SWin: A Sliding Window Summarization Approach for Coherent LLM-driven Dialogue Systems

Shreya Prithviraj Savant

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering (CIISE)

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

August 2025

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Shreya Prithviraj Savant**

Entitled: **SWin: A Sliding Window Summarization Approach for Coherent LLM-driven Dialogue Systems**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________ Chair
Dr. Mohsen Ghafouri

______________________ Examiner
Dr. Walter Lucia

______________________ Supervisor
Dr. Jun Yan

______________________ Co-supervisor
Dr. Chun Wang

Approved by ______________________
Chun Wang, Chair
Concordia Institute for Information Systems Engineering (CIISE)

__________ 2025

____
Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

SWin: A Sliding Window Summarization Approach for Coherent LLM-driven Dialogue Systems

Shreya Prithviraj Savant

Large Language Models (LLMs) have revolutionized artificial intelligence (AI) driven dialogue systems, enabling various applications through advanced conversational capabilities. However, their stateless design impedes the development of sustainable assistants capable of evolving through extended interactions. Current memory mechanisms suffer from cascading error accumulation, architectural complexity, and computational inefficiency, which hinder scalability. To address this, we propose a novel sliding window-based memory module that dynamically updates the memory state using overlapping conversation windows, state-of-the-art prompt engineering techniques, and efficient summarization. The module balances contextual continuity with computational efficiency, reducing redundant token processing. This approach enhances the functionality of current mechanisms, and thorough tests on the Multi-Session Chat (MSC) dataset demonstrate that our approach is a reliable solution that exceeds automatic metrics produced by earlier approaches while optimizing token consumption. Our module's efficient design provides a workable solution for environments with limited resources, allowing dialogue systems to improve user interactions. Code, datasets, and evaluation scripts will be open-sourced to facilitate reproducibility and further research.

Acknowledgments

First and foremost, I would like to thank God for blessing me with the courage, strength, and perseverance to embrace and achieve all that the universe has in store for me.

I am deeply grateful to my supervisors, **Dr. Jun Yan** and **Dr. Chun Wang**, for their invaluable guidance, patience, and encouragement throughout the course of my degree.

Throughout my academic journey, I have been blessed with the unwavering support of my parents **Dhanashree** and **Prithviraj Savant**. Their constant encouragement and belief in me have been a source of immense strength during times of challenge. To my beloved grandfather, **Capt. Prabhakarrao Savant**, who is watching over me from above, I carry your determination with me, and I hope I make you proud. I am grateful to my uncle and mentor, **Mahesh Vilasrao Ghatge**, who has been an ideal role model and source of wisdom in times of doubt. My brother, **Siddhesh**, has been a continual source of motivation and support.

I would like to extend my heartfelt thanks to my friends and colleagues, who have reminded me of the importance of balance, happiness, and perspective beyond work.

Finally, I am grateful to the broader research community, open-source tools and the transformative potential of artificial intelligence that together have made this journey possible and inspiring. I also wish to acknowledge the support of **NSERC**, **7Dish**, **MITACS**, and **Concordia University**, whose resources and opportunities have been essential to the completion of this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Artificial Intelligence (AI) has quickly transformed from a niche research field into a cornerstone of modern technology, transforming day-to-day tasks and increasing productivity. Over the past few years, advances in machine learning, deep learning, and data have transformed nearly every area of life, from healthcare and education to finance, entertainment, and personal productivity. Conversational AI, in particular, has emerged as one of the most widely adopted and useful forms of AI, enabling natural interactions between humans and machines. Users now expect chatbots, virtual assistants, and dialogue systems not only to provide accurate information but also to engage in conversations that feel fluid, personalized, and human-like. This shift has elevated dialogue systems from simple utility tools into complex, adaptive agents capable of sustaining meaningful interactions.

At the heart of this revolution are Large Language Models (LLMs) such as GPT-4 [1], LLaMA [2], and PaLM [29], which have demonstrated unprecedented capabilities in natural language understanding and generation. By leveraging billions of parameters trained on vast corpora of web-scraped datasets, LLMs can generate coherent, contextually rich text. These systems have proven versatile across diverse domains like drafting documents, summarizing, answering complex questions, and engaging in open-ended conversation. Their impact is not only technical but cultural: LLMs have brought AI into everyday discourse, changing the way people view both the potential and limitations of AI. The rise of AI-powered productivity tools has not only simplified daily life but also reshaped how people think about AI. By automating repetitive tasks and reducing effort, these tools have eased much of the public's initial apprehension toward AI systems.

Despite these advances, a fundamental challenge remains: **contextual coherence and memory in extended conversations**. While LLMs excel in producing fluent responses in short exchanges, they often struggle to maintain continuity across multiple dialogue turns or sessions. For instance, a chatbot may recall a user's preference for Italian food in one conversation, but fail to remember it when the user returns the next day. This lack of long-term memory can lead to interactions that feel shallow, repetitive, or inconsistent. This can lead to a lack of trust in AI systems designed for companionship, healthcare guidance, or long-term assistance. In contrast, humans naturally integrate past exchanges and information into present interactions, drawing upon memory to sustain coherent, evolving conversations.

The notion of **coherence in dialogue** refers to the logical and contextual alignment of responses with prior conversational history. A coherent dialogue system avoids contradictions, maintains topic relevance, and demonstrates an understanding of the user's evolving intent. Memory in dialogue systems goes one step further, extending this coherence beyond a single session. It enables systems to build and preserve long-term knowledge about user preferences, past interactions, and contextual cues. Achieving both coherence and memory is therefore critical for realizing the next generation of AI systems that can grow with their users, adapt over time, and provide genuinely supportive, personalized interactions.

The pursuit of efficient, scalable, and cognitively inspired solutions remains at the forefront of conversational AI research. This thesis positions itself squarely within this pursuit by introducing **SWin (Sliding Window Summarization)**, a lightweight memory mechanism designed to balance efficiency with contextual richness. By taking inspiration from the way humans summarize and retain salient information, SWin represents a step toward dialogue systems that are not only intelligent but also consistent and coherent.

1.2 Motivation

The constraint of fixed-length context windows and stateless LLMs in dialogue systems hinders them from effectively recalling and integrating prior information, resulting in unnatural or confused conversations after multiple sessions. Positional interpolation [3] and optimized self-attention [4] have improved input window extension. However, they are computationally expensive, have limited application, or suffer from recently uncovered issues, like the "Lost in the Middle" phenomenon [5]. While external memory modules [6], [7] present a promising solution in this field, they introduce additional complexity in memory management, particularly in handling the **retrieval**, **updating**, and **forgetting** mechanisms.

Among the many additional modules incorporated into LLM application architectures, memory stands out as a defining component that distinguishes agents from conventional LLMs. Whereas LLMs operate primarily as predictive engines that generate responses based on immediate prompts, the integration of memory enables an agent to act as a persistent and adaptive entity (see Figure 1.1). Memory governs how an agent accumulates and organizes knowledge over time, how it draws upon historical experience, and how it strategically retrieves relevant information to guide decision-making and action. In this sense, memory transforms an LLM into a truly autonomous agent.

As illustrated in Figure 1.1, memory serves as the central hub linking diverse applications of agents across domains such as role-playing, open-world gaming, code generation, social simulation, recommendation, personal assistance, finance, and medicine. Each surrounding panel demonstrates how memory enriches the agent's ability to contextualize information and sustain coherence across interactions: from recalling character backstories in role-playing, to tracking inventory and prior strategies in open-world games, to reusing past code snippets in software development. In social simulation, memory allows agents to embody consistent roles, while in recommendation systems, it preserves personal preferences and evolving contexts. For personal assistants, memory maintains continuity across tasks; in finance, it records market trends and prior analyses; and in medicine, it integrates patient history with current symptoms. Together, we can conclude that memory is not a peripheral accessory but rather the backbone of agency, which supports information sources, storage mechanisms, and operational processes that make intelligent systems agentic.

