-tuned chat models (2023), https://arxiv.org/abs/2307.09288
25. Yoran, O., Wolfson, T., Ram, O., Berant, J.: Making retrieval-augmented language models robust to irrelevant context (2024), https://arxiv.org/abs/2310.01558
26. Yoshida, Z., Aritsugi, M.: Rumor detection in twitter with social graph structures. In: Yang, X.S., Sherratt, S., Dey, N., Joshi, A. (eds.) Third International Congress on Information and Communication Technology. pp. 589–598. Springer Singapore, Singapore (2019). https://doi.org/10.1007/978-981-13-1165-9_54
27. Zhang, H., Zhang, R., Guo, J., de Rijke, M., Fan, Y., Cheng, X.: Are large language models good at utility judgments? In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 1941–1951. SIGIR '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3626772.3657784
28. Zhang, W., Deng, Y., Liu, B., Pan, S., Bing, L.: Sentiment analysis in the era of large language models: A reality check. In: Duh, K., Gomez, H., Bethard, S. (eds.) Findings of the Association for Computational Linguistics: NAACL 2024. pp. 3881–3906. Association for Computational Linguistics, Mexico City, Mexico (Jun 2024). https://doi.org/10.18653/v1/2024.findings-naacl.246, https://aclanthology.org/2024.findings-naacl.246/

An Ontology-Driven Approach for Querying Relational Databases

Ameni Souid[1][0000−0001−5562−8784], Mohamed Amin Gaied[1], Rose-Line Baillargeon[1], and Christina Khnaisser[1][0000−0002−1186−0300]

Groupe de Recherche Interdisciplinaire en Informatique de la Santé (GRIIS), Faculté de médecine et des sciences de la santé, Faculté des sciences, Université de Sherbrooke, Quebec, Canada
Ameni.Souid@usherbrooke.ca
Christina.Khnaisser@usherbrooke.ca

Abstract. Integrating and querying heterogeneous data stored in relational databases while maintaining semantic consistency is required to efficiently support analytical applications. It is crucial to provide end users with a unified view of heterogeneous data sources. This view can be expressed using an ontology to represent the knowledge of a specific domain. Thus, combining an ontology with a relational database would improve data storage, query formulation and systematic data retrieval. Yet, querying heterogeneous data remains a challenge when using ontologies. To address this issue, we have introduced an approach designed to simplify the formulation of queries and to automate query generation using three tasks: query design, query implementation, and data extraction. Our approach enables end users to express rich and interoperable queries using ontological entities, without needing technical expertise in the underlying data infrastructure while ensuring both semantic and structural consistency. We demonstrate the approach through an experimental evaluation using a real-world dataset and queries in the healthcare domain. This evaluation highlights the efficiency and practicality of the approach for querying complex data.

Keywords: Ontology querying · Graph-based querying · Relational database querying · Ontology-based modelling

1 Introduction

Across many domains, valuable information is distributed among databases with diverse semantic and syntax. Semantics refers to the meaning of data elements and their relationships, while syntax refers to the structure and representation of data, including formats and data types. For example, age can semantically refer to a person's life stage or the age at the time of admission and a date of birth could be stored in one column using ISO8601 format or three separate columns for the day, month and year. In complex heterogeneous environments, adopting a shareable and interoperable knowledge model has proven effective in deciphering data structures and integrating relevant elements in a semantically

coherent manner [8, 6]. Ontologies provide a formal and objective representation of knowledge with classes that denote entities and relationships that exist in reality and whose definitions adhere to sound criteria that ensure unambiguous interpretation across different domains [5, 16]. Relational database management systems (RDBMS) remain widely used across various domains, offering built-in access control, transaction management, data integrity and query efficiency [19]. Thus, combining an ontology with a relational database would improve data storage, query formulation, and systematic data retrieval [14]. The ontology will serve as a reference ensuring that all stakeholders use consistent terminology and definitions when sharing or accessing relevant data. The relational database will serve as a reliable data store.

The work [8] developed a method to derive a relational schema from an ontology, enabling explicit axiomatization of the data structure for seamless access to both data and its semantic. However, querying heterogeneous data remains a challenge when using ontologies [10]. Many users seek accurate ways to access multiple data sources without needing to manually align the semantics and the syntax of each database. Thus, we suggest a querying process for accessing data in a relational database using ontological entities without the need to master a query language or be aware of all entities of the underlying domain ontology. Our approach has the representational benefits of a knowledge graph while being able to leverage the robust functionalities of the RDBMS. The contributions in this paper can be summarized as follows:

- we define the ontological relational data model (OntoRel), a data model derived from an ontology;
- we introduce a three-task querying process, including query design, query implementation and data extraction;
- we define three implementation query layers to improve data exploration and query efficiency, depending on the data access purpose;
- we evaluate the approach and the implementation efficiency using a real-world use case in healthcare.

The rest of the paper is structured as follows. Section 2 describes the approach. Section 3 discusses the experimental evaluation and outlines future improvements. Section 4 presents existing solutions. Section 5 concludes the paper.

2 Method

Our approach leverages ontological constructs guiding users in the semantic formulation of a request that will be automatically translated into an executable query. Figure 1 illustrates the querying process. A user initiates a request using entities defined in the ontology. The query design task generates a query specification aligned with the user request using the ontological relational schema (OntoRel). The query implementation task converts the query specification into executable code tailored to the selected data sources. Finally, the data extractor runs the queries, combines their results, and delivers them to the user.

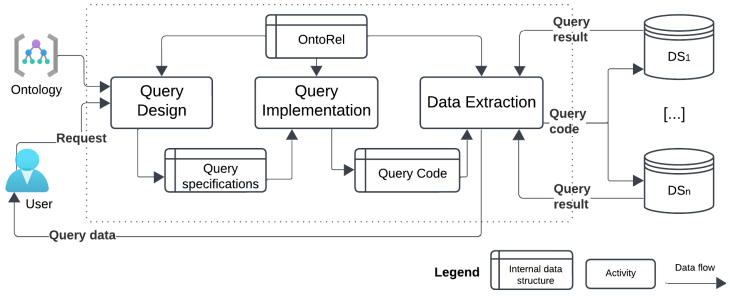


Fig. 1: Querying process.

2.1 Ontology

An ontology is composed of classes, individuals, properties (object properties and data properties), data types, annotations and axioms that define formal relationships between constructs [11]. The ontology is represented by a property graph. A property graph is a directed graph allowing multiple edges between a given pair of nodes, where both nodes and edges can carry properties.

Definition 1 (Ontology graph). *An ontology is represented by a property graph* $G_o = \langle N_o, E_o \rangle$ *where:*

- $N_o \subseteq C \cup D$ *is a finite set of nodes representing classes (C) and data types (D). Nodes are labelled using the Internationalized Resource Identifier (IRI).*
- $E_o \subseteq N_o \times N_o$ *is a finite set of edges (linking two nodes) representing axioms. We distinguish three categories of edges:*
  - *an inheritance edge, labelled with 'ISA', represents an inheritance axiom;*
  - *an object property edge, labelled with the object property IRI* (op_i)*, represents an axiom of the form* $C_1\, op_i\, C_2$*;*
  - *a data property edge, labelled with the data property IRI* (dp_i)*, represents an axiom of the form* $C\, dp_i\, D$*.*
- *The node and edge properties can be defined as follows:*
  - *label: a human-readable name in a specific language;*
  - *definition: a textual value in a specific language describing the entity;*
  - *cardinality: an interval of non-negative integers, defined only on edges, restricts the number of instances that can be linked using the property.*

Figure 2 illustrates a portion of an ontology describing a human patient visiting a healthcare organization, including the birth date, death date, and the beginning and ending of a visit. The circles, rectangles and lines symbolize classes, data types and axioms, respectively. For clarity, the example uses both English labels and the IRI.

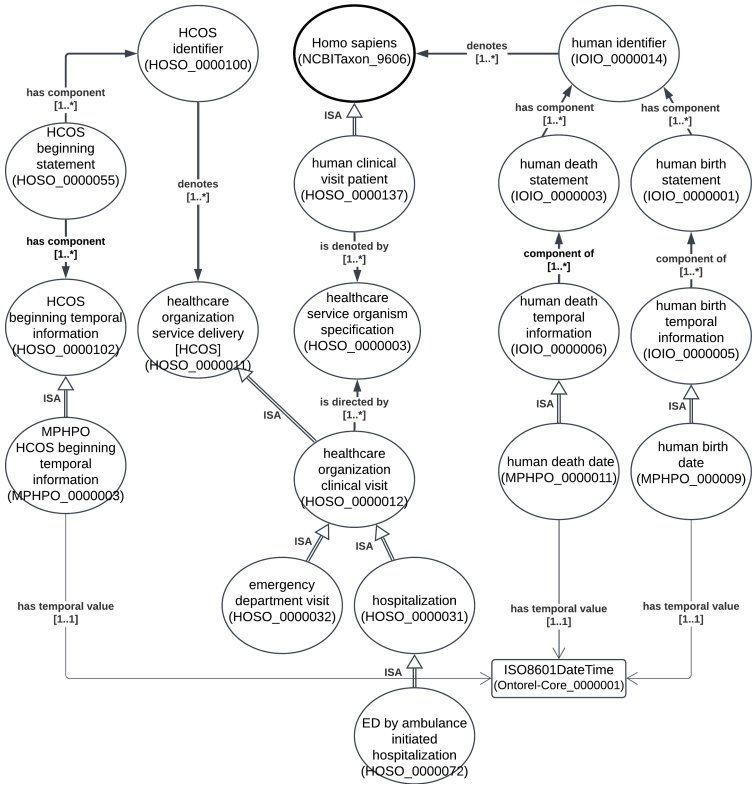


Fig. 2: An ontology graph.

2.2 Ontological relational model

Structuring a relational database based on the ontology enables queries to accurately capture data semantics. An OntoRel is a relational schema automatically derived from an ontology using conversion using *OntoRel-Application*[1]. A relational schema is constructed with a set of relations with attributes (pairs of a unique name and a datatype), tuples and constraints (mainly candidate keys, referential keys) [2]. The generated relational schema is normalized into 6th normal form, allowing a consistent handle missing values across data sources [15, 3]. Briefly, a class is converted into a relation with a candidate key. An axiom is converted into a join relation with referential keys referring to the linked entities. A cardinality restriction is converted into a constraint to check the number of individuals participating in the axiom (using triggers). Finally, the labels and definitions are used to document the database and to provide multiple access interfaces in different languages using views. The OntoRel is implemented in RDBMS to efficiently store and retrieve data with ontological annotations from

[1] The tool is available at GitHub : https://github.com/OpenLHS/ontorela-application

various applications. Definition 2 describes the correspondence rules. For formal details about the conversion rules and the process, see [8].

Definition 2 (Ontology-Relational correspondence). *Let λ be a mapping function between elements in G_o and relational constructs, including relation, attribute, datatype, candidate key [CKey] and referential key [RKey]. A relational database is specified by a schema, which consists of a set of relations $R_1, ..., R_n$. Each relation is defined over a set of attributes. The following conversion rules apply:*

- ***Class mapping*** *: $\lambda_C : C_1 \mapsto R_{c1}(k_{c1})\, CKey(k_{c1})$ a class node corresponds to a relation with an attribute forming the candidate key. The relation name is the class IRI.*
- ***Inheritance axiom mapping*** *: $\lambda_{isa} : C_1\, ISA\, C_2 \mapsto RKey\, R_{c1}(k_{c1}) \rightarrow R_{c2}(k_{c2})$ an inheritance edge corresponds to a referential key from the candidate key of the subclass relation to the candidate key of the superclass relation.*
- ***Object property axiom mapping*** *: $\lambda_{op} : C_1\, op_i\, C_2 \mapsto R_{c1\, op_i\, c2}(k_{c1}, k_{c2})$ $CKey(k_{c1}, k_{c2})$ $RKey\,(k_{c1}) \rightarrow R_{c1}(k_{c1}), RKey\,(k_{c2}) \rightarrow R_{c2}(k_{c2})$ an object property edge corresponds to a (join) relation with two attributes forming the candidate key and two referential keys referring to the candidate key of each related class relation. The relation name is the concatenation of the IRI of the subject class, the object property and the object class respectively.*
- ***Data property axiom mapping*** *: $\lambda_{dp} : C_1\, dp_i\, D \mapsto R_{c1\, dp_i\, d}(k_{c1}, dp_i : D)\, RKey\,(k_{c1}) \rightarrow R_{c1}(k_{c1})$ a data property edge corresponds to a relation with an attribute for the key, an attribute for the data property with the specified datatype. If the cardinality is [0..1] the candidate key is $CKey(k_{c1})$, if [n..m] with $n >= 0$ and $m > 1$ the candidate key is $CKey(k_{c1}, dp_i)$. Also, a referential key referring to the candidate key of the related class relation is defined. The relation name is the concatenation of the IRI of the class, the data property and the datatype.*

Each data source participating in an analytical study must implement the OntoRel enabling explicit axiomatization of the data structure for seamless access to both data and its semantic.

2.3 Ontology-based querying process

The main objective of the process is to enable querying the ontological relational schema by using terms from the ontology. The ontology plays the role of an *interface specification* to access data stored in a relational database. The process is designed to guide the formulation of queries and to automate query generation using three tasks: query design, query implementation, and data extraction. An ontological relational query consists of the following parts: the user request, the query specification, the query code and the query result.

Query design. A user formulates a *request* by selecting entities of interest from the ontology (classes and properties). Then, a *query specification* is built and translated into executable code on the target data sources (*query code*).

Definition 3 (User request). *A user request $U\langle C, OP, EXP\rangle$ is a statement formed by set of classes of interests ($C \subseteq N_o$), a set of object properties of interests ($OP \subseteq E_o$). The request can be refined with a set of expressions (EXP):*

- *A **path restriction** specifies the types of edges allowed in a path. We distinguish two categories of path restrictions:*
  - ***inheritance restriction** specifies the depth of the inheritance edges in the hierarchy;*
  - ***object property restriction** specifies which object properties are permitted along the path*
- *A **data restriction** defines boolean conditions over values to be returned in the final result.*
- *A **statistical expression** defines aggregation functions over datatype attributes, and may include the grouping of some classes.*

To guide the user in defining a request, we defined intuitive request patterns for the most common needs.

Definition 4 (Request patten). *A pattern focuses on selecting ontological entities of interest and applying some data restriction or computing aggregation functions. The following patterns are defined:*

- *Select a list of classes*
- *Select a list of classes with an inheritance restriction*
- *Select a list of classes with a data restriction*
- *Select a list of classes with a property restriction*
- *Compute a function over a datatype related to a class*
- *Compute an aggregation function over a datatype related to a class with (or without) a grouping*
- *Compute a histogram over a datatype related to a class*
- *Compute the union, intersection or difference between two paths*

The patterns can be combined to form complex ones. The request patterns are presented as questions & answers enriched with suggestions guiding the user in defining and refining the request.

Example 1. The user needs to get the temporal value (OntoRelCore_0000004) of the human birth date (MPHPO_0000009) for patients in the emergency department (HOSO_0000032) whose birth date is earlier than 2025-02-12.

The user selects the classes with a data restriction pattern with the following parameters.

- Select classes : HOSO_0000032, MPHPO_0000009
- Select data properties : OntorelCore_0000004

– Add data restriction : MPHPO_0000009 < '2025-05-15'

The user request is translated into a query specification that specifies the semantic of the query structure. A query specification is a subgraph extracted from the ontology that corresponds to the user request. If multiple paths satisfy the request, the user can choose one or more to execute, depending on the analytical objective.

Definition 5 (Query specification). *A query specification is an induced subgraph of the ontology graph that contains nodes and edges satisfying the user's request. Let* $G_q = \langle N_q, E_q \rangle$ *be a subgraph of* G_o *where:*

– $N_q \subseteq N_o$ *the finite set of nodes includes the classes of interests, the connected datatype nodes and all required classes to form a valid path.*
– $E_q \subseteq E_o$ *the finite set of edges includes the edges of the requested object properties and all data properties that links* N_q*.*

Example 2. Considering the example 1. The query specification is:

```
(HOSO_0000032)-[ISA]->(HOSO_0000012)-[OpenLHS-Core_0000004]->
(HOSO_0000003)<-[IAO_0000235]-(HOSO_00000137)-[ISA]->
(NCBITaxon_9606)<-[IAO_0000219]-(IOIO_0000014)<-[RO_0002180]-
(IOIO_0000001)<-[OpenLHS-Core_0000070]-(IOIO_0000005)<-[ISA]-
(MPHPO_0000009)-[OntoRel-Core_0000004]->(OntoRel-Core_0000001)
```

To construct the query specification and retrieve all valid paths, a traversal algorithm based on the depth-first search with backtracking is applied to the ontology graph with the following rules (see Algorithm 1 and 2) :

– start the process at the specified start node;
– if the start node is identical to the end node, the subgraph will contain that node and its connected datatype nodes;
– visit a neighbour node of incoming and outgoing edges if it is connected by a property of interest (*neighborOf*). If the set of properties of interest (E_r is empty) all the neighbours are selected.
– stop the process when the end node is reached and all neighbour nodes have been explored;
– include all data property edges and datatype nodes in the path (*addConnectedType*).
– a valid path must contain all classes of interest;

The result of the algorithm is a subgraph extracted from the ontology graph (*SubGraph*) that contains all the nodes and edges in the path.