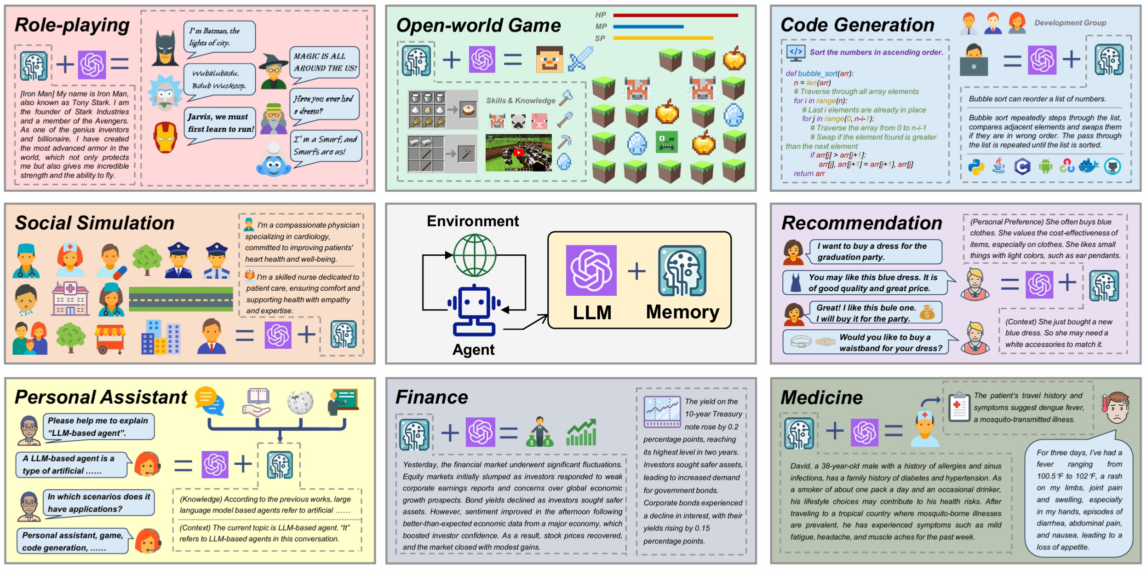

Role-playing
I'm Batman, the lights of city.
MAGIC IS ALL AROUND THE US!
Jarvis, we must first learn to run!
I'm a Smurf, and Smurfs are us!
Open-world Game
HP
MP
SP
Skills & Knowledge
Code Generation
Development Group
Sort the numbers in ascending order.
Bubble sort can reorder a list of numbers.
Social Simulation
I'm a compassionate physician specializing in cardiology, committed to improving patients' heart health and well-being.
I'm a skilled nurse dedicated to patient care, ensuring comfort and supporting health with empathy and expertise.
Environment
Agent
LLM
Memory
Recommendation
I want to buy a dress for the graduation party.
You may like this blue dress. It is of good quality and great price.
Great! I like this bule one. I will buy it for the party.
Would you like to buy a waistband for your dress?
Personal Assistant
Please help me to explain "LLM-based agent".
A LLM-based agent is a type of artificial ......
In which scenarios does it have applications?
Personal assistant, game, code generation, ...
Finance
Medicine
The patient's travel history and symptoms suggest dengue fever, a mosquito-transmitted illness.

Figure 1.1: The importance of the memory module in LLM-based agents. [48]

In response to this challenge, researchers have developed various strategies such as retrieval-augmented generation, external memory modules, and longer context windows. Yet these approaches introduce their limitations, including high computational costs, error accumulation, and architectural complexity. The motivation for this thesis is rooted in the increasing demand for dialogue systems that can sustain long-term, contextually coherent conversations. As LLM-powered assistants are deployed in high-stakes applications such as healthcare monitoring, psychological support, and education, the inability to maintain continuity across multiple sessions undermines user trust and system utility. In practice, users expect AI to "remember" prior interactions much like a human conversational partner would, and failures in this area are often perceived as frustrating or impersonal. Addressing this challenge is therefore not just a technical problem, but also a necessary step toward building socially acceptable and practically deployable AI systems.

Several promising efforts have been made in recent years to overcome these limitations. Context-window extension methods, such as positional interpolation and optimized self-attention mechanisms, allow LLMs to handle larger amounts of input text directly. While effective for short- to medium-length dialogues, these methods are prohibitively expensive to scale and still face well-documented issues like the "Lost in the Middle" phenomenon, where important information buried within long sequences is overlooked. On a different track, retrieval-augmented approaches have been proposed, in which external memory stores are queried for relevant passages to reinsert into the context. Although retrieval methods excel in factual recall, they struggle in conversational settings where subtle contextual cues or user-specific histories are needed to ensure continuity.

Another family of approaches focuses on structured long-term memory modules, often modelled after human cognition. Systems such as MemoryBank [7], MemoChat [12], and LD-Agent [13] use external storage, hierarchical summarization, or specialized persona modules to simulate memory. These approaches have achieved strong results in selected benchmarks but often introduce architectural overhead and high storage requirements, making them less practical for real-world deployment at scale. Importantly, some of these methods require extensive fine-tuning or customized model architectures, which limits their adaptability to rapidly evolving LLM ecosystems. For this reason, while they represent valuable contributions to the field, they fall outside the scope of direct experimental comparison in this thesis.

This work instead focuses on an alternative yet complementary line of investigation: summarization-based memory mechanisms. Summarization offers a practical compromise between efficiency and coherence by condensing dialogue histories into concise, semantically rich representations. Unlike retrieval-heavy methods, summarization reduces dependence on large external stores, and unlike full-context models, it avoids excessive token processing. The key challenge, however, lies in designing summarization workflows that minimize error accumulation while ensuring that critical conversational elements like user preferences, intents, and persona traits are preserved over time.

The Sliding Window Summarization (SWin) framework introduced in this thesis addresses these challenges directly. By dynamically updating memory using overlapping conversational windows and carefully engineered prompts, SWin seeks to capture the best of both worlds: computational efficiency from selective summarization, and contextual continuity from iterative memory updates. Through this approach, the thesis aims not only to demonstrate improvements over existing summarization-based methods but also to establish a generalizable framework that aligns more closely with human conversational memory patterns.

1.3 Problem Statement

Our proposed framework offers a simplified approach to long-term memory integration, ensuring the conversational relevancy of LLMs. By drawing on the foundational principles of human cognition, where memory is continuously updated and selectively utilized, we developed a streamlined approach to enhance LLMs' long-term memory capabilities. Our methodology eliminates the need for excessive computing, instead focusing on optimizing the intrinsic capabilities of LLMs through summarization, efficient prompting techniques, and reasonable models for the summarization and response generation tasks. We have built a lightweight memory module using a sliding window selector and COT prompting to surface memories based on relevance, recency, and importance [8].

1.4 Contributions

This research aims to bridge the gap between LLMs' sophisticated text-generation abilities and the nuanced requirements of long-term conversational continuity. By leveraging open-source and closed-source models, along with extensive testing and generalizability, our approach offers a scalable and efficient solution for real-world applications. Through extensive evaluation, we demonstrate the feasibility and impact of our proposed framework in advancing the frontier of long-term memory in conversational AI systems.

We evaluate our method on the Multi-Session Chat (MSC) dataset [9] and demonstrate its efficacy in producing responses that are not only contextually appropriate but also consistent with prior interactions. Our contributions are as follows:

(1) We present a novel sliding window-based summarization framework (SWin) for long-term dialogue memory management. We integrate an open-source LLM optimized for strong performance on summarization benchmarks to function seamlessly within the sliding window framework, achieving superior results compared to existing methods while maintaining lower resource requirements.

(2) An efficient memory update and retiring mechanism inspired by human cognitive models to

address cascading error accumulation and scalability in real-world applications.

(3) Extensive experimental validation across diverse dialogue scenarios showcasing improved performance over state-of-the-art quality and resource efficiency methods.

1.5 Thesis Organization

This chapter (Chapter 1) introduced the domain of open-domain chatbot systems and highlights the role of large language models (LLMs) in dialogue scenarios. The remainder of this thesis is structured as follows. Chapter 2 provides a comprehensive review of the literature, covering LLM architectures, memory mechanisms, open-source models, and advances in prompt engineering. Chapter 3 formulates the problem addressed in this study and defines the objectives of the proposed SWin framework, followed by a detailed description of its workflow and implementation. Chapter 4 describes the datasets, experimental setup, and evaluation methodology adopted. Chapter 5 reports and analyzes the empirical findings. Chapter 6 reflects on the challenges encountered, interprets the results in a broader context, and outlines potential avenues for future research. Finally, Chapter 7 summarizes the contributions of this work and presents the concluding remarks.

Chapter 2

Literature Review

This chapter provides a comprehensive review of the foundational approaches and recent developments in large language models, dialogue systems, and memory mechanisms for sustaining long-term conversational coherence. Section 2.1 traces the evolution of LLMs from early transformer-based architectures to the latest state-of-the-art models that represent modern dialogue systems. Section 2.2 examines dialogue systems more broadly, highlighting distinctions between open-domain and task-oriented approaches before delving into existing strategies for memory integration across extended interactions. Section 2.3 explores the role of LLMs in summarization tasks, including the rise of open-source alternatives, advances in prompt engineering, and emerging techniques designed to optimize performance, efficiency, and adaptability. Together, these sections establish the theoretical and methodological grounding for the sliding window summarization framework proposed in this thesis.

2.1 Large Language Models

Large Language Models have emerged as transformative architectures in natural language processing (NLP), redefining the landscape of dialogue systems, summarization, and text generation tasks. Their development builds on the introduction of the **transformer architecture** by Vaswani et al. in 2017 [43], which replaced recurrent neural networks with self-attention mechanisms capable of modeling long-range dependencies more efficiently. This innovation paved the way for scaling model parameters, data, and computational resources, leading to remarkable improvements in NLP tasks.

Early large-scale implementations, such as **BERT** [44] and **GPT-2** [45], demonstrated the utility of pre-training on massive text corpora followed by fine-tuning for downstream tasks. BERT emphasized bidirectional context for classification and understanding and GPT models showcased auto-regressive generation, enabling coherent open-ended text production. These milestones culminated in the release of **GPT-3** [28], which introduced in-context learning and few-shot prompting as alternatives to traditional fine-tuning. GPT-3's ability to generalize across tasks without additional training underscored the paradigm shift toward universal foundation models.

The trajectory of LLMs continued with instruction-tuned models such as FLAN-T5 [46] and InstructGPT, which improved alignment with human intent through reinforcement learning from human feedback (RLHF). These approaches addressed early criticisms of LLMs like their tendency to generate verbose, irrelevant, or unsafe responses by fine-tuning them with curated datasets and preference modeling. Concurrently, open-source alternatives such as LLaMA [27], Falcon, and Mistral demonstrated that smaller, publicly available models could rival closed-source systems in efficiency and accuracy. This democratization of LLM research has spurred rapid innovation while reducing reliance on proprietary platforms.

Recent developments, including GPT-4 and Qwen2.5 series, illustrate the trend toward multimodal capabilities, supporting not only text but also images, audio, and code generation. These models integrate long-context processing, improved reasoning, and chain-of-thought prompting, making them suitable for sustained dialogue, retrieval-augmented generation (RAG), and knowledge-intensive tasks. However, despite their advances, LLMs still face persistent challenges in long-term memory integration, factual consistency, and efficiency at scale. Research has highlighted issues such as the *lost in the middle* effect, where models disproportionately focus on recent context while neglecting earlier inputs, and hallucination, where fabricated content undermines reliability in sensitive domains.

Thus, the evolution of LLMs reflects a continuous negotiation between scale, efficiency, and alignment. Their success in generative and interactive tasks has positioned them as the core of modern dialogue systems. Yet, their limitations in handling extended interactions and contextual memory remain open challenges, motivating approaches such as the sliding window summarization framework proposed in this thesis.

2.1.1 Dialogue State and Statelessness in Large Language Models

In the context of dialogue systems, the state refers to an internal representation of the ongoing interaction. This representation is not merely a transcript of utterances but rather a structured encoding of the conversation history encompassing facts explicitly mentioned, implicit user preferences, dialogue goals, and unresolved intents.

Large Language Models (LLMs), are fundamentally stateless in design. Their outputs are conditioned only on the input provided within a bounded context window. Once this window is exceeded, earlier dialogue history is effectively “forgotten” unless explicitly re-supplied as input. Unlike symbolic dialogue managers, the model does not persist a latent representation of prior interactions across sessions. This statelessness creates challenges for extended or personalized conversations, where users expect continuity.

To address this limitation, researchers have explored methods of integrating external memory modules. For instance, our architecture, SWin, provides mechanisms to augment LLMs with persistent memory of past dialog. Such modules enable retrieval of relevant past dialogue segments or structured representations, thereby approximating a **pseudo-state**. Importantly, this external layer does not alter the underlying stateless nature of the LLM itself; rather, it simulates statefulness by dynamically feeding back relevant information into the model’s context.

Our work adopts this approach by introducing an external memory layer that interfaces with the LLM to preserve dialogue continuity. This enables the system to recall prior user preferences, track unresolved intents, and maintain coherence across sessions. While the model remains stateless at its core, the combined system produces an illusion of statefulness, aligning more closely with human expectations of conversational agents.

2.2 Dialogue Systems

A dialogue system is a machine-based system that aims to communicate with humans through conversation via text, speech, images, and other communication modes as input or output [40]. Simply dialogue systems, also known as conversational agents or chatbots, are computer systems designed to interact with humans in natural language, either through text or speech.

Dialogue systems can be broadly divided into two main categories: open-domain dialogue systems (which converse on any topic, like ChatGPT) and task-oriented dialogue systems (which help users accomplish specific goals, such as booking a ticket). Many modern systems explore hybrid approaches, combining elements of both to improve flexibility and utility. [41] These systems have evolved substantially over the decades, transitioning from rule-based chatbots with fixed templates to sophisticated AI-driven systems capable of open-domain conversations and multi-modal understanding. The difference between open-domain and task-oriented dialogue systems lies mainly in their goals, functionality, and conversational flexibility (As shown in Table 2.1)

Table 2.1: Comparison of Open-Domain and Task-Oriented Dialogue Systems

Aspect	**Open-Domain Dialogue Systems**	**Task-Oriented Dialogue Systems**
Goal	Engage users in broad, unrestricted conversations, not tied to specific tasks.	Help users complete specific tasks or objectives through goal-oriented interaction.
Functionality	Converse naturally and freely; support casual chats, storytelling, and general discussions.	Structured conversations using predefined tasks, ontologies, and slot-filling mechanisms.
Flexibility	Highly flexible; can switch topics seamlessly without domain restrictions.	Limited to predefined domains and tasks; less topic flexibility.
Examples	Chatbots like ChatGPT, BlenderBot, and social chatbots.	Virtual assistants like Siri, Alexa, customer service bots, and tech support agents.

At their core, dialogue systems comprise several interacting components: natural language understanding (NLU) to process user input, dialogue management modules that track conversational state and determine appropriate system actions, and natural language generation (NLG) to produce coherent, contextually relevant responses.

Modern dialogue systems frequently incorporate advanced neural models, including sequence-to-sequence architectures and large language models, to enhance their conversational capabilities and adaptability across various domains. With the popularity of transformer-based architectures and large-scale pretraining, modern dialogue systems now leverage Large Language Models to generate coherent and contextually rich responses. Despite these advancements, maintaining dialogue consistency over long interactions remains a fundamental challenge due to limitations in memory and context handling.