Algorithm 1: BuildQueryGraph

```
   inputs : G_o,N_r, E_r
   output: G_q, P
1  s ← N_r.first() ;    /*start node */
2  e ← N_r.last() ;       /*end node */
3  A = ∅ ;          /*selected nodes */
4  F = ∅ ;    /*final set of nodes */
5  P = ∅ ;   /*list of valid paths */
6  if s = e then
7  | A.add(s);
8  else
9  | P.addAll(FindPaths(G_o, s, e, E_r, ∅, ∅));
   |
10 foreach a ∈ A do
11 | F.add(AddConnectedType(G_o, a));
   |
12 G_q ← SubGraph(G_o, A)
```

Algorithm 2: FindPaths

```
   inputs : G_o, s, e, E_r, cp, V
   output: P
1  if CP = ∅ then
2  | CP.add(s) ;          /*current path */
   |
3  cn ← cp.last() ;       /*current node */
4  V.add(cn) ;           /*visited nodes */
5  P = ∅;
6  if cn = e then
7  | P.addAll(CP);
8  else
9  | N ← neighborOf(G_o, E_r, cn) − V;
10 | foreach n ∈ N do
11 | | CP.add(n);
12 | | P.addAll(FindPaths(G_o, n, e, E_r, CP, V));
   | |
13 | | CP.remove(CP.last());
   | |
14 V.remove(cn);
15 return P;
```

Query implementation. The query specification is converted into an executable SQL code for the target RDBMS. Each path in the query specification is converted into a relational expression.

Definition 6 (Query code). *The query code is an executable relational expression written in SQL. The elements of the query graph are translated as follows:*

- *Each edge is translated into a join operation in a selection expression (FROM). The operands of the join are the relations that represent the endpoints. The join condition is an equality expression on the primary keys.*
- *A data restriction is used in the restriction expression (WHERE).*
- *A statistical expression is translated into an aggregation function in the projection expression (SELECT). If a grouping is specified, the primary keys of the chosen classes are used in a grouping expression (GROUP BY).*
- *The attributes of all relations in the selection expression are used in the projection expression.*

In a statistical model, variables are needed to find answers to a research question being studied. A variable is a measurable attribute that can take different values across observations or over time. Variables represent the characteristics being studied and are used to describe, explain, or predict phenomena within the model. They can be classified into different types, such as independent variables (inputs or predictors) and dependent variables (outputs or responses). Using our approach, a data analyst will use ontology to determine which classes are needed to compute the variables (such as the age and number of admissions in some hospital departments).

Computing a variable one or multiple query may be needed. The queries may share the same ontological entities, so identical paths will be calculated multiple times. Thus, to enhance efficiency and minimize the computational cost of common paths, we suggest an implementation of the OntoRel using multiple query

layers. The creation of layers was driven by three key objectives: (1) managing query efficiency, (2) facilitating data interpretation by structuring information at different levels of abstraction, and (3) improving access control by assigning privileges based on user roles.

Definition 7 (Query layer). *A query layer is a set of relations grouped together for a specific purpose. We distinguish three layers:*

- *The Entity layer contains the set of relations derived from the ontology. A relation can correspond to a class or an axiom.*
- *The Path layer contains the set of relations representing common paths identified from multiple user requests. A common path is a join expression that appears more than once in the queries.*
- *The Variable layer contains the set of relations needed to compute a variable to answer an analytical question. A variable can be computed by using relations from all 3 layers.*

The *Entity layer* provides direct access to raw data through relations mapped to ontological entities. It is primarily used by database designers for ontology-based data modelling and mapping.

The *Path layer* provides views for frequently used ontological paths required to compute multiple variables. It is mainly used by database administrators to enhance query performance and reduce computational redundancy through view materialization.

The *Variable layer*, primarily used by data analysts, supports data exploration and the computation of statistical functions, such as preparing predictors for machine learning models.

Data extractor. In the final step of the process, the generated SQL is executed on the target RDBMS. The query code is dispatched to the connected data sources. The query results are combined then delivered to the user in the chosen format, enabling direct integration with external tools for further analysis. The final result includes the user request, the query specifications, the query code and the query result.

3 Evaluation

The approach was implemented and evaluated using a real-world use case from the healthcare domain[2].

[2] The software and the evaluation report are available on Github : https://github.com/OpenLHS/OntoRelQuery

3.1 Use case

The main goal of the use case is to build a clinical decision system to improve palliative and end-of-life care outcomes for patients at a high risk of 1-year mortality [20]. The study evaluates the clinical utility of multiple prediction models using routinely collected data among hospitalized patients. As part of this project, 13 variables were collected. An ontology, 'Mortality Prediction for Hospitalized Patients Ontology'[3], was used to define generate the OntoRel data model. The MPHPO ontology contains 41 classes, 17 object properties linking these classes, and 3 data properties.

Dataset. For the experiment, we used the real-world dataset MIMIC-IV (Medical Information Mart for Intensive Care)[7]. MIMIC-IV is a large anonymized database containing information on patients admitted to intensive care between 2008 and 2019 (a total of 431 231 admissions).

Queries. For the experiment, we programmed 13 queries (one query per variable). [4]. Table 1 presents the characteristics of the variables, including the number of requests needed to build the query, the number of ontological classes in the resulting query graph, and the numbers of joins for each query implementation : queries on the entity layer, queries on the path layer and queries on MIMIC.

Table 1: Variables characteristics

Variable	Requests	Classes	Entity Joins	Path Joins	MIMIC Joins
V1 — 1 year mortality	3	6	23	2	1
V2 — Age	2	9	23	2	1
V3 — Sex	1	5	14	2	1
V4 — Emergency visits	4	8	30	2	4
V5 — Ambulance admission	3	8	16	2	4
V6 — Weeks recently hospitalized	3	9	19	2	2
V7 — Admission type	1	4	5	0	0
V8 — Admission service	1	11	33	1	1
V9 — Flu season	1	7	14	0	1
V10 — ICU Admission	1	2	2	1	2
V11 — Urgent 30 day readmission	4	9	24	3	3
V12 — Admission type	1	2	2	1	1
V13 — Visible comorbidities	3	9	24	2	3

Implementation strategies. In this experiment, we captured the runtime of 13 queries for four different implementations (4 sets of queries). In the first implementation, queries were written directly on the MIMIC tables (baseline).

[3] The ontology is available on Github : https://github.com/OpenLHS/MPHPO

[4] The full report of the query design is available on Github (evaluation-report.pdf): https://github.com/OpenLHS/OntoRelQuery/tree/main/doc

In the three others, queries were written through the OntoRel query layers: using the tables in the entity layer (*Entity*), using views in the path layer (*Path-View*) and using the materialized views in the path layer (*Materialized-Path*). This setup enables the assessment of the trade-offs between direct querying and implementation layers, including the impact of materialization on query performance.

Infrastructure. The experiment was conducted on an Intel(R) Xeon(R) Gold 5120 CPU @ 2.2GHz with 32GB main memory running Ubuntu 22.04.5 LTS. As a database system, we used PostgreSQL 17.4, where all configuration parameters are kept to the default values.

3.2 Results

The first interesting result to consider is that one or more user requests might be necessary to calculate a variable. The number of joins is higher in the entity layer due to the OntoRel schema being fully normalized to manage incomplete and dispersed data (as explained in section 2.2). The primary benefit is the ability to use generated queries for multiple layers, leading to greater reusability. First, common paths can be reused across multiple variables within the same analytical context, thanks to the implementation of the path layer. Also, views can be created to encapsulate common paths (a set of common join expressions), which significantly reduces the number of joins in the final variable query. The path layer can be automatically generated by the tool. Second, since paths are derived from the ontology, the same queries can be reused directly in other projects that use that same ontology.

The results, presented in Figure 3, reveal that transitioning from the MIMIC data model to OntoRel via *Materialized-Path* layer leads to an average improvement of 44.95% of the execution time. Overall, *Materialized-Path* demonstrates significant and consistent performance advantages over *MIMIC*, *Entity*, and *Path-View*. The remarkable improvement observed for Q10 (83.76%) is explained by the mapping process, which selects only the necessary data limiting the number of restrictions and traversal depth within the path. Most queries exhibit runtime reductions of around 70% (Q01, Q04, Q05, Q10, and Q13), highlighting the effectiveness of path materialization, particularly for complex queries involving multiple paths. Queries such as Q06, Q07 and Q09 exhibit improvements ranging from 30% to 50% due to the computational complexity of the involved variables and the optimization achieved by using materialized common paths. A closer examination reveals that simple queries, such as Q02, benefit only marginally (approximately 5.5%) due to their structural simplicity, where the materialization overhead partially offsets potential gains. However, two queries Q03 and Q11 exhibit a slight performance degradation of approximately 6%. This behaviour is primarily explained by the fact that, in the direct MIMIC implementation, attributes such as gender are immediately accessible without requiring joins, whereas in the OntoRel implementation, even when using materialized views, retrieving such information minimally requires two joins

due to the ontology structure. Thus, the overhead introduced by entity traversals outweighs the potential benefits of materialization for simple attribute selection.

Moreover, the layers have a significant advantage in overcoming the structural constraints of the original database. Specifically, in a production setting, the source schema frequently prohibits structural adjustments, such as creating an index. OntoRel query layers solve this problem by allowing indexing on the Materialized-Path layer. This results in significant performance gains without the need to modify the source database.

Ultimately, choosing to materialize the views should be guided by the performance and data freshness requirements of the application domain. Not materializing the views can be adequate in situations where slower execution times are acceptable, but up-to-date data is essential. Finally, carefully balancing user expectations, semantic expressiveness, query performance and infrastructure constraints are essential for choosing an implementation.

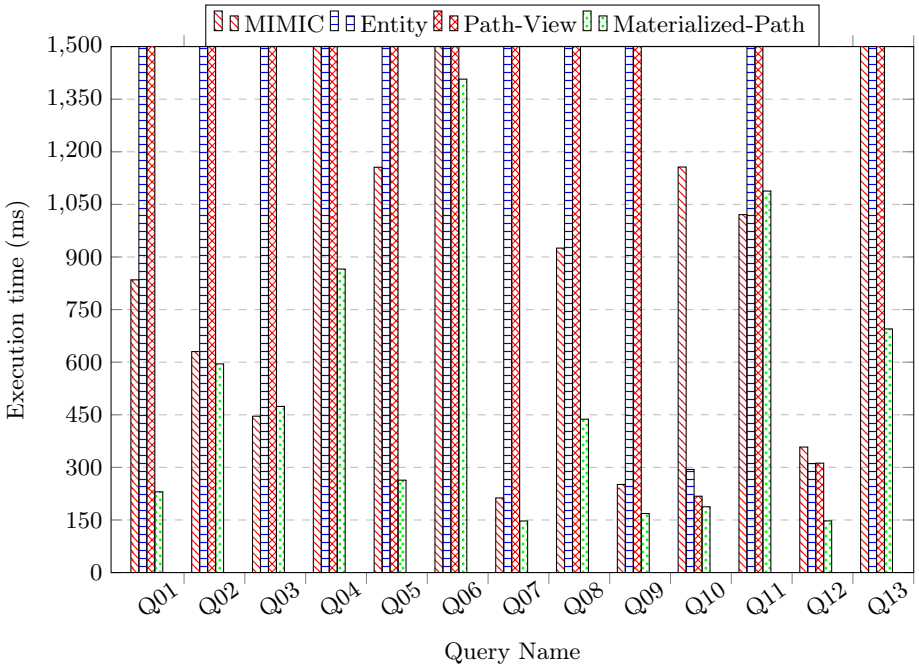


Fig. 3: Execution time for 13 queries.

3.3 Limitations and future work

Our approach offers an ontology-based and semantically coherent way to query heterogeneous relational databases. While initial results are encouraging, several areas require further development. To begin, we will leverage the power of logical reasoning to implement *path shortcuts* and graph summarization methods

[9]. These techniques will semantically reduce unnecessary joins, improving the efficiency of query execution. Then, we plan to automate the generation of query layers by analyzing user requests to identify common paths between variables. Ultimately, users will be able to submit a request for each variable while the system generates the necessary views in the layers. At the moment, users interact with the tool through a command line by choosing a request pattern and providing the set of input parameters. A graphical interface is under development to improve usability. Also, our approach allows data analysts to request data from one or multiple sources using the same ontology to compute one or multiple variables. However, sending the queries to each RDBMS is still manual. Development is needed to automatically adapt the SQL dialect to the target RDBMS, send the query and combine the results. Finally, we plan to evaluate the trade-offs between performance gains and view maintenance costs, and to investigate indexing strategies for materialized views to enhance query efficiency in a distributed environment.

4 Related work

Given the increasing complexity of databases and the expanding demand for flexible data access, various tools have been created to simplify querying both relational and ontological databases [10]. To better position our approach, we conducted a scoping review of 59 papers published between 2013 and 2023. We found various approaches for facilitating query formulation and generation. Among the examined solutions, we selected six that best addressed the requirements[5]. The solutions can be categorized into three main approaches. **Manual query design**, such as Ontop [1] and OntoGrate [4], provides powerful semantic capabilities. However, they requires advanced knowledge of the ontology and RDF query language (SPARQL) posing a barrier for non-technical users. **Visual query design**, such as OptiqueVQS [18] and DAFO [12], improves usability but offers limited expressiveness. They support only simple conjunctive queries that do not involve complex operations (e.g., aggregations, negations, universal quantification) or nested constructs. This restricts their applicability in complex scenarios. **Natural language interfaces**, such as Von-QBE [13] and ATHENA++ [17], translate user queries expressed in natural language into executable code using semantic parsing and ontological reasoning to capture user intent. However, they struggle with linguistic ambiguity and lack mechanisms to guide users in refining or correcting misunderstood queries. Yet, the translator is limited to English.

To compare these approaches, we established a comprehensive evaluation using 80 criteria grouped into eight main functional and non-function requirements:

[5] The survey full report is available on Github : https://github.com/OpenLHS/OntoRelQuery/blob/main/doc/survey-report.pdf

- **FR1 — Ontology processing:** the system must handle ontological structures efficiently, including graph traversal, multi-ontology management, detection of relationship patterns and reasoning.
- **FR2 — Query design:** the system must provide intuitive interfaces for non-technical users to formulate a wide range of queries.
- **FR3 — Query generation:** the system must generate optimized and syntactically correct SQL or SPARQL queries tailored to users' specific needs.
- **FR4 — Result processing:** the system must guarantee the execution of the query in one or multiple DBMS with consolidation of results, including presentation of the data and metadata in multiple output formats.
- **NFR1 — Availability and maintainability:** the system must offer different distribution options and comprehensive technical documentation for long-term maintainability.
- **NFR2 — Compatibility:** the system must be designed for easy integration with various technological environments, including relational and NoSQL databases.
- **NFR3 — Interoperability of query languages**: the system must support code generation and translation for multiple query languages.
- **NFR4 — Extensibility:** the system must allow the addition of new functionalities via a modular extension mechanism and offer advanced configuration possibilities.