2.2.1 Memory Mechanisms for Long-Term Dialogue Systems

Table 2.2 presents a comparative analysis of existing memory management techniques, datasets, and challenges encountered in long-term discussions. The table highlights the key strengths and weaknesses of existing methods, including insufficient attention to the accumulation of errors and the challenges associated with long context window models. The method proposed in this study overcomes these limitations by enabling LLMs to generate and update comprehensive conversational summaries. This approach increases the consistency and coherence of the conversation by allowing the model to leverage these summaries.

Wang et al. [6] propose a novel and effective memory mechanism to address the limitations of large language models in maintaining coherence and consistency across long-term dialogues. Their approach, termed recursively summarizing, enables LLMs to generate and update memory representations by successively summarizing past conversational sessions as shown in 2.1. In this method, the sessions are denoted by S_i for $i = 1, 2, 3$, while the corresponding memory updates are represented by M_i for $i = 1, 2, 3$. Unlike prior memory-based methods that rely on static or externally managed memory banks, this framework prompts the LLM itself to act as both a memory manager and response generator. At the end of each dialogue session, the model updates its memory by integrating the current session's content with the previously stored memory, effectively compressing relevant semantic information in a scalable, language-native format. This recursive process builds a dynamic memory that evolves with the conversation, enabling the model to maintain persona

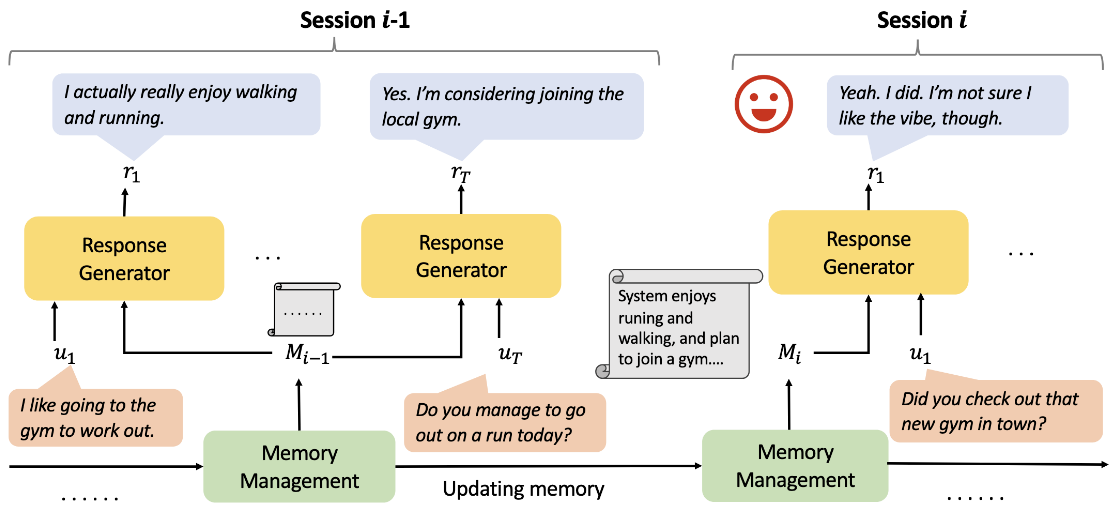

Session *i*-1
Session *i*
I actually really enjoy walking and running.
Yes. I'm considering joining the local gym.
Yeah. I did. I'm not sure I like the vibe, though.
r_1
r_T
r_1
Response Generator
Response Generator
Response Generator
. . .
. . .
......
System enjoys runing and walking, and plan to join a gym....
u_1
M_{i-1}
u_T
M_i
u_1
I like going to the gym to work out.
Do you manage to go out on a run today?
Did you check out that new gym in town?
Memory Management
Memory Management
Updating memory
......
......

Figure 2.1: The schematic overview of the Recursively Summarizing method. u_t and r_t represent the dialogue utterances from the user and system at step t, respectively. When the i - 1 session ends, we update the memory M_i - 1 to M_i using the whole session context. [6]

traits, track user preferences, and preserve discourse coherence over extended interactions. Empirical results on the Multi-Session Chat dataset demonstrate that the proposed method consistently outperforms baseline strategies, including vanilla LLMs, MemoChat, and MemoryBank, in both automatic and human evaluations. Notably, the approach is model-agnostic and complements existing strategies such as retrieval-augmented generation and context window extension. Moreover, the authors highlight that recursively generated memory, though compact, tends to be more coherent and usable than gold (human-written) memory annotations, suggesting its utility in both zero-shot and few-shot settings.

Lee et al. [11] present a novel methodology for constructing high-quality open-domain conversational agents using a modular, prompting-based approach with large pre-trained language models, sidestepping the need for computationally expensive fine-tuning. Their proposed system, as shown in 2.2.1, Modular Prompted Chatbot (MPC), decomposes the chatbot architecture into specialized modules, such as clarifiers, memory processors, summarizers, and utterance generators, each leveraging prompt engineering techniques like few-shot learning and chain-of-thought (CoT) prompting. This modularization allows the system to maintain long-term persona consistency and contextual coherence across extended dialogues. By retrieving and updating contextual memory through Dense

Passage Retrieval (DPR) and strategically condensing relevant information, MPC effectively tracks user and bot states throughout a conversation. Notably, Lee et al. demonstrate that their approach performs on par with or better than state-of-the-art fine-tuned models like BlenderBot 3 in human evaluations across metrics of sensibleness, consistency, engagingness, and user preference. The study further reveals that instruction-tuned and larger base models enhance performance, while the modular design confers robustness and flexibility, making it easier to substitute or optimize individual components. Overall, this work substantiates the efficacy of prompted LLMs for scalable, adaptable, and coherent dialogue systems and contributes a compelling alternative to traditional fine-tuning in chatbot development.

In the paper *MemoChat: Tuning LLMs to Use Memos for Consistent Long-Range Open-Domain Conversation* [12], the authors introduce a novel instruction tuning framework, *MemoChat*, that equips LLMs with the ability to self-generate and utilize structured "memos" to maintain consistency across extended open-domain dialogues. The approach is motivated by the growing need for chatbots to handle long-range conversations that span multiple, diverse topics, a setting in which conventional methods relying on extended context windows or external memory retrievers often struggle due to limitations in input size and error accumulation in retrieval. MemoChat circumvents these challenges by embedding a self-contained, LLM-internal pipeline composed of three core stages, memorization, retrieval, and response, each supported by custom-designed instruction templates also demonstrated in Fig. 2.2.1. These instructions are reconstructed from public datasets (TopicoQA, DialogSum, Alpaca-GPT4) and guide LLMs to categorize dialogues by topic, summarize them in real time, retrieve relevant prior content based on new queries, and generate coherent responses. The authors build and annotate a dedicated test set, MT-Bench+, to evaluate consistency under various long-range scenarios (retrospection, continuation, and conjunction). Extensive experiments show that MemoChat-tuned models, particularly larger-scale ones like Vicuna-13B and 33B, significantly outperform both vanilla ChatGPT and state-of-the-art memory-augmented baselines (e.g., MPC-ChatGPT, MemoryBank-ChatGPT) in response consistency, even with modest amounts of fine-tuning data. Furthermore, the paper details key challenges in instruction design, such as prompt leakage and catastrophic forgetting, and proposes practical solutions, such as balanced training data and dummy variable substitution. Overall, *MemoChat* emerges as a powerful

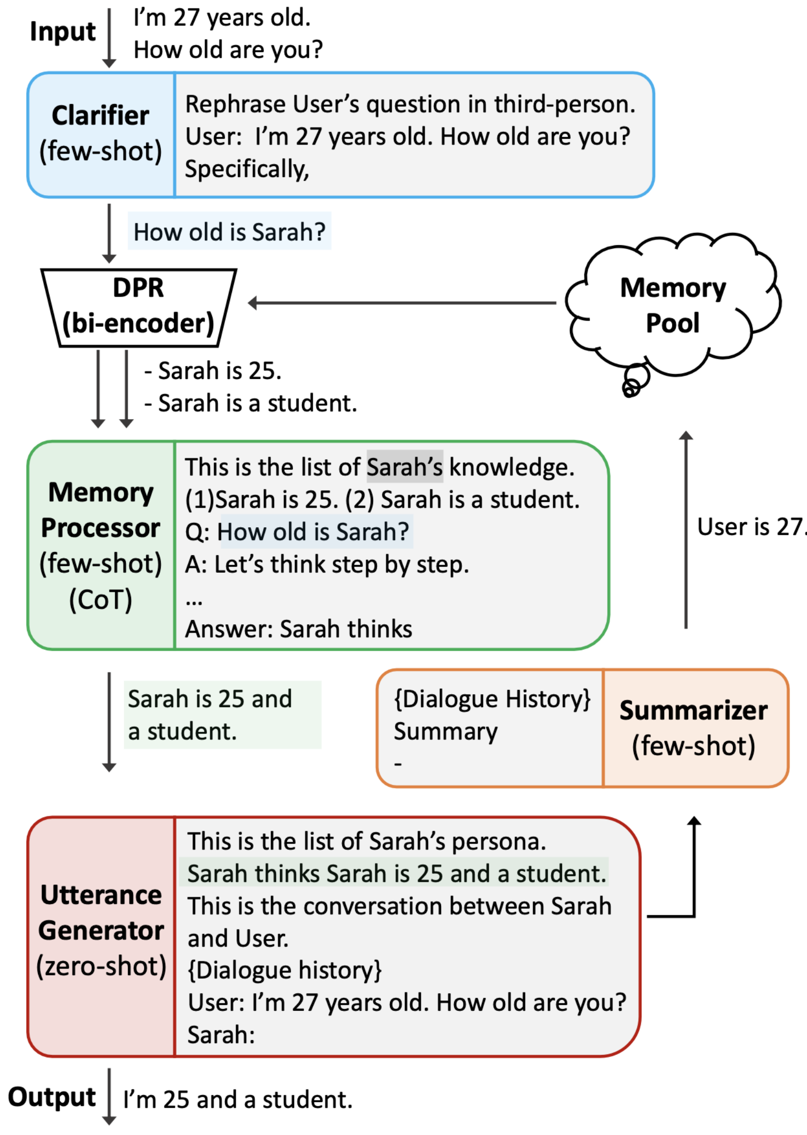


Figure 2.2: The Prompted LLMs as Chatbot modules [11] workflow.

framework that enhances the long-range memory capabilities of LLMs without external augmentation.

In their influential paper, Zhong et al. [7] present *MemoryBank*, a pioneering memory mechanism designed to endow LLMs with human-like long-term memory. *MemoryBank* introduces a modular architecture encompassing three key components: a memory storage that logs conversations, summarizes events, and synthesizes user personality; a retrieval module based on dense passage retrieval for context-aware memory access; and a biologically inspired memory updating

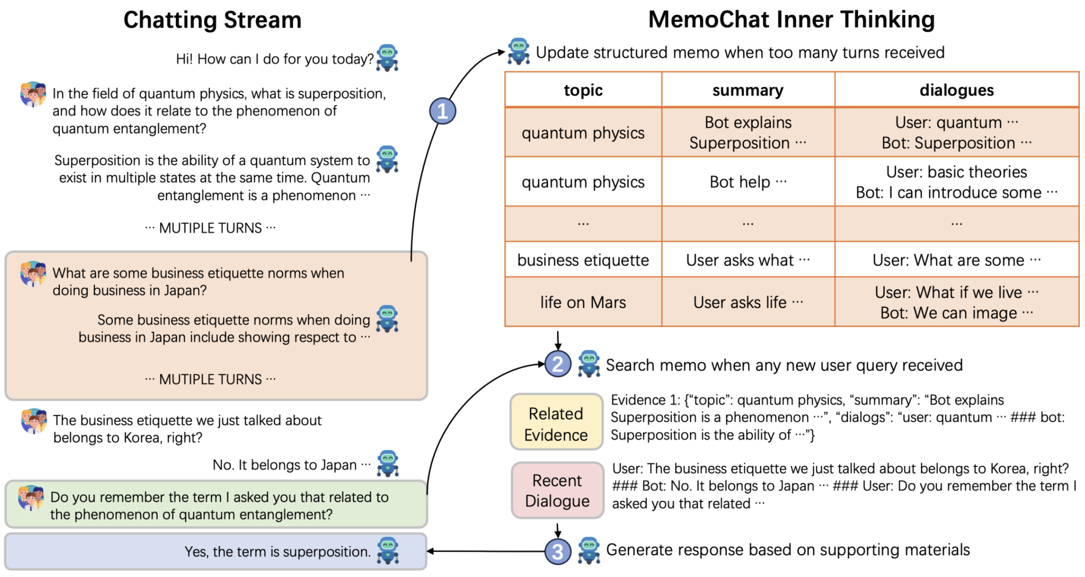

Chatting Stream
MemoChat Inner Thinking
Hi! How can I do for you today?
In the field of quantum physics, what is superposition, and how does it relate to the phenomenon of quantum entanglement?
Superposition is the ability of a quantum system to exist in multiple states at the same time. Quantum entanglement is a phenomenon ···
··· MUTIPLE TURNS ···
What are some business etiquette norms when doing business in Japan?
Some business etiquette norms when doing business in Japan include showing respect to ···
··· MUTIPLE TURNS ···
The business etiquette we just talked about belongs to Korea, right?
No. It belongs to Japan ···
Do you remember the term I asked you that related to the phenomenon of quantum entanglement?
Yes, the term is superposition.
1 Update structured memo when too many turns received
topic | summary | dialogues
quantum physics | Bot explains Superposition ··· | User: quantum ··· Bot: Superposition ···
quantum physics | Bot help ··· | User: basic theories Bot: I can introduce some ···
··· | ··· | ···
business etiquette | User asks what ··· | User: What are some ···
life on Mars | User asks life ··· | User: What if we live ··· Bot: We can image ···
2 Search memo when any new user query received
Related Evidence
Evidence 1: {"topic": quantum physics, "summary": "Bot explains Superposition is a phenomenon ···", "dialogs": "user: quantum ··· ### bot: Superposition is the ability of ···"}
Recent Dialogue
User: The business etiquette we just talked about belongs to Korea, right? ### Bot: No. It belongs to Japan ··· ### User: Do you remember the term I asked you that related ···
3 Generate response based on supporting materials

Figure 2.3: The MemoChat [12] workflow.

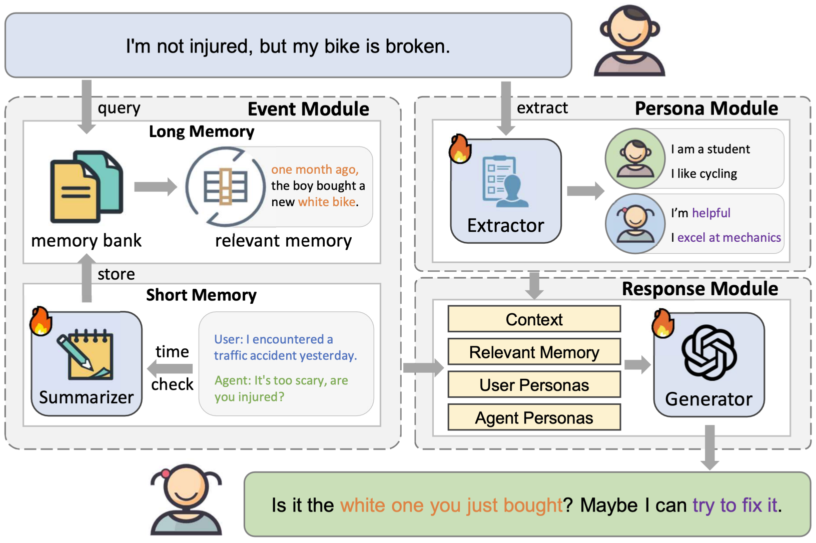


Figure 2.4: The LD-Agent [13] framework.