Req.	Ours	ATHENA++	VON-QBE	OptiqueVQS	DAFO	OntoGrate	Ontop
FR1. (/8)	6	5	5	5	4	4	5
FR2. (/12)	9	3	5	7	5	2	4
FR3. (/25)	21	17	4	7	9	6	16
FR4. (/14)	11	4	1	5	3	2	8
Total FR (/59)	47	29	15	24	21	14	33
NFR.1 (/9)	5	3	5	5	3	3	8
NFR.2 (/6)	2	2	2	2	1	1	3
NFR.4 (/2)	2	0	1	2	0	0	2
Total NFR (/17)	9	5	8	9	4	4	13
Total (/76)	**56**	**34**	**23**	**33**	**25**	**18**	**46**

Fig. 4: Requirements evaluation total scores

Figure 4 presents the scores. We compute for each solution a score for each requirement (+1 if the criterion is met and 0 otherwise, see Appendix for details). None of the solutions meets 100% of the criteria. Our solution **OntoRelQuery** (56/76) and Ontop (46/76) obtained a high score covering more than 50% of the functional and non-functional criteria. Ontop stands out in non-functional

requirements because it has been around for 15 years, while others face limitations that may hinder large-scale deployment. **OntoRelQuery** is available on GitHub and is built using open source libraries; thus, it is easy to reuse and extend. Most solutions support basic ontology processing (FR1). ATHENA++ stands out for its use of reasoning capabilities to optimize the graph traversal. **OntoRelQuery** offers options to customize the graph traversal algorithms such as selecting the type of edges to include (inheritance and properties of interests). OptiqueVQS offers the most complete support—through guided assistance and graphical visualization, while the others lack broader assistance features (FR2). **OntoRelQuery** offers request patterns presented as questions & answers enriched with suggestions guiding the user in building the request. Also, the query design is iterative, meaning that the user can visualize the query graph and add new components to refine the request. ATHENA++ and Ontop provide advanced support for SQL features (i.e. negation, nested queries) and the others are limited to basic operators (FR3). **OntoRelQuery** stands out by offering support to common table expressions and set operations (union, intersection and difference) to combine multiple paths. Finally, all solutions show limited features for result processing (FR4). Ontop stands out by handling distributed query execution on various RDBMS. **OntoRelQuery** offers many features for processing results that include the data and the metadata forming a comprehensible and reusable output. These include the query graph, the associated ontological metadata, the data in csv format.

5 Conclusion

Querying integrated data using an ontology provides a more comprehensive understanding of the analytical variables, making it easier and faster to extract valuable insights. Our approach combines three complementary tasks: (1) a query design expressed through ontologies using request patterns with customized algorithms based on the graph structure, (2) a query implementation that automatically generates executable code in multiple query layers to support efficient storage and execution, and (3) a data extraction that dispatches queries to the target RDBMS, runs the queries, combines their results, and delivers them to the end user. Our approach allows users to express rich, interoperable and reusable queries using ontology terms, without requiring detailed knowledge of the database schema or query languages. As a result, this approach enhances both the usability and the expressiveness of querying by extracting meaningful and shareable information.

Acknowledgments. This study is funded by Ministère de l'Économie, de l'Innovation et de l'Énergie du Québec and Natural Sciences and Engineering Research Council of Canada. We thank Prof. Luc Lavoie of computer science and Prof. Jean-François Ethier of Medicine at the Université de Sherbrooke for guidance on data modelling and clinical questions. Also, we thank Paul Fabry PhD student for building the ontology and Samuel Dussault PhD student for mapping the data.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: Answering sparql queries over relational databases. Semantic Web **8**(3), 471–487 (Dec 2016). https://doi.org/10.3233/sw-160217
2. Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. Communications of the ACM **13**(6), 377–387 (1970). https://doi.org/10.1145/362384.362685
3. Date, C.J.: SQL and Relational Theory: How to Write Accurate SQL Code. O'Reilly Media, Inc., 3rd edn. (2015)
4. Dou, D., Qin, H., Lependu, P.: Ontograte: Towards automatic integration for relational databases and the semantic web throught an ontology-based framework. International Journal of Semantic Computing **04**(01), 123–151 (Mar 2010). https://doi.org/10.1142/s1793351x10000961
5. Guarino, N.: The Ontological Level: Revisiting 30 Years of Knowledge Representation, p. 52–67. Springer Berlin Heidelberg (2009). https://doi.org/10.1007/978-3-642-02463-4_4
6. Hoseini, S., Theissen-Lipp, J., Quix, C.: A survey on semantic data management as intersection of ontology-based data access, semantic modeling and data lakes. Journal of Web Semantics **81**, 100819 (Jul 2024). https://doi.org/10.1016/j.websem.2024.100819
7. Johnson, A., Bulgarelli, L., Pollard, T., Horng, S., Celi, L.A., Mark, R.: MIMIC-IV (2020). https://doi.org/10.13026/A3WN-HQ05, `https://physionet.org/content/mimiciv/0.4/`
8. Khnaisser, C., Lavoie, L., Fraikin, B., Barton, A., Dussault, S., Burgun, A., Ethier, J.F.: Using an ontology to derive a sharable and interoperable relational data model for heterogeneous healthcare data and various applications. Methods of Information in Medicine **61**(S 02), e73–e88 (Jun 2022). https://doi.org/10.1055/a-1877-9498
9. Liu, Y., Safavi, T., Dighe, A., Koutra, D.: Graph summarization methods and applications: A survey. ACM Computing Surveys **51**(3), 1–34 (Jun 2018). https://doi.org/10.1145/3186727
10. Masmoudi, M., Ben Abdallah Ben Lamine, S., Karray, M.H., Archimede, B., Baazaoui Zghal, H.: Semantic data integration and querying: A survey and challenges. ACM Computing Surveys **56**(8), 1–35 (Apr 2024). https://doi.org/10.1145/3653317
11. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) (2012), `https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/`
12. Pankowski, T., Bąk, J.: DAFO: An Ontological Database System with Faceted Queries, p. 152–155. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-32327-1_30
13. Peres, L., L. Coelho da Silva, T., Macedo, J., Araujo, D.: Ontology-Schema Based Query by Example, p. 204–212. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-33223-5_17
14. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking Data to Ontologies, p. 133–173. Springer Berlin Heidelberg (2008). https://doi.org/10.1007/978-3-540-77688-8_5
15. Rönnbäck, L., Regardt, O., Bergholtz, M., Johannesson, P., Wohed, P.: Anchor modeling — agile information modeling in evolving data environ-

ments. Data and Knowledge Engineering **69**(12), 1229–1253 (Dec 2010). https://doi.org/10.1016/j.datak.2010.10.002
16. Schuler, J.C., Ceusters, W.M.: The Problems of Realism-Based Ontology Design: a Case Study in Creating Definitions for an Application Ontology for Diabetes Camps. AMIA Symposium **2017**, 1517–1526 (2017). https://doi.org/10.1145/1321631.1321672
17. Sen, J., Lei, C., Quamar, A., Özcan, F., Efthymiou, V., Dalmia, A., Stager, G., Mittal, A., Saha, D., Sankaranarayanan, K.: Athena++: natural language querying for complex nested sql queries. Proceedings of the VLDB Endowment **13**(12), 2747–2759 (Aug 2020). https://doi.org/10.14778/3407790.3407858
18. Soylu, A., Kharlamov, E., Zheleznyakov, D., Jimenez-Ruiz, E., Giese, M., Skjæveland, M.G., Hovland, D., Schlatte, R., Brandt, S., Lie, H., Horrocks, I.: Optiquevqs: A visual query system over ontologies for industry. Semantic Web **9**(5), 627–660 (Aug 2018). https://doi.org/10.3233/sw-180293
19. Spanos, D.E., Stavrou, P., Mitrou, N.: Bringing relational databases into the semantic web: A survey. Semantic Web **3**(2), 169–209 (2012). https://doi.org/10.3233/sw-2011-0055
20. Taseen, R., Ethier, J.F.: Expected clinical utility of automatable prediction models for improving palliative and end-of-life care outcomes: Toward routine decision analysis before implementation. Journal of the American Medical Informatics Association **28**(11), 2366–2378 (2021). https://doi.org/10.1093/jamia/ocab140

Appendix

This appendix presents the evaluation of the existing solutions. Figure 5 shows the results for the non-functional requirements and Figure 6 shows the results for the functional requirements.

Legend: ✓ indicates that that the criterion is met. ∼ indicates that the criterion is partially supported. ? indicates that the information in the article is insufficient to evaluate its suitability for the criterion. ∅ indicates that the criterion is not applicable.

Req.	Criterion	Sub-criterion	OntoRelQuery	ATHENA++	VON-QBE	Optique-VQS	DAFO	OntoGrate	Ontop
NFR1	Distribution mode	Open source	✓	✗	✓	✓	✗	✗	✓
		Proprietary with API	✗	✗	✗	✗	✗	✗	✓
		Commercial license	✗	✓	✗	✗	✗	✗	✗
	Support and maintenance	Technical documentation	✓	✓	✓	✓	✓	✓	✓
		Tutorials and examples	✓	?	✓	✓	✓	✓	✓
		Regular updates	~	?	?	?	✗	?	✓
	Community support	Active community	~	✗	~	~	✗	✗	✓
		Community platforms	GitHub	✗	GitHub	GitHub	✗	✗	GitHub
	Development technology	Programming language	Java	Java	Java	Java	C#	Java	Java
NFR2	Supported DBMS	Relational databases	✓	✓	~	✗	✓	✗	✓
		NoSQL databases	✗	✗	✗	✗	✗	✗	✗
		Triplestores	✗	✗	✓	✓	✗	✓	✓
	Portability	Multi¯platform	✓	✓	✓	✓	?	?	✓
		Cloud deployment	~	Ø	Ø	Ø	Ø	Ø	Ø
	Integration	Available APIs	✗	✗	✗	✗	✗	✗	✗
NFR3	Input languages	Formulation languages	Questions & Answers Keyword & Faceted search	Natural Language	Natural Language Keyword Search	Faceted search	Keyword Search	SPARQL	SPARQL
	Output languages	Generated languages	SQL	SQL	SPARQL	SPARQL	SQL	SQL	SQL
NFR4	Functional extensions	Addition of features	✓	?	✓	✓	?	?	✓
		Custom configuration	✓	?	?	✓	?	?	✓

Fig. 5: Non-functional requirements evaluation

Req.	Criterion	Sub criterion	Ours	ATHENA++	VON-QBE	OptiqueVQS	DAFO	OntoGrate	Ontop
FR1	Ontology graph traversal	Path traversal	✓	✓	✓	✓	✓	✓	✓
		Traversal algorithms	✓	✓	✓	¯	¯	✓	✓
		Variable path exploration	✓	~	✓	✓	✓	✗	✗
	Structural analysis	Detection of ontological components	✓	✓	✓	✓	✓	✓	✓
		Pattern identification	✗	✗	✗	?	?	?	✗
		Cycle detection	✓	✓	✗	✓	?	?	✓
	Traversal optimization	Customized traversal algorithms	✓	✗	✓	?	¯	✓	✗
		Ontological graph reasoning	✗	✓	✗	✓	✓	✗	✓
FR2	Request expression methods	By graphical interface	✗	✗	✗	✓	✓	✗	✓
		Keyword¯based system	✓	✗	✓	✗	✓	✗	✗
		By guided question¯answer (Q&A)	✓	✗	✗	✗	✗	✗	✗
		By natural language (NL)	✗	✓	✓	✗	✗	✗	✗
		By custom language	✓	?	✗	✗	✗	✓	✗
	Request formulation assistance	Contextual auto¯completion	✓	Ø	Ø	✓	Ø	Ø	Ø
		Semantic suggestions	✗	Ø	✓	✓	✓	Ø	Ø
		Entity recognition	✓	✓	✓	✓	✓	✓	✓
		Interactive visualizations during	✓	✗	✗	✓	✗	?	✗
		Syntactic error detection and reporting	✓	✗	?	?	✗	?	✓
		Semantic error detection and reporting	✓	✓	?	?	✗	?	✓
	Progressive request formulation	Iterative formulation	✓	✗	✓	✓	✓	?	Ø
		History management	✗	¯	✗	✓	?	?	Ø
FR3	Data restriction capabilities	Simple restrictions	✓	✓	✓	✓	✓	✓	✓
		Composite restrictions	✓	✓	?	✓	✓	✓	✓
		Pattern¯based restriction	✓	✓	?	✓	✓	¯	✓
		Set¯based restriction	✓	✓	?	✓	✓	?	✓
	Data selection	Selective projection	✓	✓	✓	✓	✓	✓	✓
		Element renaming	✓	✓	✓	✗	✓	✗	✓
		Ordering	✓	✓	✗	✗	✗	✗	✓
		Result limitation	✓	✗	✗	✗	✗	✗	✓
	Analytical functions	Data aggregation	✓	✓	✗	✗	?	?	✓
		Grouping	✓	✓	✗	✗	?	?	✓
		Conditions on aggregates	✓	✓	✗	✗	?	?	✓
		Window function	✗	✗	✗	✗	?	?	✗
	Join operations	Joins	✓	✓	✓	✓	✓	✓	✓
		Outer joins	✓	✓	✗	?	?	?	✓
		Existential operators	✓	✓	✗	✗	✓	?	✗
		Subqueries	✓	✓	✗	✗	?	✓	✓
	Set operations	Union	✓	✓	✗	✗	✓	✓	✓
		Intersection	✓	✓	✗	✗	?	?	✗
		Difference	✓	✓	✗	✗	?	?	✗
	Advanced structures	Common table expressions (CTE)	✓	¯	✗	✗	✗	✗	✗
		String manipulation	✗	¯	✗	✗	?	✗	¯
		Conditional expressions	✓	✗	✗	✗	?	✗	✓
	Abstraction and reuse	Parameterizable query	✗	✗	✗	✗	?	¯	✗
		Abstraction	✓	✗	✗	✓	?	✗	✗
FR4	Visualization of generated queries	SQL/SPARQL code display	✓	✓	✓	✓	✓	?	✓
		Graphical representation	✓	¯	✗	✓	✓	?	?
	Artifact export formats	Source code export	✓	✓	?	✓	?	?	✓
		Visual diagrams	✓	✗	?	?	?	?	✗
		Associated metadata	✓	✗	?	?	?	?	Ø
		Complete documentation	✓	✗	?	?	?	?	✗
	Visualization of execution results	Tabular display	✓	✓	?	?	?	?	✓
		Textual representations	✗	✗	?	?	?	?	✓
		Interactive visualizations	✗	✗	?	?	?	?	Ø
	Result export options	Structured formats	✓	?	?	?	?	?	Ø
		Document formats	✓	?	?	?	?	?	Ø
	Execution capabilities	Local execution ¯ same DBMS	✓	✓	?	✓	✓	✓	✓
		Local execution ¯ multiple DBMS	✗	?	?	✓	?	✓	✓
		Distributed execution ¯ same DBMS	✓	?	?	?	?	?	✓
		Distributed execution ¯ multiple DBMS	✗	?	?	?	?	?	✓

Fig. 6: Functional requirements evaluation

Understanding Linguistic Schema Ambiguity in Natural Language Interfaces to Databases

Markus Cservenka[1][0009-0003-9997-8956]

[1] Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria
h11706187@s.wu.ac.at

Abstract. Natural language interfaces to relational databases have advanced rapidly in recent years, primarily due to progress in large language models. Yet most Text-to-SQL benchmarks continue to rely on database schemas with linguistically transparent, human-readable names. In practical database settings, however, schemas often contain abbreviated, compressed, or domain-specific naming conventions that introduce linguistic ambiguity and weaken the alignment between user queries and schema elements. This paper examines linguistic schema ambiguity as a systematic and controllable factor influencing the performance of natural-language database interfaces. We introduce a deterministic, heuristics-driven schema transformation pipeline that injects realistic naming ambiguity at multiple levels while maintaining schema validity and executability. To measure ambiguity, we propose the Schema Ambiguity Score (SAS), an embedding-based metric. Using this framework, we generate multi-level ambiguity variants of three widely used benchmarks – Spider, BIRD-SQL, and KaggleDBQA – and evaluate both closed-source and open-source large language models in a unified zero-shot setting. Our results reveal that the impact of schema ambiguity varies substantially across datasets and models: while some models exhibit substantial robustness to schema obfuscation, others degrade noticeably as schema explicitness decreases.

Keywords: Text-to-SQL, Natural Language Database Interfaces, Practical Database Applications

1 Introduction

Text-to-SQL is defined as the task of translating natural language questions into executable SQL statements given a particular relational database. It combines natural language understanding as well as semantic parsing with database systems, and has received substantial attention due to its practical relevance for enabling business users to access structured data. Recent advances in large language models (LLMs) have led to substantial enhancements in

performance, with state-of-the-art systems [3, 10, 13, 14] achieving high execution accuracy on established benchmarks [8, 19].

Despite this progress, current Text-to-SQL benchmarks capture only a limited subset of the challenges encountered in real-world deployments. While conversational samples [9, 17, 18] were introduced relatively early, experiments on other obstacles – such as multi-language queries [1, 6, 11] and diverse question types [4, 15, 16, 20] – have only been conducted more recently. Linguistic schema ambiguity represents another significant challenge of real-world settings. In practical database systems, schema object names are rarely as explicit, consistent, or human-readable as those found in curated research datasets [5, 7, 8, 19]. Instead, developers commonly utilize abbreviations, compressed identifiers, legacy naming conventions, as well as domain-specific or even idiosyncratic prefixes and suffixes to manage large schemas and evolving systems [2, 12]. These naming practices introduce linguistic schema ambiguity, weakening the lexical richness that Text-to-SQL systems rely on for schema linking and semantic grounding.