mechanism modelled on **Ebbinghaus' Forgetting Curve**, which emulates selective forgetting and reinforcement. This system enables LLMs to recall past interactions, personalize responses over time, and adapt to individual users. To validate this framework, the authors develop SiliconFriend, an AI chatbot fine-tuned with 38,000 psychological dialogues to enhance emotional intelligence and empathy. Integrated with MemoryBank, SiliconFriend supports both open-source (ChatGLM,

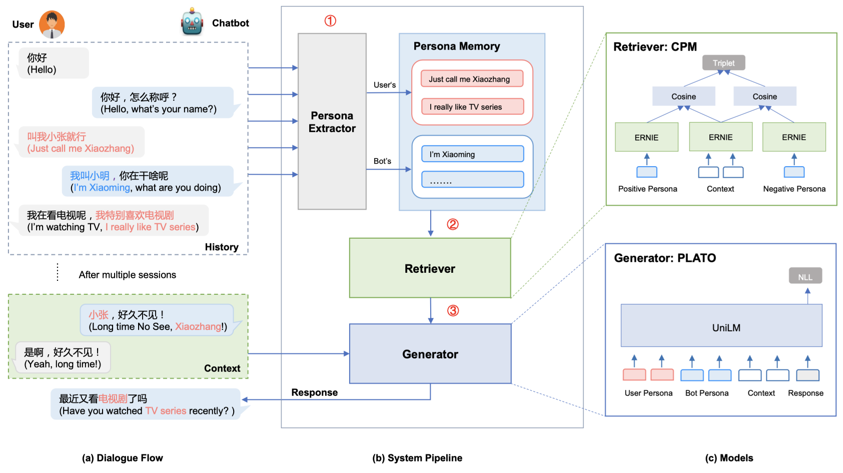


Figure 2.5: Illustration of the PLATO-LTM system (a) shows their dialogue flow. (b) describes their modules and the pipeline of the system. (c) details the generator PLATO-2 and ranker CPM (Context Persona Matching) [14].

BELLE) and closed-source (ChatGPT) models and operates bilingually in English and Chinese. Experimental results, including qualitative user dialogues and a quantitative benchmark using 194 memory-probing queries across 10 days of simulated interactions with diverse user personas, show significant improvements in memory recall, contextual coherence, and empathetic response generation. The model not only retains factual recall but dynamically constructs user profiles, enabling tailored and emotionally aware interactions. The findings suggest that MemoryBank substantially enhances the realism and usability of LLMs in long-term engagement scenarios, offering a compelling pathway toward more human-like, adaptive AI systems.

Hello Again! LLM-powered Personalized Agent for Long-term Dialogue by Li et al. [13] is another such mechanism introducing LD-Agent. The study responds to the limitations of conventional dialogue systems, which are typically constrained to short-term, single-session interactions, by emphasizing the importance of event memory and persona consistency across prolonged conversations. LD-Agent is built upon three modular components: event memory perception, dynamic persona extraction, and response generation. They work in concert to emulate human-like memory, identity, and conversational reasoning. The memory module (shown in Fig. 2.2.1) integrates both long-term memory banks (for high-level summaries of prior sessions) and short-term caches (for current dialogue contexts), enhanced by a topic-based retrieval mechanism that improves relevance through

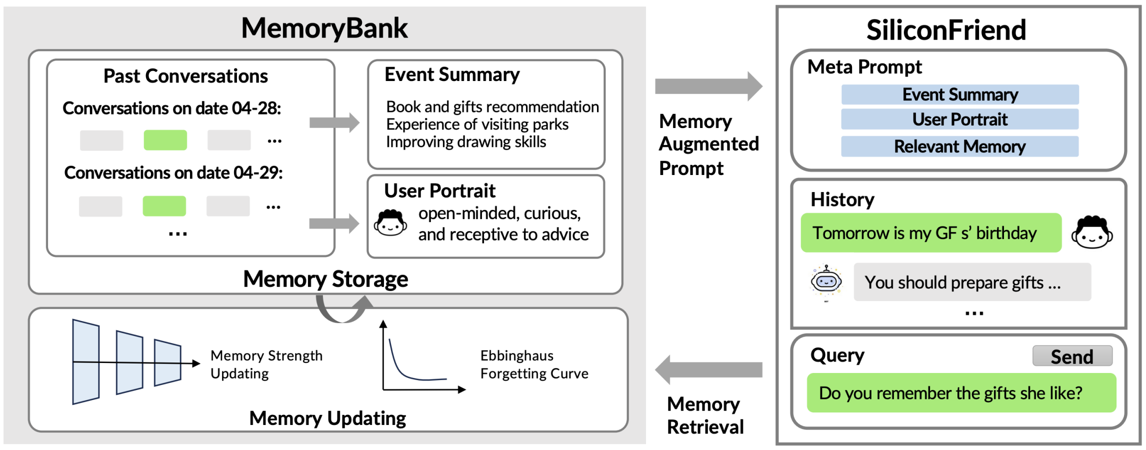

MemoryBank
Past Conversations
Conversations on date 04-28:
Conversations on date 04-29:
Event Summary
Book and gifts recommendation
Experience of visiting parks
Improving drawing skills
User Portrait
open-minded, curious, and receptive to advice
Memory Storage
Memory Strength Updating
Ebbinghaus Forgetting Curve
Memory Updating
Memory Augmented Prompt
Memory Retrieval
SiliconFriend
Meta Prompt
Event Summary
User Portrait
Relevant Memory
History
Tomorrow is my GF s' birthday
You should prepare gifts ...
Query
Send
Do you remember the gifts she like?

Figure 2.6: Overview of MemoryBank [7]. The memory storage stores past conversations, summarized events and user portraits, while the memory updating mechanism updates the memory storage. Memory retrieval recall relevant memory. SiliconFriend serves as an LLM-based AI companion augmented with MemoryBank.

semantic similarity, topic overlap, and temporal decay. Meanwhile, the persona module employs a bi-directional user-agent modelling strategy, using instruction tuning and LoRA-enhanced persona extraction to ensure fidelity and adaptability. Responses are generated by synthesizing insights from both memory and persona modules, allowing the agent to exhibit contextual awareness and character coherence. Extensive experiments across benchmark datasets, including MSC, CC, and Ubuntu IRC, demonstrate LD-Agent's superiority over existing models like HAHT and BlenderBot, with notable improvements in BLEU, ROUGE, and METEOR scores, as well as human-rated coherence, fluency, and engagement. The framework proves robust across zero-shot and fine-tuned settings, traditional and LLM-based architectures, and even multiparty dialogues.