Existing benchmarks such as Spider [19] and BIRD-SQL [8] only partially reflect this reality. While they emphasize cross-domain generalization, query complexity, and value grounding, they largely assume explicit and semantically transparent schema object names. As a result, an important dimension of real-world difficulty – how models behave when schema names are linguistically fuzzy or ambiguous – remains underexplored and poorly quantified.

This paper addresses this gap by introducing a controlled and reproducible framework for studying schema ambiguity in Text-to-SQL. At a high level, we propose a deterministic procedure for transforming existing database schemas in order to introduce realistic linguistic ambiguity at multiple levels. We further construct a quantitative metric for measuring schema ambiguity based on lexical semantics, and conduct a systematic evaluation of how increasing schema ambiguity affects modern LLMs across multiple datasets. Our central research question is: „How does increasing linguistic ambiguity in database schema object names affect the performance and robustness of Text-to-SQL models?“

To answer this question, we apply our transformation procedure to three widely used benchmarks – being Spider, BIRD-SQL, and KaggleDBQA – creating multi-level ambiguity variants while preserving database executability and question semantics. We then evaluate both a closed-source model as well as a strong open-source model under a unified zero-shot setting. Subsequently, we analyze performance trends and error patterns as schema ambiguity increases. In summary, this paper aims at making the following contributions:

1. A deterministic, heuristics-based schema transformation pipeline that introduces realistic linguistic ambiguity into database object names while preserving schema validity and reproducibility.
2. The Schema Ambiguity Score (SAS), an embedding-based metric that quantitatively measures the linguistic explicitness of schema object names and enables comparison across datasets and ambiguity levels.

3. Multi-level ambiguity variants of three established Text-to-SQL benchmarks, providing a new evaluation dimension without altering query distributions or database contents.
4. An empirical study of schema ambiguity effects on both closed-source and open-source LLMs, revealing model- and dataset-dependent robustness patterns and highlighting limitations of standard evaluation metrics under realistic conditions.

The source code, data, and other artifacts have been made available at https://github.com/mcservenka/t2sql-schema-ambiguity.

2 Related Work

Text-to-SQL has traditionally been studied as a semantic parsing problem, where natural language questions are mapped to executable SQL queries over relational databases. Early approaches [13, 19, 22] relied on supervised neural parsers with explicit schema encoding and schema-linking components. More recently, large language models have reshaped the field by enabling strong zero-shot and few-shot performance through in-context learning and prompt engineering [6, 14]. LLM-based systems often leverage structured schema representations – typically in Data Definition Language (DDL) or other structured formats [6] – and benefit from extensive pretraining on code and natural language, achieving competitive or state-of-the-art execution accuracy without task-specific fine-tuning. While these advances have substantially improved performance, they have also shifted the evaluation focus toward benchmark-driven results under idealized schema conditions.

Benchmark design plays a central role in Text-to-SQL research. WikiSQL [29] introduced large-scale supervision with relatively simple queries and single-table schemas, emphasizing lexical matching and aggregation. Spider [26] advanced the field by introducing complex, cross-domain databases with multi-table queries and compositional structures, essentially becoming a standard for evaluation. Subsequent benchmarks have expanded this landscape. BIRD-SQL [12] emphasizes realistic databases and value grounding at scale, while Spider-2.0 [11] targets more challenging reasoning and robustness settings. KaggleDBQA [10] draws from real-world Kaggle databases and includes natural language questions written without explicit schema awareness.

Despite these advances, most benchmarks share an implicit assumption: schema object names are linguistically explicit and semantically transparent. Table and column names in Spider and related datasets are typically human-readable, well-segmented, and closely aligned with natural language expressions used in questions. Even benchmarks that aim for realism, such as BIRD-SQL or KaggleDBQA, only partially reflect the degree of abbreviation, compression, and idiosyncratic naming conventions found in production systems [1, 3, 16]. As a result, linguistic schema ambiguity is underrepresented as an explicit evaluation dimension. A substantial body of work has explored schema representation and schema linking, including graph-based encodings [19], relation-aware attention [22], and prompt-

based schema serialization [5, 6, 14]. These methods primarily aim to improve how models use schema information, rather than to question how explicit that information is presented. Related efforts on ambiguous or unanswerable questions [21, 23, 27] focus on query intent ambiguity or missing information in the question text, not on ambiguity arising from the schema itself.

There is also limited work on schema anonymization or obfuscation, typically motivated by privacy or domain transfer [4, 20], yet these approaches are usually random or binary – in the sense of either original or anonymized without varying degrees of ambiguity. They do not provide graded difficulty levels, do not aim for linguistic realism, and lack quantitative measures of the ambiguity they introduce.

In this work, we focus on linguistic schema ambiguity, defined as the degree to which table and column names deviate from standard natural language words and thus weaken lexical grounding between questions and schema objects. Linguistic explicitness refers to schema names that are easily interpretable by humans and language models alike (e.g., customer_id, order_date), whereas ambiguous names include abbreviations, truncated forms, flattened identifiers or opaque tokens (e.g., custid, orddt, c01). This notion is distinct from structural complexity or value ambiguity and targets a specific, underexplored dimension of Text-to-SQL difficulty.

In contrast to prior work, we treat schema ambiguity as a first-class, controllable variable. Our work fills this gap by introducing a deterministic, multi-level schema transformation procedure that produces realistic linguistic ambiguity, a quantitative metric to measure schema explicitness, and benchmark variants that enable systematic evaluation of Text-to-SQL robustness under increasing schema ambiguity. Together, these contributions complement existing benchmarks and provide new tools for analyzing the behavior of modern LLM-based Text-to-SQL systems beyond idealized schema assumptions.

3 Schema Ambiguity Pipeline

This section introduces a deterministic, heuristics-based schema ambiguity pipeline – in the following also referred to as schema anonymization pipeline – designed to generate controlled and realistic linguistic ambiguity in existing relational database schemas. The pipeline transforms table and column names while preserving schema executability, referential consistency, and compatibility with existing Text-to-SQL benchmarks.

The primary design goal is to simulate developer-style naming practices [3, 16] – such as abbreviations, compressed identifiers, and structural noise – without relying on randomness or learned models. This enables reproducible experimentation under controlled ambiguity conditions.

3.1 Design Principles

The schema anonymization pipeline is designed to introduce linguistic ambiguity into database schemas in a controlled, realistic, and reproducible manner. Its central objective is not to arbitrarily obfuscate schemas, but to apply naming conventions commonly found in real-world operational and analytical databases, where object names are often compressed, abbreviated, or shaped by legacy constraints rather than optimized for semantic clarity.

A core principle of the pipeline is determinism. For a fixed database schema and ambiguity level, the transformation outcome is fully reproducible. No stochastic sampling or learning-based components are employed. Instead, variability is derived from deterministic functions, ensuring that experimental results can be replicated and that differences across ambiguity levels can be attributed solely to controlled design configurations.

Closely related is the principle of pseudo-probabilistic behavior without randomness. The pipeline selects among multiple ambiguity operators, defined in section 3.2.2, using level-dependent preferences that resemble probabilistic sampling, while the actual selection mechanism is deterministic. Operator choice considers applicability constraints and is driven by a hash function that maps the object name and the applied ambiguity level to a float between 0 and 1. The resulting float is then used to select an operator based on level- and applicability-dependent weights. This produces a behavior that mimics probabilistic diversity while remaining fully auditable and reproducible.

Another key principle is linguistic realism. Transformations are motivated by naming practices observed in real database systems [2, 12], such as the use of abbreviations, vowel removal, case normalization, and the addition of structural prefixes or suffixes. These operations aim to preserve the syntactic plausibility of identifiers while gradually reducing their semantic transparency. At the same time, aforementioned feasibility constraints ensure that transformations are only applied when they produce valid and interpretable identifiers, preventing pathological outputs such as excessively short tokens or malformed names.

The pipeline is further designed to support controlled ambiguity scaling. Rather than applying a uniform anonymization strategy, ambiguity is introduced through discrete levels that reflect increasing degrees of ambiguity. This approach enables systematic analysis of how Text-to-SQL systems behave as schemas become progressively less explicit, without conflating mild normalization with extreme anonymization.

Finally, schema integrity is strictly preserved. All transformations are collision-aware, SQL-safe, and applied consistently across database instances and gold SQL queries. The resulting schemas remain executable and compatible with existing evaluation frameworks, allowing direct comparison with original benchmarks.

3.2 Transformation Procedure

For each database schema, we first construct a normalized schema representation in dictionary form and store it as a JSON file. All table and column names within the schema are

then collected into a single set, ensuring that each unique name is processed exactly once. Hence, table and column names are not distinguished, meaning identical origin forms always receive identical transformations, regardless of their role in the schema.

Name Features. Each name is tokenized using underscore separation, whitespace, and camel-case boundaries. For every token, we heuristically compute a set of lexical features, including character length, vowel and consonant counts, alphanumeric ratios, as well as additional indicators such as *is_numeric*, *is_abbreviation*, *looks_like_id*, or *is_single_letter*. These token-level features are aggregated into name-level features, including the number of tokens, presence of abbreviations, vowel richness, and overall alphabetic density. Additionally, we classify the pattern of the original name based on predefined categories – such as lowercase, uppercase, camel-case, snake-pattern and mixed.

These features form the basis for determining which ambiguity transformations are applicable for a given name

Ambiguity Operators. The pipeline defines four ambiguity operators, each modeling a distinct linguistic phenomenon observed in real database schemas:

- Abbreviation: Long or semantically transparent names are shortened by truncation or replaced using a small dictionary of common developer abbreviations (e.g., amount to amt, number to num, quantity to qty). Numeric or identifier-resembling tokens are preserved, and already abbreviated names remain unchanged.
- Vowel Dropping: Internal vowels are removed while keeping the first character and adhering to a minimum length constraint. If vowel removal yields a name that is too short or unreadable, one vowel is deterministically restored.
- Case Flattening: Naming conventions are normalized depending on the ambiguity level. At lower levels, camel-case names are converted to underscore-separated lowercase forms. At higher levels, all separators are removed, producing fully flattened identifiers.
- Noise Wrapping: Names are wrapped with deterministic prefixes or suffixes. At lower ambiguity levels, these are semantically meaningful – such as *tbl_* or *col_*, whereas higher levels may introduce arbitrary character sequences.

Each operator is subject to feasibility constraints derived from the name's features. For example, abbreviation is infeasible for very short tokens or numeric identifiers, while vowel dropping is avoided for already compressed names.

For each ambiguity level $L \in \{L0, L1, L2, L3\}$, the pipeline defines a preference distribution P over operators o, including the identity operator – which applies no transformation. These distributions encode how strongly each level favors different types of ambiguity but do not constitute hard rules. For a given name n, infeasible operators – determined by $f(o,n)$ – are masked out by assigning them zero weight. The remaining weights are renormalized to form a valid probability distribution:

$$P(o|n,L) = \frac{w_L(o) \times 1[f(o,n)]}{\sum_{o'} w_L(o') \times 1[f(o',n)]} \quad (1)$$

Rather than sampling randomly, the pipeline computes a deterministic hash of the tuple (*n*, *L*), which yields a value between 0 and 1. This value is used to select an operator from the cumulative distribution. As a result, operator choice behaves probabilistic, while remaining fully deterministic and reproducible.

If all non-identity operators are infeasible, the identity operator is selected by construction.

Ambiguity Levels. The four ambiguity levels correspond to increasing degrees of linguistic obfuscation:

- L0 (Original): No transformation is applied.
- L1 (Minimal ambiguity): Mostly unaltered. Only light normalization or short abbreviations are applied.
- L2 (Moderate ambiguity): A balanced mix of abbreviation, vowel dropping, case flattening, and limited noise.
- L3 (High ambiguity): Strong compression and obfuscation. Most names are transformed aggressively, with frequent noise wrapping.

For example, in the Spider [19] dataset's *world_1* database, the column name *localname* remains unchanged at levels L0 and L1, is transformed into *loc_name* at L2, and further compressed to *lcl_name* at level L3. Similarly, in BIRD's [8] superhero database, the column *attribute_value* is preserved at L0 and L1, becomes *attrbt_vla* at L2 through abbreviation and vowel removal, and is transformed into *attr_val* at L3, reflecting stronger compression while remaining syntactically valid.

Collision Resolution. After transformation, each generated name is checked against previously produced names within the same schema. If a collision occurs – for instance the names *cite* and *cited* both map to *ci* – a deterministic numeric suffix is appended (*ci_1*). Additionally, all transformed names are checked against SQL reserved keywords. If a conflict is detected – as in *notes* being transformed to *not* – a deterministic prefix (*not_f*) is added to ensure syntactic validity. This safety check is applied uniformly across all ambiguity levels.

3.3 Schema Reconstruction

The final output of the pipeline is a mapping from canonicalized original names to transformed names for each database schema. This mapping is applied consistently to all SQLite database files as well as all gold SQL queries. The SQL queries are rewritten using a SQL parser. All transformed queries are executed against the transformed databases to verify correctness. Only schemas and queries that pass this execution-based sanity check are included in the final datasets.

4 Schema Ambiguity Score

To systematically study the impact of schema naming ambiguity on Text-to-SQL performance, we introduce the Schema Ambiguity Score (SAS), a quantitative, embedding-based measure of linguistic explicitness in database schemas. SAS is designed to capture how easily schema object names can be grounded in natural language, independently of any particular model or task performance.

The score is completely deterministic, reproducible, and comparable across datasets as well as ambiguity levels. It operates directly on schema object names and is agnostic to database structure, values, or query content.

4.1 Overview

Given a dataset consisting of multiple database schemas, SAS is computed hierarchically. First, an ambiguity score is assigned to each token of a table or column name based on its semantic proximity to standard English vocabulary. Token-level scores are then aggregated into a name-level ambiguity score. Name-level scores are averaged to obtain a database-level ambiguity score, and finally, dataset-level SAS is computed as the mean over all databases in the dataset.

Importantly, SAS measures linguistic ambiguity, not task difficulty. It quantifies how far schema identifiers deviate from recognizable natural language expressions, thereby characterizing the degree of semantic significance available to Text-to-SQL systems.

For each database, all table and column names are extracted without de-duplication. Each name is tokenized following the same principles as described in section 3.2.1 for the anonymization pipeline – hence splitting on underscores, whitespace, and camel-case boundaries. All tokens are lowercased prior to further processing.

Let V denote a fixed English reference vocabulary consisting of 200,000 alphabetic words derived from the English *fastText* model. All non-alphabetic entries and words shorter than three characters are excluded. This vocabulary is fixed globally and shared across all datasets, ensuring reproducibility and comparability.

4.2 Token-Level Ambiguity

Each token t is embedded using pre-trained fastText Common Crawl embeddings. fastText is chosen because schema object names are short lexical units and rather independent, unlike complete sentences. Furthermore, fastText's sub-word modeling enables robust embeddings for abbreviations, truncated words, and out-of-vocabulary strings.

To quantify the semantic transparency of a token, we compute its maximum cosine similarity $sim_{max}(t)$ to any word in the reference vocabulary V:

$$sim_{max}(t) = \max_{w \in V} \cos(e_t, e_w) \quad (2)$$

where e_t and e_w denote the fastText embeddings of token t and vocabulary word w, respectively.

Cosine similarity ranges from −1 to 1, with higher values indicating stronger semantic relatedness. We convert this similarity into a raw ambiguity score:

$$A_{raw}(t) = \min(1 - sim_{max}(t), 1) \quad (3)$$

This formulation reflects the intuition that tokens with high similarity to known English words are less ambiguous, while tokens with low or negative similarity are highly ambiguous. Similarity values below zero are treated as maximally ambiguous, as they do not feature additional semantic structure beyond non-relatedness.

Empirical calibration reveals that raw ambiguity scores do not span the full range between 0 and 1. In particular, completely random strings do not yield raw ambiguity values close to 1, but rather cluster around intermediate values.

To address this, we introduce linear scaling based on two anchor points. The first anchor $A_{CLEAR}=0.0$ is obtained by sampling 5,000 standard English words from V, which consistently yield near-zero raw ambiguity. The second anchor $A_{NOISE}=0.6$ is obtained by sampling 5,000 random alphabetic strings, which empirically represent maximally uninformative tokens under fastText embeddings. The scaled ambiguity score is then computed as:

$$A_{scaled}(t) = \frac{A_{raw}(t) - A_{CLEAR}}{A_{NOISE} - A_{CLEAR}} \quad (4)$$

Finally, to prevent zero-valued tokens from collapsing higher-level scores, we enforce a lower bound:

$$A(t) = \max(A_{scaled}(t), \varepsilon) \quad (5)$$

where $\varepsilon = 0.1$. This ensures that even highly transparent tokens contribute a minimal amount of ambiguity, preserving the influence of other tokens in multi-token names.