Xu et al. [14] propose a novel task, Long-term Memory Conversation (LeMon), to address this limitation. Recognizing that existing systems often struggle with long-term coherence due to an inability to retain and utilize historical persona information, the authors develop both a new benchmark dataset, DuLeMon, and a dialogue generation framework, PLATO-LTM, to fill this gap. Unlike earlier persona-based models that either rely on static self-descriptions or ignore user persona altogether, DuLeMon introduces dynamic, mutual persona modelling in Chinese, incorporating both chatbot and user persona information that evolves across sessions. The dataset explicitly annotates persona grounding at the utterance level and is designed to simulate realistic, multi-turn dialogues

Table 2.2: Comparison of Memory Mechanisms

Approach / Framework	**Key Features**	**Limitations**
Recursively Summarizing [6]	Builds memory by summarizing previous dialogue recursively after each turn.	Prone to error accumulation; consumes many tokens and LLM calls.
MPC [11]	Summarizes user personality and dialogue facts for later retrieval.	No dynamic updates; memory storage is inefficient.
MemoChat [12]	Encodes dialogue facts and trains LLMs to retrieve relevant memory during generation.	High storage overhead; lacks incremental updating.
MemoryBank [7]	Captures important dialogue segments and merges them after sessions.	Limited conversational granularity during interaction.
HAHT [13]	Stores long-term conversational memory for context in multi-session chats.	Scalability challenges with memory size.
LeMon [14]	Dynamically maintains user and bot personas to ensure consistency.	Focuses only on persona-related memory; lacks general-purpose recall.

where one participant plays the user and the other the chatbot. The PLATO-LTM framework enhances PLATO-2 by integrating a plug-and-play Long-Term Memory (LTM) module comprising a persona extractor, dual memory banks, and a retrieval-augmented generation module. This structure (in fig. 2.2.1) allows for real-time persona extraction, memory updating, and retrieval without requiring long-session training data. Experiments, including both automatic and human evaluations, demonstrate that PLATO-LTM significantly improves dialogue consistency, coherence, and engaging-ness over strong baselines like PLATO-2 and PLATO-FT. Notably, the persona extractor module contributes to superior memory precision, while the memory mechanism boosts both recall and personalization of generated responses.

In Beyond Goldfish Memory: Long-Term Open-Domain Conversation [9], Xu, Szlam, and Weston (2022) identify a critical gap in the development of dialogue systems: the inability of current state-of-the-art models to handle long-term, multi-session conversations. While the then-existing models, such as BlenderBot and Meena, achieve high performance on short, single-session dialogues, they fail to maintain continuity over time due to their limited input context windows (often capped at 128 tokens) and the absence of mechanisms for memory retention. To address this, the authors introduce the Multi-Session Chat (MSC) dataset, a novel and publicly available resource containing extended human-human dialogues across up to five distinct sessions, where speakers build rapport and recall prior exchanges. This dataset is purposefully constructed to emulate realistic conversational dynamics over time, incorporating both full dialogue histories and concise, crowd-sourced session summaries that encapsulate key personal information shared by interlocutors. The paper rigorously evaluates standard encoder-decoder architectures alongside two advanced long-context modeling approaches: retrieval-augmented generation (RAG, FiD, FiD-RAG) and a proposed memory-augmented model, SumMem-MSC, which uses abstractive summarization to persist salient dialogue features. Empirical results, spanning both automatic metrics (e.g., perplexity, BLEU, F1) and human evaluations, consistently show that models with explicit memory mechanisms, especially those leveraging summarization and retrieval, outperform traditional baselines in generating coherent, engaging, and contextually grounded responses across sessions. The authors' findings not only challenge the prevailing paradigm of short-context evaluation in dialogue research but also offer a practical blueprint for building more personalized and temporally-aware conversational agents, with direct implications for deployment in real-world settings where user re-engagement and memory continuity are crucial.

2.3 Summarization using LLMs

LLMs have demonstrated exceptional abilities in various language-related tasks, including conversation, summarization, and creative writing. LLMs have become central to automatic text summarization, demonstrating strong capabilities in both extractive and abstractive summarization. Modern LLMs such as the GPT family (e.g., GPT-3 and its successors) offer a paradigm-flexible approach that can seamlessly integrate extractive and abstractive techniques within a single model [42]. LLM-driven summarization has broad applications across domains: they are used to condense news articles, scientific and medical literature, legal documents, and other lengthy texts into manageable summaries, addressing information overload in those fields. Nonetheless, despite their advantages, LLM-based summarizers face persistent challenges. They can hallucinate content that was not in the source, resulting in factual inconsistencies, and may struggle with domain-specific terminology or knowledge gaps if not properly fine-tuned for the domain.

2.3.1 Open-Source Models for Summarization

In recent advancements in natural language processing, particularly within the domain of text summarization, a comparative study of both closed-source and open-source large language models has revealed notable insights relevant for academic and industrial applications. The study rigorously evaluated the performance of state-of-the-art proprietary models like GPT-4 [1], GPT-3.5 [28], and PaLM-2 [29] against leading open-source alternatives like LLaMA-2 [16]. A key outcome of the investigation is the finding that smaller, open-source models can deliver summarization performance on par with their larger, closed-source counterparts, even in zero-shot settings where no task-specific fine-tuning is performed. This performance parity underscores a significant shift in the practical viability of open-source LLMs for real-world applications. Moreover, from a strategic point of view, open-source models offer critical advantages in terms of cost efficiency, transparency, and data privacy factors that are especially consequential in industrial deployments where control over data and operational expenses is paramount. As such, the study positions open-source summarization models not merely as competent alternatives but as increasingly preferable solutions for organizations seeking scalable and secure natural language solutions without reliance on proprietary platforms. This aligns with the broader movement towards democratizing AI tools and highlights the accelerating maturity of open-source ecosystems in high-impact NLP tasks.

2.3.2 Prompt Engineering

Prompt engineering has emerged as a crucial technique for optimizing the performance of LLMs in various natural language processing tasks, including summarization. This approach involves carefully crafting input prompts to guide LLMs toward generating more accurate and relevant outputs [17]. In the context of summarization, prompt engineering has shown significant promise. In domains like healthcare and IT operations, researchers have employed techniques such as shot prompting and pattern prompting to enhance the quality of automated medical reports [18]. Similarly, in IT operations, prompt engineering has been used to generate insightful summaries of incident data, helping Site Reliability Engineers quickly understand and resolve faults [19] [20]. Beyond summarization, prompt engineering has applications in text classification, question-answering systems, and other NLP tasks [21]. The exploration of Chain-of-Thought (COT) and Reflection techniques for building more sophisticated LLM-based systems is one of the most transformative applications of this technique. [22]. We find that one-shot prompts, combined with COT reasoning, yield the best results in our work.

Prompt engineering is a critical technique for optimizing the performance of LLMs and generative AI systems. It has emerged as an essential skill in the age of artificial intelligence, fundamentally changing how we interact with and leverage these powerful models [34].

Research has identified numerous prompt engineering techniques, with comprehensive surveys documenting 58 LLM prompting techniques and 40 techniques for other modalities [35]. The most prominent techniques include:

Chain-of-Thought (CoT): Prompting emerges as one of the most effective methods, particularly for mathematical and logical reasoning tasks. [37] This technique guides models through step-by-step reasoning processes, significantly improving performance on complex problems. Studies show that prompts incorporating CoT can outperform basic approaches by 6.3% on MultiArith and 3.1% on GSM8K [36].

Few-shot prompting involves providing the model with a few examples of the desired task before asking it to perform a similar task. This technique has proven particularly effective for tasks requiring pattern recognition and consistency [38].

In-Context Learning (ICL) represents another crucial technique where models make predictions based on contexts augmented with examples [39]. Research indicates that ICL-centric strategies are particularly effective for natural language understanding tasks.

Step-by-Step Reasoning (SSR) and **Tree of Thought (ToT)** techniques have shown particular promise for complex reasoning tasks [38]. These methods guide models through structured thinking processes, leading to more accurate and reliable outputs.