4.3 Name-Level Ambiguity

Given a name n consisting of tokens $\{t_1,\ldots,t_k\}$, we define its name-level ambiguity score as the product of token-level ambiguities:

$$A(n) = \prod_{i=1}^{k} A(t_i) \quad (6)$$

The product aggregation is chosen deliberately over alternatives such as summation or averaging. A mean-based aggregation would treat a single clear token as sufficient to reduce overall ambiguity, even if the remaining tokens are highly opaque. In contrast, the product

formulation follows the intuition that multiple clear tokens jointly provide stronger semantic evidence, significantly reducing ambiguity, while multiple ambiguous tokens only gradually decrease ambiguity, which reflects the weak and diffuse information they provide.

This behavior aligns with the desired properties of the score. Names consisting of a single clear token receive low ambiguity scores, while names with multiple clear tokens receive exponentially lower scores. Names composed of noisy or abbreviated tokens yield high ambiguity scores, and mixed names naturally fall in between. The lower bound ε prevents clear tokens from nullifying ambiguity entirely and ensures that all parts of a name contribute to the final score.

4.4 Schema-Level Aggregation

For a given database d, the database-level ambiguity score is computed as the mean over all name-level ambiguity scores:

$$A(d) = \frac{1}{|N_d|} \sum_{n \in N_d} A(n) \tag{7}$$

where N_d represents the set of all table and column names in database d. Finally, the Schema Ambiguity Score (SAS) for a dataset containing multiple databases is defined as the mean over all database-level scores:

$$SAS = \frac{1}{|D|} \sum_{d \in D} A(d) \tag{8}$$

where D is the set of databases in the dataset. This hierarchical aggregation yields a single continuous score between 0 and 1 that characterizes the overall linguistic ambiguity of a dataset's schemas.

4.5 Interpretation

Low SAS values indicate schemas whose object names closely resemble standard English words and are therefore linguistically transparent. High SAS values correspond to schemas dominated by abbreviations, truncated forms, or opaque identifiers that provide weak lexical grounding signals.

SAS does not measure structural complexity, query difficulty, or model performance. Instead, it isolates a single dimension of schema quality: the degree to which schema object names are accessible through natural language semantics. In the following section, we use SAS to characterize benchmark variants and analyze how Text-to-SQL models behave under increasing schema ambiguity.

5 Benchmark Construction

Using the deterministic schema anonymization pipeline described in section 3, we construct multi-level ambiguity variants of three established Text-to-SQL benchmarks: Spider [19], BIRD-SQL [8], and KaggleDBQA [7]. For all datasets, we operate exclusively on the development sets, as these are used for evaluation in subsequent experiments.

According to the practice described in section 3.3, for each dataset we only transform the database schemas and corresponding gold SQL queries, while leaving the natural language questions and dataset splits unchanged. Consequently, the number of databases and question-query pairs – illustrated in Table 1– remains identical across all ambiguity levels.

As a result of the anonymization pipeline we receive four variants per dataset: the original schema (L0) and three increasingly ambiguous versions (L1–L3). To quantify the linguistic ambiguity introduced at each level, we compute the Schema Ambiguity Score (SAS) as defined in Section 4. The resulting SAS-values are shown in Table 2.

Across all three dataset families, the highest ambiguity is consistently observed at L3, confirming that the anonymization procedure successfully increases linguistic ambiguity on average and that SAS captures the intended direction of change. The occasional non-monotonic behavior between intermediate levels is expected rather than malfunctioning. The anonymizer applies deterministic string-level transformations, whereas SAS measures lexical ambiguity in embedding space, and these two mechanisms do not enforce strict monotonicity.

Table 1. Dataset Characteristics.

Dataset	Databases	Samples
Spider	20	1,034
BIRD	11	1,534
KaggleDBQA	8	457

Table 2. Schema Ambiguity Scores across all dataset variants.

Dataset	L0	L1	L2	L3
Spider	.031	.071	.122	.217
BIRD	.112	.208	.193	.256
KaggleDBQA	.171	.244	.211	.297

In practice, several effects contribute to this behavior. Token splitting may reduce ambiguity by exposing clear English sub-words, some abbreviations coincidentally resemble valid English words, and prefixes or suffixes often contribute less to embedding similarity than

the core token. Additionally, schemas that are already highly ambiguous in their original form – such as the *thrombosis_prediction* database in BIRD – may exhibit saturation effects, resulting in limited SAS variation across levels.

Overall, these benchmark variants provide a controlled and reproducible test environment for studying Text-to-SQL performance under systematically increasing schema ambiguity.

6 Experimental Setup

In this section we describe the models, prompting strategy, and evaluation protocol used to assess Text-to-SQL performance under varying levels of schema ambiguity.

6.1 Models

We evaluate two large language models that represent distinct and widely used paradigms in modern Text-to-SQL research. As a closed-source model, we use GPT-5.2, accessed via the official OpenAI API. As an open-source alternative, we use Llama-3.3-70B, accessed via the TogetherAI API. Collectively, these models represent the dominant architectural and training-related paradigms employed in contemporary LLM-based Text-to-SQL systems and enable a direct comparison between proprietary and open-weight models under identical experimental conditions.

Both models are evaluated exclusively on the development sets of the benchmark variants described in Section 5.

6.2 Prompting

All experiments are conducted in a zero-shot setting. No model receives task-specific fine-tuning, in-context examples, or dataset-specific adaptation. This choice ensures that observed performance differences can be attributed solely to schema ambiguity rather than memorization or example-driven reasoning.

The prompting procedure is fully standardized across models and datasets. We adopt a schema serialization format inspired by the M-Schema representation [3], which encodes tables and columns in a structured, hierarchical form. Each model request consists of two system messages and one user message. The first system message provides a general instruction for generating valid SQLite-compatible SQL queries. The second system message contains the serialized schema representation. The user message only contains the natural language question.

To enforce consistent and machine-readable outputs, we employ explicit tool-function instructions that constrain the model's response to a single dictionary with one field, namely {'sql': '...'}. This eliminates variability in output formatting and ensures reliable downstream

execution and evaluation. In case the particular model fails to format the output accordingly, we set the SQL-response to *Null*.

6.3 Evaluation Metrics

Model performance is assessed using Execution Accuracy (ExA). A prediction is considered correct if the generated SQL query executes successfully on the target database and produces the same result as the gold query.

For Spider and KaggleDBQA, we use the official Spider evaluation test suite [21]. For BIRD-SQL, we use the evaluation module provided with BIRD-SQL [8], which supports execution-based comparison across heterogeneous databases. Execution Accuracy is the predominant evaluation metric in Text-to-SQL research [3, 10, 13, 14] and is widely regarded as more meaningful than exact match accuracy, as it accounts for the existence of multiple semantically equivalent SQL formulations.

7 Results

This section presents the empirical results of our evaluation across all datasets, while incorporating the four schema ambiguity levels and prompting the abovementioned LLMs. We first report standard ExA results to remain comparable with prior work, then analyze the limitations of strict ExA on realistic datasets, and finally provide a detailed error analysis to explain the observed trends.

7.1 Overall Performance

Overall, we observe that the effect of schema ambiguity is highly model- and dataset-dependent and noticeably less uniform than one might expect given the substantial increase in linguistic ambiguity measured by SAS. Table 3 illustrates the dataset- and level-specific ExA scores for both LLMs.

On Spider, GPT-5.2 exhibits remarkable robustness to increasing schema ambiguity. Execution accuracy remains effectively stable across all ambiguity levels, with variations well within one percentage point between L0 and L3. Even under heavy schema obfuscation, GPT-5.2 maintains performance comparable to the original schema setting.

This suggests that, on a clean benchmark with well-specified questions and strong structural regularities, a frontier model can largely compensate for reduced schema explicitness. In contrast, Llama-3.3-70B shows a clear and monotonic decline as ambiguity increases. From L0 to L3, execution accuracy drops by more than four percentage points, despite starting from a similar baseline. This divergence highlights a meaningful capability gap: while both models perform competitively on explicit schemas, the open-source model is substantially more sensitive to obfuscated or abbreviated schema names.

Table 3. Execution Accuracy by LLM and Dataset Variant.

Dataset	Level	GPT-5.2	Llama-3.3-70B
Spider	L0	76.79	76.21
	L1	76.02	74.37
	L2	75.44	73.98
	L3	76.40	71.76
BIRD	L0	36.57	35.72
	L1	35.01	33.96
	L2	34.68	34.49
	L3	33.38	33.05
KaggleDBQA	L0	19.69	43.98
	L1	19.91	42.23
	L2	19.26	36.54
	L3	19.04	43.11

On BIRD-SQL, absolute performance is significantly lower for both models, reflecting the well-known difficulty of the dataset. Here, increasing schema ambiguity leads to a gradual degradation for both GPT-5.2 and Llama-3.3-70B. However, the extent of this effect is moderate compared to the overall difficulty of the task. The drop from L0 to L3 is on the order of three percentage points for GPT-5.2 and slightly less for Llama-3.3-70B. This pattern suggests that, on BIRD-SQL, schema naming ambiguity is not the dominant bottleneck. Instead, challenges such as complex value grounding, longer queries, and more complex reasoning requirements appear to play a larger role in limiting performance.

On KaggleDBQA, the results initially present an irritating picture. Under strict execution accuracy, GPT-5.2 achieves very low performance, around 20%, substantially below Llama-3.3-70B, which reaches over 40% ExA on the original schema. Moreover, GPT-5.2's performance shows little variation across ambiguity levels, seemingly contradicting expectations based on both model strength and the SAS increase. This behavior stands in contrast to the trends observed on Spider and BIRD-SQL and motivates a closer inspection of the evaluation protocol.

To better understand this discrepancy, we additionally compute the Soft Execution Accuracy (Soft-ExA), a projection-invariant diagnostic metric that treats a prediction as correct if the gold result can be obtained as a subset of the columns in the predicted result. Importantly, Soft-ExA closely tracks ExA on Spider and BIRD-SQL, serving as a sanity check that confirms consistent evaluation behavior on those datasets. On KaggleDBQA, however, Soft-ExA reveals a noticeably different picture for GPT-5.2. While strict ExA remains low, Soft-ExA is substantially higher and much closer to that of Llama-3.3-70B. Manual inspection confirms that GPT-5.2 frequently generates semantically correct queries that include additional projected columns, which are penalized by strict execution accuracy despite yielding valid and informative results.

These findings indicate that GPT-5.2's apparent underperformance on KaggleDBQA is largely an artifact of the evaluation metric rather than a failure of semantic understanding. In general, they highlight a limitation of execution accuracy when applied to realistic datasets with weakly specified questions and multiple acceptable query formulations.

7.2 Robustness to Schema Ambiguity

Taken together, the results reveal that robustness to schema ambiguity is not a universal attribute of modern LLMs when it comes to Text-to-SQL, but depends strongly on both the underlying model and the characteristics of the dataset. GPT-5.2 demonstrates a high degree of robustness on Spider, maintaining stable performance even as schema ambiguity increases substantially. This suggests that the model relies less on direct lexical matching between question tokens and schema object names and can instead exploit structural cues, learned SQL patterns, and universal semantic reasoning to compensate for reduced schema transparency.

Llama-3.3-70B, by contrast, exhibits a more traditional sensitivity profile. As schema names become increasingly obfuscated, performance decreases consistently, particularly on Spider, where structural complexity is relatively low and schema linking plays a central role. This behavior indicates a stronger dependence on explicit lexical cues and highlights a gap between closed-source models and current open-weight alternatives in handling reduced schema explicitness.

On BIRD-SQL, both models show similar trends, with ambiguity-induced degradation remaining secondary to the dataset's structural difficulty. This reinforces the view that schema ambiguity is only one of several interacting factors that determine Text-to-SQL performance and that its relative importance varies across benchmarks.

Overall, these findings challenge the assumption that increasingly ambiguous schemas necessarily lead to dramatic performance collapse. Instead, they suggest that modern LLMs – particularly frontier models – can partially internalize schema abstraction and tolerate substantial linguistic degradation. At the same time, the observed differences between models underline the continued relevance of schema explicitness as a controlled difficulty dimension, especially for evaluating robustness, generalization, and progress in open-source and smaller LLMs when confronted with Text-to-SQL.

7.3 Error Analysis

To better understand how schema ambiguity affects Text-to-SQL systems beyond aggregate execution accuracy, we conduct a structured error analysis using a two-layer system. At the first layer, erroneous model outputs are classified into three high-level outcome categories: invalid LLM output (IOE), SQL execution errors (EXE), and executable but incorrect results (IRE). At the second layer, execution errors are further subdivided into schema-related errors, structural SQL errors, and runtime or dialect-specific errors.

First-Layer Errors. At the highest level, both models overwhelmingly fail through executable but incorrect results rather than through invalid SQL or execution failures. However, the relative proportions differ substantially between models. Table 4 summarizes the error distributions aggregated across all datasets, grouped by model and ambiguity level.

For GPT-5.2, IREs account for more than 98% of all errors across all ambiguity levels. Execution errors remain rare, consistently around 1-1.5%, and invalid outputs are negligible. Importantly, these proportions remain stable as schema ambiguity increases from L0 to L3. This stability indicates that increasing schema ambiguity does not cause GPT-5.2 to produce malformed SQL or to reference non-existent schema elements. Instead, the model continues to generate executable queries, while errors primarily arise from semantic mismatches, such as incorrect joins, filters, or aggregations.

Table 4. Layer 1 Error Distribution by LLM and Level.

LLM	Level	IRE	EXE	IOE
GPT-5.2	L0	98.48	1.01	0.51
	L1	98.01	1.43	0.56
	L2	98.71	1.05	0.25
	L3	98.23	1.41	0.37
Llama-3.3-70B	L0	89.18	10.35	0.47
	L1	85.54	14.07	0.39
	L2	87.53	12.08	0.38
	L3	87.14	12.54	0.32

In contrast, Llama-3.3-70B exhibits a noticeably different error pattern. While incorrect results still dominate, execution errors constitute a much larger fraction of failures, ranging from approximately 10% at L0 to over 14% at L1 and remaining elevated at higher ambiguity levels. This shift indicates that increased schema ambiguity leads to greater execution instability for the open-source model, even when overall execution accuracy drops are moderate. Invalid outputs remain rare for both models, suggesting that output formatting and basic SQL generation are not the primary failure modes.

These aggregated patterns already explain a key observation from section 7.1: execution accuracy for GPT-5.2 remains stable because errors rarely arise from schema misuse or execution failures, whereas Llama-3.3-70B increasingly struggles at the execution level as schema explicitness decreases. Across both models and all datasets, executable but incorrect results remain the dominant error category. This outcome is expected and consistent with prior Text-to-SQL research. Crucially, the dominance of this category does not imply that schema ambiguity is inconsequential. Instead, it indicates that schema ambiguity primarily affects semantic correctness rather than syntactic validity or executability, especially for strong frontier models. As schema names become less explicit, models are more likely to generate logically plausible but subtly incorrect queries, rather than invalid SQL.

Second-Layer Errors. A closer inspection of execution error subtypes – illustrated in Table 5 – reveals further differences in how the two models respond to schema ambiguity.

For GPT-5.2, execution errors are both rare and heterogeneous. When they occur, they are primarily split between unknown execution errors, schema column errors, and other runtime issues. Notably, schema column errors remain a minority across all ambiguity levels and do not exhibit a clear increasing trend with ambiguity. This reinforces the conclusion that GPT-5.2 rarely fails due to incorrect schema linking, even when table and column names are heavily obfuscated. Instead, the few execution errors that occur, appear arbitrary and are not strongly tied to schema ambiguity.

Table 5. Execution Error Distribution by Type, Model and Level.

Dataset	Error	L0	L1	L2	L3
GPT-5.2	Table	0.00	0.00	0.00	0.00
	Column	25.00	69.57	52.94	43.48
	Ambiguity	0.00	0.00	0.00	0.00
	Syntax	0.00	0.00	0.00	0.00
	Dialect	0.00	0.00	0.00	0.00
	Other	75.00	30.43	47.06	56.52
Llama-3.3-70B	Table	1.32	0.92	2.12	0.00
	Column	38.96	37.33	39.68	34.85
	Ambiguity	0.00	0.46	0.53	1.01
	Syntax	16.88	13.36	6.88	10.61
	Dialect	1.35	0.46	0.00	0.00
	Other	41.56	47.46	50.79	53.54

For Llama-3.3-70B, execution error subtypes paint a more structured picture. Across all ambiguity levels, schema column errors and unknown execution errors together account for the majority of execution failures. Schema column errors consistently represent a large share – around one third – indicating systematic difficulties in column-level schema linking. Furthermore, unknown execution errors increase steadily with ambiguity, suggesting that obfuscated schema names amplify instability during query execution. Structural SQL errors, such as syntax errors, are present but secondary, and schema table errors and ambiguity-specific errors remain relatively rare.

These patterns show that schema ambiguity affects Llama-3.3-70B primarily by increasing schema-linking errors and execution instability, rather than by degrading high-level semantic reasoning alone. This behavior aligns with the monotonic accuracy degradation observed on Spider and BIRD-SQL.

8 Conclusion

This paper investigated the role of schema ambiguity in Text-to-SQL and introduced a controlled and reproducible framework for studying its impact. We presented a deterministic, heuristics-based schema transformation pipeline that generates realistic ambiguity through graded levels, enabling systematic difficulty scaling without compromising schema validity or benchmark compatibility. To quantify the resulting degree of ambiguity, we proposed the Schema Ambiguity Score, an embedding-based metric that captures the linguistic explicitness of schema object names and allows direct comparison across datasets and transformation levels.

Using this framework, we constructed multi-level variants of Spider, BIRD-SQL, and KaggleDBQA and evaluated two large language models under a unified zero-shot setting. The empirical results reveal that the effect of schema ambiguity is highly model- and dataset-dependent. While schema ambiguity increases SAS as intended, execution accuracy does not necessarily degrade sharply. In particular, a frontier model such as GPT-5.2 exhibits substantial robustness to schema obfuscation on clean benchmarks, whereas an open-source model shows more pronounced sensitivity, especially in the form of increased execution and schema-linking errors. On more challenging datasets such as BIRD-SQL, schema ambiguity plays a secondary role relative to other sources of difficulty, while on KaggleDBQA, strict execution accuracy is shown to conflate semantic correctness with syntactic minimality, obscuring meaningful differences in model behavior.

The error analysis further demonstrates that schema ambiguity rarely induces invalid SQL for strong models. Instead, its primary effect lies in subtle semantic mismatches rather than definite execution errors. This finding challenges the assumption that explicit schema naming is a prerequisite for reliable Text-to-SQL performance and suggests that modern LLMs can partially internalize schema abstraction beyond surface-level lexical cues.

Despite these insights, our study has limitations. The anonymization procedure focuses exclusively on linguistic ambiguity and does not address structural, value-level, or distributional sources of difficulty. SAS, while effective at capturing lexical ambiguity, does not measure contextual or task-specific relevance. Future work could extend this framework to additional ambiguity dimensions, incorporate human-grounded validation, and explore fine-tuning or other learning strategies that explicitly leverage controlled schema ambiguity. We hope that the proposed pipeline, metric, and benchmarks provide useful tools for advancing more realistic and diagnostic evaluation of Text-to-SQL systems.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Dou, L. et al.: MultiSpider: Towards Benchmarking Multilingual Text-to-SQL Semantic Parsing. In: The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23). pp. 12745–12753 (2023).
2. Eessaar, E.: On the Naming of Database Objects in the SQL Databases of Some Existing Software. CSOC 2023. 534–550 (2023).
3. Gao, Y. et al.: XiYan-SQL: A Multi-Generator Ensemble Framework for Text-to-SQL. Presented at the December (2024).
4. Guo, Z. et al.: Evaluating and Enhancing LLMs for Multi-Turn Text-to-SQL with Multiple Question Types. Presented at the December (2024).
5. Hazoom, M. et al.: Text-to-SQL in the Wild: A Naturally-Occurring Dataset Based on Stack Exchange Data. In: NLP4Prog 2021. pp. 77–78 (2021).
6. Kanburoğlu, A.B., Tek, F.B.: TUR2SQL: A Cross-Domain Turkish Dataset for Text-to-SQL. In: 8th International Conference on Computer Science and Engineering (UBMK) 2023. (2023).
7. Lee, C.-H. et al.: KaggleDBQA: Realistic Evaluation of Text-to-SQL Parsers. Presented at the June (2021).
8. Li, J. et al.: Can LLM Already Serve as a Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs. In: NeurIPS 2023. pp. 42330–42357 (2023).
9. Li, Y. et al.: Pay More Attention to History: A Context Modelling Strategy for Conversational Text-to-SQL. In: Interspeech 2022. pp. 2718–2722 (2022).
10. Li, Z. et al.: PET-SQL: A Prompt-Enhanced Two-Round Refinement of Text-to-SQL with Cross-Consistency. Presented at the June (2024).
11. Nguyen, A.T. et al.: A Pilot Study of Text-to-SQL Semantic Parsing for Vietnamese. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 4079–4085 (2020).
12. Papamichail, A. et al.: Do People Use Naming Conventions in SQL Programming? SOFSEM 2020: Theory and Practice of Computer Science. 429–440 (2020).
13. Pourreza, M. et al.: CHASE-SQL: Multi-Path Reasoning and Preference Optimized Candidate Selection in Text-to-SQL. Presented at the October (2024).
14. Pourreza, M., Rafiei, D.: Evaluating Cross-Domain Text-to-SQL Models and Benchmarks. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 1601–1611 (2023).
15. Wang, B. et al.: Know What I Don't Know: Handling Ambiguous and Unanswerable Questions for Text-to-SQL. In: Findings of the Association for Computational Linguistics: ACL 2023. pp. 5701–5714 (2023).
16. Wu, C.-K. et al.: I Need Help! Evaluating LLM's Ability to Ask for Users' Support: A Case Study on Text-to-SQL Generation. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 2191–2199 (2024).
17. Yu, T. et al.: CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases. In: Proceedings of the 2019 Conference on Empirical

Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 1962–1979 (2019).
18. Yu, T. et al.: SParC: Cross-Domain Semantic Parsing in Context. In: Association for Computational Linguistics. pp. 4511–4523 (2019).
19. Yu, T. et al.: Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In: EMNLP 2018. pp. 3911–3921 (2018).
20. Zhang, Y. et al.: Did You Ask a Good Question? A Cross-Domain Question Intention Classification Benchmark for Text-to-SQL. Presented at the (2020).
21. Zhong, R. et al.: Semantic Evaluation for Text-to-SQL with Distilled Test Suites. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 396–411 (2020).

Scaling Natural Language Database Interfaces: How Schema Growth Shapes Context Load and System Efficiency

Markus Cservenka[1][0009-0003-9997-8956]

[1] Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria
h11706187@s.wu.ac.at

Abstract. Natural language database interfaces rely heavily on Text-to-SQL models to translate user queries into executable database commands. However, these systems are typically assessed using benchmarks with small, static schemas, representing an unrealistic setting compared to practical database environments, where schemas expand continually as new tables and attributes are introduced. As a result, the impact of schema size on model performance and cost remains largely unclear. We address this gap by introducing a deterministic, structure-preserving schema scaling procedure that enlarges existing databases in Text-to-SQL benchmarks while keeping all original queries executable and semantically unchanged. Using scaled variants of Spider-dev with up to hundreds of tables, we evaluate two large language models under increasing schema-induced context pressure. In addition to execution accuracy, we introduce budget-normalized metrics that quantify token consumption and end-to-end runtime, enabling a more holistic evaluation of practical deployment costs. Our results demonstrate that, although execution accuracy degrades only moderately as schema size increases, computational cost grows sharply. We further evaluate lightweight schema filtering and find that it substantially reduces cost in clean schema expansions but can severely compromise accuracy when faced with realistic schema ambiguity. Overall, our findings underscore schema size as a crucial, yet underexamined dimension in evaluating and designing scalable natural language database interfaces.

Keywords: Text-to-SQL, Natural Language Database Interfaces, Practical Database Applications.

1 Introduction

The task of translating natural language questions into structured SQL queries – typically referred to as Text-to-SQL – has become a main concern when it comes to bridging human intent and relational databases access. In this task, a model must not only understand the semantic content of a user's question but also accurately map it to the underlying database

schema in order to generate a syntactically valid SQL statement that retrieves the desired information. Recently, Text-to-SQL research has experienced remarkable progress, enabled by domain-diverse benchmarks and increasingly powerful large language models (LLM).

Despite the positive developments in Text-to-SQL systems, these advances are largely based on evaluations on datasets with limited database schema sizes. For example, the widely used Spider benchmark [31] contains dozens of databases with up to around ten tables each and was designed to test compositional reasoning on multi-table joins and nested queries. However, it does not consider schema growth beyond a relatively modest size. Similarly established benchmarks such as BIRD-SQL [15] and LiveSQLBench [1] primarily focus on data volume and diverse SQL patterns rather than the scale of schema structure.

Consequently, even though real-world databases may exhibit schemas with up to hundreds of tables and thousands of columns due to iterative operational growth, existing evaluation practices do not systematically measure the impact of this kind of schema complexity on model performance. A database schema defines the blueprint of data organization – including tables, columns, relationships, and constraints – and serves as the structural foundation of database management and query execution. In production environments, schemas grow over time [3, 24] as new data sources, business requirements, and analytics use-cases are introduced, often resulting in increasing schema sizes far beyond those encountered in current benchmarks.

This structural growth can introduce context pressure that challenges modern LLMs. As the number of tables and columns increases, accurately identifying the correct schema elements presumably becomes harder, and the total input length for a model correspondingly grows.

In this paper, we systematically investigate the effects of schema size on Text-to-SQL performance. We suggest that the field's focus on datasets with fixed or moderately sized schemas leaves an important dimension of model behavior unexplored: performance under realistic schema expansion, independent of data volume or query complexity. Our study is guided by the following research questions:

- RQ1: How does increasing database schema size affect the performance of modern LLM-based Text-to-SQL systems, in terms of execution accuracy and practice-relevant efficiency metrics?
- RQ2: To what extent can low-overhead schema filtering methods mitigate schema-induced context pressure?

To answer these questions, we introduce a controlled procedure for synthetically scaling database schemas while preserving semantic structure and question applicability. Through this procedure, we generate a spectrum of schema-scaled variants of an existing benchmark and evaluate two state-of-the-art LLMs. In summary, this paper makes the following contributions:

1. We introduce a deterministic, embedding-based, structure-preserving schema scaling procedure that expands a base database into larger schema variants and generates domain-related synthetic database objects, supporting two difficulty levels.
2. We conduct a context pressure evaluation of two large language models across multiple scaled versions of the Spider-dev dataset – including original and variants with up to hundreds of tables – while isolating the schema size effects by maintaining identical questions and SQL targets across variants.
3. We propose and analyze budget-normalized performance metrics that reflect practical cost and efficiency trade-offs, beyond traditional execution accuracy.
4. We investigate low-overhead schema filtering methods as simple schema compression baselines and evaluate their effects in reducing context pressure.

The source code, data, and other artifacts have been made available at https://github.com/mcservenka/t2sql-schema-size-matters.

2 Related Work

Historically, Text-to-SQL has been framed as a semantic parsing problem grounded in relational databases, while facing core challenges, such as compositional reasoning, correct linking of schema elements and handling joins as well as nesting and aggregation [22, 26]. In recent years, large language models (LLMs) have shifted the dominant paradigm from task-specific parsers toward prompted or agent-based systems, often combining instruction following with external tools – including schema linking, execution feedback, self-correction [6, 7, 16, 21]. This trend has improved robustness and reduced the need for training data in new domains. However, these approaches typically rely on multi-step pipelines that achieve higher accuracy scores but require additional inference calls and longer prompts.

A key limitation in this area of research is the emphasis on execution accuracy during evaluation while diminishing the significance of efficiency [6, 18]. In practice, many agent-based approaches repeatedly preprocess schema metadata, retrieve auxiliary descriptions, or perform iterative refinement. These actions may substantially increase runtime and token usage, making them infeasible under real-world conditions. LiveSQLBench [1] represents one of the first benchmarks that addresses this issue by incorporating an average cost metric in their leaderboard. Furthermore, it highlights the inefficient behavior of agent-based designs and motivates researchers to develop alternatives that reduce context length and expensive actions. As a result, evaluation broadens to consider not only whether a method can solve a benchmark, but also how its cost scales with increasing design complexity or schema size.

Text-to-SQL benchmarks have gradually expanded along several dimensions of realism, including schema structures [9, 13], conversational interactions [29, 30], multilingual settings [4, 10, 19], and more diverse query intents [8, 25, 28, 32]. BIRD-SQL [15], for

example, was introduced to narrow the gap between academic evaluation and practical settings by emphasizing large database volume as well as diverse SQL coverage in comparison to small and light-weight databases [31, 34]. The aforementioned LiveSQLBench [1] follows this direction and represents a continuously evolving, contamination-aware benchmark consisting of databases of various sizes.

Nevertheless, the dominant evaluation setting – along with Spider-style Text-to-SQL – remains centered on schemas that are generally manageable for direct serialization into a single LLM prompt. More recent efforts such as Spider 2.0 [13] move toward enterprise-like workflows and agent-based interaction and reveal that existing systems can struggle when realistic workflows and scales are introduced. However, even in these newer benchmarks, realism often comes from various sources of complexity – including tool use, external documents, multi-step tasks – rather than a controlled study of schema growth itself.

Our study complements these benchmark directions by isolating a specific and practically motivated point of view: structural schema scaling, independent of data volume, domain shift, or conversational context. This is particularly relevant because schema growth is a common reality in operational databases [3, 24], yet its implications on SQL parsing remain unclear, as existing benchmarks do not support systematic evaluation. A substantial share of modern Text-to-SQL advances is driven by schema linking [7, 16]. This step – typically performed during preprocessing – aims to identify the tables and columns that are relevant to the input question. Although accuracy optimization remains the central focus of current research, schema linking also serves the essential function of reducing context size by excluding irrelevant portions of the database schema. Consequently, it also plays a central role in schema-filtering methods used for large catalogs or multi-database environments. Representative approaches include methods that explicitly decouple linking from generation [14], retrieval-augmented schema linking for massive schemas [5] and systems designed for scalable real-world schema linking across large multi-database environments [27]. Robust schema-linking variants further incorporate multi-stage selection and self-correction to improve recall under noise [2]. Recent work also targets prompt compaction for wide schemas via structured filtering and reranking [17, 20].

While these methods often improve end-task performance, they generally also increase real-world cost by introducing additional retrieval stages, further LLM calls, or iterative refinement loops. Moreover, schema filtering introduces a unique failure mode in which the correct SQL query cannot be produced if a necessary table is removed during filtering. Summarizing, existing research has made major progress on LLM-based Text-to-SQL including real-world resembling benchmarks and schema-linking pipelines. However, two gaps remain insufficiently addressed:

- Lack of controlled large-schema evaluation: Modern benchmarks increasingly incorporate realism through harder queries, larger data contents, or workflow complexity, but do not provide a controlled mechanism to vary schema size while keeping question-query pairs fixed.

- Missing efficiency-centric evaluation: Reporting is typically dominated by execution accuracy, whereas practical deployment requirements, including token usage and runtime under constrained contexts, are frequently ignored.

These gaps motivate our approach. We propose a deterministic, structure-preserving schema scaling procedure that enables controlled context-pressure evaluation, paired with budget-normalized metrics. Furthermore, we evaluate low-overhead schema filtering methods as deployable baselines and measure the benefit of reducing schema-size-induced context pressure while assessing the risk of making instances unanswerable by removing required tables. However, in contrast to most prior schema-linking literature, our goal is not to propose a new linking model.

3 Schema Scaling Procedure

In this section we introduce a deterministic schema scaling procedure that enlarges an existing relational database schema while preserving its original semantics and query answerability. Starting from a single base schema, the procedure produces scaled variants with substantially more tables and columns, enabling controlled evaluation of schema-induced context pressure in Text-to-SQL systems.

Crucially, the procedure treats the original database as immutable. All original tables, columns, primary keys, foreign keys, and data types remain unchanged. As a result, every gold SQL query from the original benchmark executes without modification on the scaled schema and returns identical results. Schema scaling is achieved exclusively by adding synthetic tables and columns that do not interfere with existing query paths. To balance experimental control and realism, the scaler operates at two levels of difficulty. Level 1 isolates the effect of schema size by introducing semantically fitting but irrelevant schema objects, while Level 2 introduces controlled ambiguity that reflects realistic schema evolution in production systems.

3.1 Design Principles

The schema scaler is guided by the following design principles, which apply across all datasets, sizes, and levels:

- Original schema preservation: The original schema is left entirely untouched. Consequently, we do not rename or modify existing tables, columns, or relational constraints.
- No semantic collisions: Newly generated names are required to be domain-related but never identical or synonymous with the original table or column names.

- Deterministic generation: Schema generation is fully deterministic. Given the original database schema, the predefined level as well as a seed, the resulting scaled schema is identical across runs.
- Preserved query answerability: All gold queries execute consistently and return identical results on the scaled schema as on the original database.

These invariants ensure that any observed changes in model behavior can be attributed to schema size and structure rather than to semantic obscurity or query invalidation.

3.2 Candidate Name Selection

To generate a set of candidate names for the schema scaler, we first determine the lexical footprint of the original schema. All table names and column names are collected and embedded using a fixed embedding model For our experiments, we employ fastText's English Common Crawl. The resulting embedding space defines a semantic region that captures the domain of the database.

Some schemas – particularly in benchmarks such as Spider – contain only a small number of schema tokens, which can lead to an unstable or overly narrow semantic representation. To mitigate this effect, we optionally expand the schema vocabulary with a small set of anchor words. These anchor words are used to expand the semantic scope of the database without introducing unrelated domains. Specifically, we deterministically select k_{anchor} semantically related nouns from a global noun vocabulary based on similarity to the centroid of the original schema embeddings. These anchor words are used solely to stabilize the embedding space. They are not part of the original schema and do not introduce ambiguity with existing schema objects.

A final schema centroid is then computed from the embeddings of the original schema tokens and, if applicable, the anchor words. Using this centroid, we construct a candidate pool of table and column names from the global vocabulary. Each candidate word must adhere to several constraints: it must not already appear in the schema, must not be a SQL keyword and is not allowed to be overly similar to any original schema object names. To satisfy this requirement, candidate names are constrained to a bounded similarity interval relative to the schema centroid – ranging between sim_{min} and sim_{max} – ensuring that generated names are thematically related to the domain without being near-synonyms of existing tables or columns. For very small schemas, these similarity bounds are deterministically relaxed until a sufficient candidate pool is obtained.

The resulting vocabulary provides a large, domain-adjacent name space from which synthetic schema objects can be generated while maintaining strict control over lexical ambiguity.

3.3 Level 1: Simple Schema Inflation

On Level 1 the schema scaler performs pure schema inflation without introducing ambiguity relative to the original schema. Its purpose is to isolate the effect of schema size and increased context length. New tables are generated up to a predefined target size and subsequently partitioned into three categories with fixed proportions across all datasets:

- Entity tables, which resemble standard domain entities with a single primary key, several descriptive columns, and realistic data types.
- Join tables, which model many-to-many relationships using composite primary keys and foreign keys, mirroring common relational design patterns.
- Metadata tables, which simulate auxiliary subsystems such as logs, caches, or configuration records. These tables primarily serve to increase schema size and context length and are only weakly connected to the rest of the schema.

Column generation follows deterministic templates. Each table receives a primary key column with an integer type, and additional columns are selected from fixed attribute patterns, such as names, descriptions, counts, timestamps. This ensures internal consistency and avoids implausible or degenerate schema parts. The number of columns added to each synthetic table is determined pseudo-randomly within a predefined range, using a fixed seed to ensure full reproducibility. Column names are unique within each table and do not collide with column names in the original schema.

Foreign key integration is handled conservatively. Only a small, fixed number of new tables are allowed to reference original tables via foreign keys, preventing unrealistic patterns. The majority of foreign key relationships are created among synthetic tables, producing realistic relational subgraphs without interfering with existing query paths.

Importantly, Level 1 never reuses original table names or column names in synthetic tables, and it avoids near-synonyms of original schema objects. As a result, the original schema remains lexically and structurally dominant, and new tables are largely irrelevant to the semantics of existing queries. Thus, Level 1 isolates the impact of schema size and context pressure without introducing structural or linguistic schema ambiguity.

3.4 Level 2: Structural Schema Inflation

Level 2 introduces additional mechanisms that reflect how real-world schemas tend to evolve over time.

Original-Inspired Tables. To model schema redundancy, Level 2 allows the creation of original-inspired synthetic tables. With a probability of p_{orig}, a newly generated entity table is named by combining an original table name with the selected candidate word (e.g., singer_notes, concert_archive). These tables reuse a subset of columns from the corresponding original table, while always introducing a newly named primary key to preserve schema integrity. The original table itself remains unchanged.

This mechanism models common production patterns such as denormalization, shadow tables, and partial replicas, where multiple tables represent overlapping aspects of the same real-world entity. Naturally, these tables are schema-only and contain no data, ensuring that gold query results remain unchanged while introducing realistic schema grounding ambiguity.

Join Competition. Level 2 further introduces controlled join competition. For selected original foreign key relationships, the scaler deterministically generates a single synthetic bridge table that provides an alternative, foreign-key-consistent join path between the same pair of tables. For instance, two original tables, where *employee* references *department*, can be used to create a bridge table, *employee_department_bridge,* which in turn references both original tables. The original join as well as the alternative join path are both valid and executable, but only the original reflects the intended schema semantics and leads to a correct execution result. This construction introduces structural ambiguity without modifying the original schema or invalidating existing queries. It models realistic enterprise artifacts such as legacy association tables, staging intermediates, or partially redundant relationship encodings.

3.5 Purpose

Across both levels, determinism is enforced by fixed seeds and ordered vocabularies. Given the same original schema, target size, level, and seed, the procedure always produces the same enlarged schema. Level 1 yields large but semantically clean schemas that primarily increase input length and computational cost. Level 2 further introduces realistic redundancy and structural ambiguity that naturally emerge as schemas evolve. In both cases, the original database remains unchanged, and all original Text-to-SQL queries remain executable with identical results.

Together, the two-level functionality of the schema scaler enables controlled evaluation of context pressure, schema linking, join reasoning and cost–accuracy trade-offs, hence revealing effects that are not captured by aggregate execution accuracy alone.

4 Experimental Setup

In the following, we provide a comprehensive overview of our experimental setup, covering the models, data, prompting strategies, and evaluation criteria.

4.1 Models

We evaluate two representative large language models to cover both closed- and open-source settings. GPT-5.2 is accessed via the official OpenAI API, and Llama-3.3-70B is

accessed via the TogetherAI API. This choice allows us to study schema-scaling effects across distinct model families and deployment regimes while keeping the evaluation protocol identical.

4.2 Datasets

All experiments are performed on Spider-dev – which includes 20 databases and 1,034 question-query pairs – as well as on its schema-scaled variants. On average, each database in the original version of spider-dev consists of 4 tables with 5.5 columns each. Within the schema scaling procedure we set $k_{anchor} = 30$ and use similarity bound of $sim_{min} = 0.55$ and $sim_{max} = 0.25$. For each database, synthetic tables are added in proportions of 60% entity tables, 25% join tables, and 15% metadata tables. Additionally, we instruct the schema scaler to generate up to five foreign key references to original tables in Level 1 and to form original-inspired entity tables with $p_{orig} = 0.4$ in Level 2.

We apply the schema scaler using the abovementioned settings on spider-dev and yield level-specific dataset variants with schema sizes of 50, 100, 250, 500, and 800 tables, resulting in 10 additional dataset versions in total. We intentionally focus on Spider-dev as the foundational dataset to isolate the effect of schema size from other known sources of difficulty. Spider's relatively clear questions and compact original schemas enable strong zero-shot baselines, making it well suited for controlled analysis of schema scale. Incorporating datasets such as BIRD-SQL [15] or KaggleDBQA [12] would introduce additional confounding factors – such as higher linguistic ambiguity, domain complexity, or data-volume effects – which is not the focus of this study.

4.3 Prompting

All experiments are performed under a zero-shot setting, with models receiving no task-specific fine-tuning, in-context examples, or dataset-tailored adaptation. This design ensures that performance differences reflect challenges and ambiguities introduced by schema size, rather than memorization effects or reliance on example-driven reasoning. We standardize the prompting procedure across models and dataset variants and use a schema serialization format inspired by M-Schema [7], providing a structured, hierarchical encoding of tables and columns. Each model request is structured equally. We first pass two system messages – a general task instruction as well as the serialized schema representation – and eventually add one user message, which is the natural language question.

To enforce consistent and machine-readable outputs, we constrain the model using explicit tool-function directives, requiring responses to be a single dictionary containing one field: {"sql": "..."}. This standardization removes variability in output formatting, enabling consistent downstream execution and evaluation. Any prediction that deviates from this prescribed format is treated as *Null*.

4.4 Evaluation Metrics

Traditionally, we evaluate the correctness of the predicted SQL-query using Execution Accuracy (ExA), computed with Spider's official distilled test suite [33]. In addition, we report two efficiency metrics that capture practical deployment costs:

- Token Efficiency (TE): measures the number of total tokens used in an LLM per query.
- Latency Efficiency (LE): measures the runtime in seconds per query.

Inspired by [6], we quantify cost as the total number of tokens – including both prompt and output tokens – consumed by an LLM during generation. This metric is consistent across open- and closed-source models and implicitly captures any LLM-driven schema-linking steps. Although all models are prompted with identical text and schema representations, reported token counts differ across models due to differences in tokenizer design and internal prompt serialization. We report token usage as measured by each model's API, as this reflects the actual cost accumulated during deployment and is therefore the most appropriate efficiency metric. While providers such as OpenAI and TogetherAI charge based on usage, we report token counts as a model-agnostic cost proxy.

Latency is measured as the end-to-end request time for LLM inference. When schema filtering is applied, we additionally measure the filtering time – including question embedding and schema slicing over pre-embedded schemas. Other computations such as schema serialization, loading of embedding models, or formatting are excluded, as they can be in practical deployment scenarios. We are aware that runtime can fluctuate due to infrastructure variability, background services, and other environmental factors. However, these measurements remain informative, as they capture the relative overhead of the most time-intensive stages in the pipeline [27].

4.5 Schema Filtering Baselines

In order to study whether reducing schema context can reduce schema-induced context pressure, we evaluate two non-LLM-based schema filtering baselines: BM25-based lexical retrieval [23] and dense embedding–based retrieval [11]. We intentionally restrict ourselves to lightweight, pre-LLM filtering methods to avoid introducing additional language model calls, which would increase token consumption and latency. These approaches are consistent with the evaluation metrics introduced in section 4.4, thereby contributing positively to our efficiency analysis.

The BM25 filter performs lexical matching between the natural language question and table-level schema representations. Each table is converted into a bag-of-tokens document derived from its name, column names, data types, and foreign key metadata. Given a question, BM25 scores all tables and retains the top $k_{top} = 10$ tables. This method introduces

negligible computational overhead and relies solely on fixed BM25 statistics, with no additional learned components.

The dense filter embeds both the question and table-level schema representations using a sentence embedding model, for which we use *all-MiniLM-L6-v2* in our experiments. Table embeddings are computed once and cached. At inference time, the question embedding is compared against all table embeddings using cosine similarity, and again the $k_{top} = 10$ tables are retained. This approach introduces a moderate but restricted amount of overhead that is independent of LLM inference and does not scale with prompt length.

Both filters operate exclusively at the table level, meaning columns are unaffected and only tables not selected by the filter are removed from the schema. Foreign key constraints are pruned accordingly to maintain schema consistency. This design aligns with our schema scaling procedure, which primarily increases the number of tables while keeping column distributions stable, and allows us to isolate the effect of table-level schema reduction.

5 Results

In the following, we present our experimental results, analyzing the effects of schema size on Text-to-SQL performance as well as the impact of schema filtering under varying levels of schema complexity.

5.1 Schema Scaling Effects

Aggregate Analysis. We first examine the comprehensive effects of schema scaling on the overall performance across all Spider-dev variants, models, and difficulty levels. Table 1 and Table 2 summarize the results of Execution Accuracy (ExA), Token Efficiency (TE) and Latency Efficiency (LE) for GPT-5.2 and Llama-3.3-70B, respectively.

Across both models, ExA declines slowly and non-monotonically as schema size increases. For GPT-5.2, ExA slightly decreases from 76.79% on the original (O) Spider-dev schema to 73.98% on the 800-variant in Level 1 and to 73.21% in Level 2. Llama-3.3-70B exhibits a similar pattern, with ExA declining from 76.21% to 70.21% and 71.86% at the largest schema size. Importantly, even under extreme schema inflation both models retain a substantial fraction of their baseline accuracy.

This relative robustness suggests that aggregate execution accuracy alone understates the practical impact of schema growth. In particular, the modest ExA drop may give the impression that large schemas pose little challenge to modern LLMs, an interpretation that we show to be incomplete once efficiency metrics are considered.

In contrast to ExA, efficiency metrics exhibit a strong and nearly linear dependence on schema size. Token consumption per query grows dramatically as schemas expand. For GPT-5.2 in Level 1, the number of tokens per query increases from approximately 615 on the original schema to over 56,000 on the 800-variant, representing a two-order-of-

magnitude increase. Llama-3.3-70B follows a nearly identical scaling trend. Level 2 schemas consistently show slightly lower token counts at all sizes, reflecting small differences in schema serialization length, but the overall growth pattern remains unchanged. Runtime exhibits a similar, though less steep, trend. Average latency per query roughly doubles for GPT-5.2 between the smallest and largest schemas, and increases by a comparable factor for Llama-3.3-70B. Fig. 1 illustrates ExA and token consumption per query as a function of schema size.

Table 1. GPT-5.2 Performance across Variants.

Level	Variant	ExA	TE	LE
Original	spider-dev-O	76.79	614	1.21
Level 1	spider-dev-50	74.47	3,898	1.45
	spider-dev-100	74.08	7,344	1.42
	spider-dev-250	75.15	17,759	1.57
	spider-dev-500	74.47	35,527	1.88
	spider-dev-800	73.98	57,135	2.28
Level 2	spider-dev-50	73.60	3,818	1.55
	spider-dev-100	73.79	7,016	1.69
	spider-dev-250	74.76	16,834	1.88
	spider-dev-500	73.89	33,454	1.93
	spider-dev-800	73.21	53,317	2.21

Table 2. Llama-3.3-70B Performance across Variants.

Level	Variant	ExA	TE	LE
Original	spider-dev-O	76.21	656	1.40
Level 1	spider-dev-50	76.89	3,890	0.96
	spider-dev-100	74.56	7,287	2.39
	spider-dev-250	72.44	17,596	2.84
	spider-dev-500	70.50	35,119	3.43
	spider-dev-800	70.21	56,472	2.19
Level 2	spider-dev-50	73.02	3,828	0.94
	spider-dev-100	72.92	6,998	0.93
	spider-dev-250	74.66	16,744	1.51
	spider-dev-500	73.50	33,208	1.82
	spider-dev-800	71.86	52,853	2.74

When comparing Level 1 and Level 2, we observe that schema ambiguity has a limited effect on overall ExA and does not mitigate efficiency degradation. In some mid-range schema sizes, Level 2 even slightly outperforms Level 1 in ExA. However, these differences are small and inconsistent. In contrast, the increase in tokens and runtime per query is largely dictated by schema size rather than ambiguity level, reinforcing that context pressure is primarily a function of input length, not semantic difficulty alone.

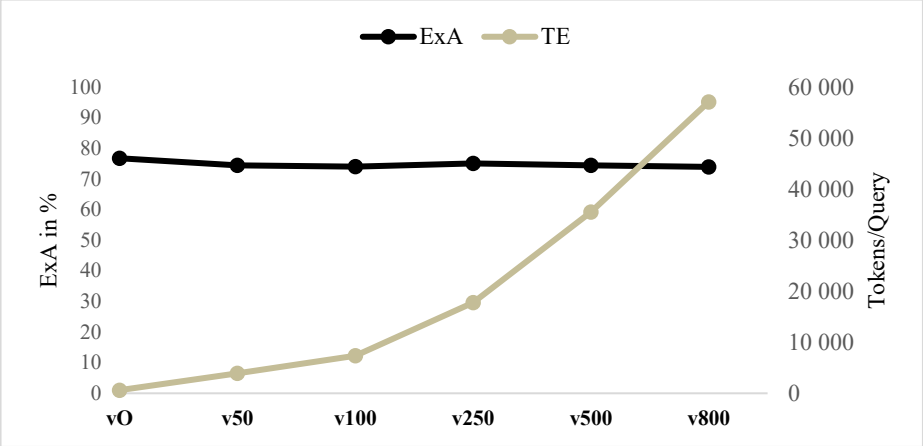


Fig. 1. Execution Accuracy vs. Token Efficiency of GPT-5.2 on Level 1 by Variant.

Overall, these results reveal a key asymmetry: while execution accuracy remains relatively stable under aggressive schema scaling, computational cost and latency grow substantially. Although expected, this divergence highlights the importance of evaluating Text-to-SQL systems beyond ExA, especially in realistic settings where large schemas are common.

In the following sections, we analyze this phenomenon in greater detail by focusing on join-inducing samples and schema grounding behavior, which expose failure modes that aggregate accuracy metrics fail to capture.

Table Joining Errors. Aggregate execution accuracy masks substantial heterogeneity across query types. To better understand the mechanisms through which schema scaling impacts model behavior, we restrict our analysis to join-inducing samples, defined as those samples whose gold SQL includes at least one JOIN clause.

This subset comprises 408 of the 1,034 Spider-dev queries and is inherently more challenging than the full set, as reflected by lower baseline accuracy on the original schema for both models. Table 3 summarizes ExA across all scenarios. For GPT-5.2, execution accuracy on join-queries remains remarkably stable as schema size increases. In Level 1, Join-ExA fluctuates within a narrow band, decreasing only slightly from 66.42% on the original

schema to 64.95% on the 800-variant. A similar pattern can be found in Level 2, where Join-ExA ranges between 61.76% and 64.71% across all scaled variants. These results indicate that, for GPT-5.2, the additional context introduced by large schemas does not substantially impair join reasoning once the correct tables are identified.

In contrast, Llama-3.3-70B exhibits a noticeable degradation in join performance as schema size increases. In Level 1, Join-ExA drops from 67.65% on the original schema to 53.19% on the 800-variant, a reduction of more than 14 percentage points. Level 2 shows a similar downward trend, with Join-ExA declining from 60.54% on spider-dev-50 to 54.41% on spider-dev-800. This decrease is substantially steeper than what is observed in overall ExA, highlighting that Join-relevant samples are disproportionately affected by schema scale for this model.

Table 3. Execution Accuracy by Variant, Model and Level on the JOIN-subset.

ExA (JOIN)	**GPT-5.2**		**Llama-3.3-70B**	
Variant	**L1**	**L2**	**L1**	**L2**
spider-dev-O	66.42		67.65	
spider-dev-50	65.93	61.76	65.2	60.54
spider-dev-100	63.24	62.75	61.76	61.03
spider-dev-250	64.22	64.71	57.60	62.50
spider-dev-500	63.97	63.48	54.66	58.82
spider-dev-800	64.95	62.99	53.19	54.41

These findings reveal that schema-induced context pressure interacts strongly with join reasoning, but in a model-dependent manner. While GPT-5.2 maintains stable ExA scores under extreme schema scaling, Llama-3.3-70B struggles to reliably identify and compose correct join-paths as the number of competing tables grows. This divergence suggests that overall execution accuracy can mask important failure modes and is inherently sensitive to the specific question-query-pairs present in the dataset. Even when aggregate ExA declines only modestly, performance on structurally complex queries may degrade substantially.

Synthetic Table Errors. To further understand how schema size related ambiguity affects model behavior, we analyze synthetic table usage, defined as cases where a predicted SQL query references at least one synthetic table introduced by the schema scaler. Since every extracted sample in this group yields an incorrect final prediction, the analysis shifts focus from end-task correctness to examining schema grounding behavior.

As expected, Level 1 exhibits almost no synthetic table usage for either model. Since synthetic tables on Level 1 use lexically distinct names and do not overlap with original schema objects, both GPT-5.2 and Llama-3.3-70B rarely reference them. Across all schema sizes, the number of predictions involving synthetic tables remains near zero, with only

sporadic occurrences confirming that Level 1 successfully isolates schema size effects without introducing grounding ambiguity.

In contrast, Level 2 shows substantially higher rates of synthetic table usage, reflecting the intended introduction of realistic ambiguity via original-inspired table names and competing join paths. For smaller schema sizes, both models frequently reference synthetic tables, with counts reaching up to 27 for GPT-5.2 and over 40 for Llama-3.3-70B. This behavior indicates that models are often drawn to schema objects that partially resemble original tables or provide alternative join paths, even though these tables are semantically irrelevant to the gold queries.

Interestingly, synthetic table usage decreases as schema size increases. At larger schema sizes – such as spider-dev-500 and spider-dev-800 – the number of synthetic references drops to single-digit counts. This trend is counterintuitive, as one might expect larger schemas to increase the likelihood of grounding errors. A plausible explanation is that, as schemas scale, models rely more on consistently relevant original elements and progressively ignore larger portions of the expanded schema. We leave a deeper investigation of this phenomenon to future work.

As stated, we do not emphasize ExA on the subset of synthetic-table predictions. In nearly all cases, referencing synthetic tables leads to incorrect results, as these tables contain no data and are not part of the intended scope of query. Rare exceptions occur when both the gold query and the predicted query produce identical sets of empty outputs – meaning the same set of columns with Null-values only. This results in an execution match despite incorrect schema grounding. Since such cases are artifacts of execution-based evaluation rather than meaningful successes, we treat synthetic table usage primarily as a diagnostic signal of schema grounding failure, independent of ExA.

This analysis confirms that Level 2 introduces realistic schema grounding challenges that are largely absent in Level 1. Moreover, it highlights that grounding errors can occur even when aggregate execution accuracy remains relatively stable, reinforcing the need for complementary analyses beyond ExA. These observations also motivate the schema filtering study in the next section, where we examine whether reducing schema context size can reduce these kinds of linking errors.

5.2 Schema Filtering Effects

In this section we investigate the extent to which lightweight schema filtering can reduce schema-induced context pressure and how this behavior is affected by realistic schema ambiguity. As defined in section 4.5, we evaluate BM25 and dense table retrieval with a fixed table set of $k_{top} = 10$ tables. This choice is motivated by the observation that most Spider gold queries reference no more than 3-5 tables. A systematic study of k_{top} extends beyond the scope of this paper and is therefore left for future work.

Table 4 and Table 5 report execution accuracy as well as the efficiency metrics for the variants spider-dev-100 and spider-dev-800, both levels and models for BM25- and dense-

filtering, respectively. The selected dataset sizes are representative of moderate and extreme scaling.

Across both models and both filtering methods, schema filtering effectively decouples prompt cost and latency from schema size. Token usage drops from over 50,000 without filtering to approximately 1,000 tokens per query, and remains nearly constant when increasing the schema from 100 to 800 tables. This trend was expected, since k_{top} naturally determines and limits context size. Runtime exhibits a similarly stable profile, indicating that pre-LLM filtering removes the dominant source of scaling cost without introducing substantial overhead. These gains are consistent for GPT-5.2 and Llama-3.3-70B and hold for both BM25 and dense retrieval.

Table 4. BM25-Filtering Performance.

Variant	LLM	L	ExA	TE	LE
spider-dev-100	GPT-5.2	L1	70.89	1,045	1.31
		L2	54.84	875	2.11
	Llama-3.3-70B	L1	71.57	1,079	0.84
		L2	54.45	915	0.91
spider-dev-800	GPT-5.2	L1	71.08	1,041	1.76
		L2	27.66	852	2.05
	Llama-3.3-70B	L1	70.70	1,075	1.02
		L2	29.98	877	0.72

Table 5. Dense-Filtering Performance.

Variant	LLM	L	ExA	TE	LE
spider-dev-100	GPT-5.2	L1	73.21	1,005	1.91
		L2	49.71	815	2.01
	Llama-3.3-70B	L1	73.02	1,041	0.81
		L2	45.84	855	0.82
spider-dev-800	GPT-5.2	L1	71.08	944	1.87
		L2	22.53	812	2.06
	Llama-3.3-70B	L1	70.50	979	0.87
		L2	22.82	832	0.84

In case of Level 1, where schema inflation introduces no lexical overlap or structural ambiguity, execution accuracy remains largely stable across schema sizes and models. With BM25 filtering, ExA stays around 71% for both GPT-5.2 and Llama-3.3-70B on spider-

dev-100 and spider-dev-800. Dense retrieval results unveil comparable or slightly higher accuracy on spider-dev-100 and show only minor degradation on the 800-variant. These results demonstrate that large schemas alone do not harm Text-to-SQL performance when relevant tables are preserved by the filter, even under aggressive schema scaling.

In contrast, when schema inflation introduces name-overlapping and structurally similar distractor tables – as Level 2 does – execution accuracy degrades considerably and seems to worsen with schema size, despite identical token budgets and nearly constant latency. For GPT-5.2, BM25 filtering reduces ExA from 54.84% on the 100-variant to 27.66% on the 800-variant. Dense filtering drops even further, from 49.71% to 22.53%. Llama-3.3-70B exhibits the same pattern, with ExA falling below 30% on spider-dev-800 for both filters.

Notably, the decline is more pronounced for dense retrieval than for BM25. This suggests that semantic similarity, while being beneficial under clean conditions, increases ambiguity when distractor tables are intentionally designed to resemble original schema elements. In such cases, dense retrieval is more likely to select semantically plausible but incorrect tables, making correct query generation impossible once relevant tables are filtered out.

These results reveal a fundamental accuracy-efficiency trade-off. Low-overhead schema filtering is highly effective and safe under clean schema expansion, dramatically reducing cost without significantly sacrificing accuracy. However, when facing realistic schema ambiguity, aggressive filtering can eliminate necessary schema components and consequently lead to severe accuracy loss that stronger LLM reasoning alone cannot compensate. This exposes a fundamental limitation of efficiency-driven schema compression methods and emphasizes the importance of economical, schema-aware filtering strategies that explicitly anticipate ambiguity, particularly in large and evolving schemas.

6 Conclusion

This paper studied the impact of database schema size on Text-to-SQL performance, an aspect that is largely overlooked by existing benchmarks and evaluation protocols. We introduced a deterministic, structure-preserving schema scaling procedure that enables controlled enlargement of existing databases while keeping all original queries executable and semantically unchanged. Based on that method, we proposed a context-pressure evaluation framework that jointly considers execution accuracy and practice-relevant efficiency metrics. Furthermore, we examined lightweight schema filtering as a cost-oriented mitigation strategy.

Our empirical results reveal a clear asymmetry. While execution accuracy degrades only moderately as schema size increases – suggesting apparent robustness of modern LLMs – token consumption and latency grow dramatically, hence exposing substantial costs that are invisible to accuracy-only evaluation. Closer examination indicates that this robustness does not apply consistently across all cases. Join-inducing samples are significantly more

sensitive to schema size. In particular, open-source models exhibit pronounced grounding errors when faced with realistic schema growth that includes overlapping names and redundant structures. These effects are further amplified under schema filtering, where efficiency gains remain large but accuracy collapses under realistic ambiguity, revealing a fundamental accuracy-efficiency trade-off.

Together, these findings indicate that large schemas primarily stress Text-to-SQL systems through context pressure and schema grounding, rather than through query complexity alone. They also show that efficiency-oriented schema compression is viable under clean schema expansion, but fragile when schemas evolve in realistic, redundant ways.

Despite its contributions, this work has certain limitations. Our study focuses on a single benchmark and fixed filtering budgets, and does not explore adaptive retrieval, column-level filtering, or LLM-based schema linking. Furthermore, the counterintuitive decline in grounding errors at the largest schema sizes requires additional investigation. Future work should extend schema scaling to additional datasets, study ambiguity-robust filtering strategies, and develop evaluation protocols that jointly optimize correctness, cost, and robustness under schema evolution.

Overall, we argue that schema size must be treated as a significant evaluation dimension for Text-to-SQL systems. Ignoring it risks overestimating model readiness for real-world deployment, where schema growth is the norm rather than the exception. Beyond accuracy, efficiency metrics are crucial for assessing real-world usability, as deployment feasibility is inherently constrained by cost and response time.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. BIRD Team: LiveSQLBench: A Dynamic and Contamination-Free Benchmark for Evaluating LLMs on Real-World Text-to-SQL Tasks, https://github.com/bird-bench/livesqlbench, (2025).
2. Cao, Z. et al.: RSL-SQL: Robust Schema Linking in Text-to-SQL Generation, https://doi.org/10.48550/arXiv.2411.00073, (2024). https://doi.org/10.48550/arXiv.2411.00073.
3. Cleve, A. et al.: Understanding database schema evolution: A case study. Science of Computer Programming. 97, 113–121 (2015).
4. Dou, L. et al.: MultiSpider: Towards Benchmarking Multilingual Text-to-SQL Semantic Parsing. In: The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23). pp. 12745–12753 (2023).
5. Eden, J. et al.: RASL: Retrieval Augmented Schema Linking for Massive. In: KDD Workshop on Structured Knowledge for Large Language. (2025).

6. Gao, D. et al.: Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. Proceedings of the VLDB Endowment. 1132–1145 (2024).
7. Gao, Y. et al.: XiYan-SQL: A Multi-Generator Ensemble Framework for Text-to-SQL. Presented at the December (2024).
8. Guo, Z. et al.: Evaluating and Enhancing LLMs for Multi-Turn Text-to-SQL with Multiple Question Types. Presented at the December (2024).
9. Hazoom, M. et al.: Text-to-SQL in the Wild: A Naturally-Occurring Dataset Based on Stack Exchange Data. In: NLP4Prog 2021. pp. 77–78 (2021).
10. Kanburoğlu, A.B., Tek, F.B.: TUR2SQL: A Cross-Domain Turkish Dataset for Text-to-SQL. In: 8th International Conference on Computer Science and Engineering (UBMK) 2023. (2023).
11. Karpukhin, V. et al.: Dense Passage Retrieval for Open-Domain Question Answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 6769–6781 (2020).
12. Lee, C.-H. et al.: KaggleDBQA: Realistic Evaluation of Text-to-SQL Parsers. Presented at the June (2021).
13. Lei, F. et al.: Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. Presented at the November (2024).
14. Li, H. et al.: RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL. In: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence. pp. 13067–13075 (2023).
15. Li, J. et al.: Can LLM Already Serve as a Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs. In: NeurIPS 2023. pp. 42330–42357 (2023).
16. Li, Z. et al.: PET-SQL: A Prompt-Enhanced Two-Round Refinement of Text-to-SQL with Cross-Consistency. Presented at the June (2024).
17. Liu, G. et al.: Solid-SQL: Enhanced Schema-linking based In-context Learning for Robust Text-to-SQL. In: Proceedings of the 31st International Conference on Computational Linguistics. pp. 9793–9803 (2025).
18. Mitsopoulou, A., Koutrika, G.: Analysis of Text-to-SQL Benchmarks: Limitations, Challenges and Opportunities. Presented at the February (2025).
19. Nguyen, A.T. et al.: A Pilot Study of Text-to-SQL Semantic Parsing for Vietnamese. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 4079–4085 (2020).
20. Nguyen, T. et al.: Scaling Text2SQL via LLM-efficient Schema Filtering with Functional Dependency Graph Rerankers, https://doi.org/10.48550/arXiv.2512.16083, (2025). https://doi.org/10.48550/arXiv.2512.16083.
21. Pourreza, M. et al.: CHASE-SQL: Multi-Path Reasoning and Preference Optimized Candidate Selection in Text-to-SQL. Presented at the October (2024).
22. Qi, J. et al.: RASAT: Integrating Relational Structures into Pretrained Seq2Seq Model for Text-to-SQL. Presented at the October (2022).
23. Robertson, S., Zaragoza, H.: The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval. 3, 4, 333–389 (2009).

24. Skoulis, I. et al.: Growing up with stability: How open-source relational databases evolve. Information Systems. 53, 363–385 (2015).
25. Wang, B. et al.: Know What I Don't Know: Handling Ambiguous and Unanswerable Questions for Text-to-SQL. In: Findings of the Association for Computational Linguistics: ACL 2023. pp. 5701–5714 (2023).
26. Wang, B. et al.: RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 7567–7578 (2020).
27. Wang, Y. et al.: LinkAlign: Scalable Schema Linking for Real-World Large-Scale Multi-Database Text-to-SQL. Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing. 978–992 (2025).
28. Wu, C.-K. et al.: I Need Help! Evaluating LLM's Ability to Ask for Users' Support: A Case Study on Text-to-SQL Generation. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 2191–2199 (2024).
29. Yu, T. et al.: CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 1962–1979 (2019).
30. Yu, T. et al.: SParC: Cross-Domain Semantic Parsing in Context. In: Association for Computational Linguistics. pp. 4511–4523 (2019).
31. Yu, T. et al.: Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In: EMNLP 2018. pp. 3911–3921 (2018).
32. Zhang, Y. et al.: Did You Ask a Good Question? A Cross-Domain Question Intention Classification Benchmark for Text-to-SQL. Presented at the (2020).
33. Zhong, R. et al.: Semantic Evaluation for Text-to-SQL with Distilled Test Suites. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 396–411 (2020).
34. Zhong, V. et al.: Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. Presented at the November (2017).

Author List

ISBN 978-1-98-839218-9