Table 2.3: Key works on long-term memory in LLMs

Title	**Authors & Year**	**Venue / Source**	**Key Contribution**
MemoryBank: Enhancing Large Language Models with Long-Term Memory	Wanjun Zhong et al., 2023	Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, Canada	Introduces a memory mechanism mimicking human forgetting (Ebbinghaus curve) with reinforcement. Enables LLMs (ChatGPT, ChatGLM) to maintain evolving, user-aware memory. Demonstrated via empathetic chatbot (SiliconFriend).
MemoChat: Tuning LLMs to Use Memos for Consistent Long-Range Open-Domain Conversation	Junru Lu et al., 2023	arXiv (Aug 2023)	Proposes "memorization–retrieval–response" pipeline using self-composed memos. Improves consistency in long-range conversations across models.
HAHT: History-Aware Hierarchical Transformer for Multi-Session Dialogue	Tong Zhang et al., 2023	Findings of the Association for Computational Linguistics: EMNLP 2022	Hierarchical memory encoding across sessions; uses attention and "history-aware" vocabulary switch. Outperforms baselines in maintaining context.
HMT: Hierarchical Memory Transformer	Zifan He et al., 2024	arXiv (May 2024)	Uses segment-level recurrence and hierarchical memory for long-context tasks; matches/surpasses LLMs with fewer resources.
Recursively Summarizing Enables Long-Term Dialogue Memory	Qingyue Wang et al., 2023/2025	Neurocomputing (2025) / arXiv (2023)	Employs recursive summarization to build/update memory for dialogues. Complements long-context and retrieval methods.
Evaluating Very Long-Term Conversational Memory of LLM Agents	Maharana et al., 2024	ACL (2024)	Introduces LoCoMo dataset (600-turn, 16K-token dialogues across 32 sessions). Benchmarks LLMs and RAG systems, exposing memory limitations.
On Memory Construction & Retrieval for Personalized Conversational Agents (SeCom)	Zhuoshi Pan et al., 2025	Preprint / OpenReview (Feb 2025)	Shows impact of conversational granularity & compression on retrieval. Introduces a segmentation-based memory bank with denoising.
M3-Agent: Multimodal Agent with Long-Term Memory	Lin Long et al., 2025	arXiv (Aug 2025)	Multimodal agent with episodic/semantic memory graphs from real-time video & audio. Outperforms baselines on M3-Bench video QA.

Chapter 3

Methodology

3.1 Problem Formulation

The development of long-term dialogue systems using Large Language Models presents a fundamental challenge: maintaining coherence, contextual relevance, and computational efficiency over extended multi-session interactions. While LLMs such as GPT-4 and LLaMA-3 demonstrate strong capabilities in short-term text generation, their ability to sustain meaningful context over prolonged exchanges is constrained by finite context windows and the absence of persistent memory mechanisms. Existing solutions, including full-history, retrieval-augmented generation, and recursive summarization, either incur prohibitive computational costs or suffer from cascading error accumulation. Our goal is to formalize this challenge as a technical problem that can be addressed through an efficient memory design.

3.1.1 Dialogue Representation

Let a conversation at time t be represented as a sequence of dialogue turns:

$$C_t = \{T_1, T_2, \ldots, T_n\},$$

where each turn T_i consists of a user utterance u_i and a corresponding chatbot response b_i:

$$T_i = (u_i, b_i).$$

At step k, a new user query Q_k is introduced, requiring the system to generate a response R_k. The challenge lies in generating R_k such that it is

i. coherent with prior exchanges
ii. relevant to the user query
iii. efficient in terms of token and computational usage.

3.1.2 Memory-Aware Dialogue Generation with Prompts

We define a memory module M_{k-1} that selectively preserves salient information from past interactions up to turn $k-1$. In our framework, both response generation and memory updating are conditioned not only on dialogue context but also on specifically designed prompts:

$$R_k = f_{\text{resp}}(Q_k, M_{k-1}, P_{\text{resp}}),$$

where f_{resp} denotes the response generation function of the LLM, and P_{resp} is the structured response prompt that guides the model toward coherent, contextually appropriate answers.

Similarly, the memory update process is defined as:

$$M_k = f_{\text{update}}(M_{k-1}, T_k, P_{\text{mem}}),$$

where f_{update} denotes the summarization-based update function, and P_{mem} is the memory update prompt that enforces concise, faithful, and non-redundant summaries of past interactions.

3.1.3 Problem Constraints

- **Contextual Continuity:** R_k should reflect dependencies across multiple sessions, preserving user persona, preferences, and previously learned facts.

- **Resource Efficiency:** The memory update and retrieval process should minimize redundant token usage, scaling sublinearly with the length of the dialogue history.
- **Error Robustness:** The system should avoid error propagation that arises from recursively summarizing entire sessions or from inaccurate retrievals.
- **Prompt Dependence:** Both P_{resp} and P_{mem} must be carefully designed to balance informativeness with conciseness, as they directly affect model behaviour.

3.1.4 Research Objective

The technical problem can thus be stated as follows:

Given a multi-session dialogue C_t with evolving user interactions, design a memory module M such that the generated response R_k at each turn is coherent, relevant, and resource-efficient.

To address this, we propose **SWin**, a sliding window-based summarization mechanism that updates memory dynamically using overlapping conversational segments and structured prompts $(P_{\text{resp}}, P_{\text{mem}})$. This formulation serves as the foundation for the design and algorithms described in the following sections.

3.1.5 Steps in Sliding Window Summarization

Define the Window Size: A fixed number of dialogue turns (or data entries) is chosen to represent the "window." For example, a window size of 3 means the system processes 3 dialogue turns simultaneously. An optimal window size is crucial and should be determined through empirical testing, as it varies based on the specific use case, domain requirements, and the conversational complexity of the AI system.

Segment the Data: The conversation history is divided into overlapping or non-overlapping segments based on the window size. Overlapping segments ensure context continuity, where some portion of one window overlaps with the next.

Summarize Each Window: For each window, the system generates a summary that encapsulates the key facts, events, or conversational details while incorporating context from the previous window's summary. The summarization process includes extracting key entities, defining user and bot personas, and crafting a concise and coherent memory, a thorough representation of the past dialogue.

Update and Store: As the conversation progresses, the oldest window is replaced with a new one, and the corresponding summaries are updated dynamically. For every new session, we start with the previous session's resultant summary and reset the window for subsequent conversation turns between the user and the bot.

Retrieval of Memory for Context: When generating responses, the system uses the most recent summary to provide contextually relevant and coherent answers. The summarized memory ensures efficient retrieval with no need to process the entire conversational history (naive Vanilla ChatGPT approach) or be given additional dialogue context(Recursively summarizing approach), significantly reducing the number of tokens required to generate a response.

Objectives for the Response. The goal is to optimize the response generation such that:

(1) **Coherence:** R_k should exhibit continuity with previous conversational exchanges, avoiding contradictions or abrupt topic shifts.

(2) **Relevance:** R_k must directly address the user query Q_k while incorporating relevant prior context.

(3) **Efficiency:** By leveraging M_k, the system minimizes the need to process the entire conversation history and reduces cascading errors.

3.2 Design Overview: Sliding Window Mechanism

The proposed solution to the problem formulated in Section 3.1 is the **Sliding Window Summarization (SWin)** framework. This memory architecture dynamically updates long-term conversational memory through overlapping windows of dialogue and structured summarization prompts. **SWin** is designed to address the limitations of full-history replay, retrieval-based memory, and recursive summarization by providing an efficient, modular, and cognitively plausible mechanism for sustaining coherence in extended dialogues.

3.2.1 System Architecture

An overview of the system architecture is illustrated in Figure 3.1. The design consists of three key modules:

(1) **Input Processor:** Segments incoming dialogues into manageable windows for subsequent summarization.

(2) **Memory Module:** Maintains a compact and updated representation of the conversation using overlapping window summaries guided by the memory prompt P_{mem}.

(3) **Response Generator:** Produces the system's reply to the latest user query by conditioning the LLM on both the updated memory and the response prompt P_{resp}.

3.2.2 Workflow

The overall workflow of SWin is depicted in Figure 3.2.2. The process follows four main stages:

(1) **Problem Definition:** The system receives a new user query Q_k in the context of an ongoing dialogue.

(2) **Inputs:** The latest dialogue turn T_k and the current memory state M_{k-1} are taken as inputs.

(3) **Processing:** The Input Processor segments the dialogue into overlapping windows. The Memory Module then applies f_{update} with prompt P_{mem} to generate an updated memory M_k. In parallel, the Response Generator applies f_{resp} with prompt P_{resp} to produce a contextually coherent response.

(4) **Output:** The updated memory M_k and the system response R_k are delivered, completing the cycle for turn k.

3.2.3 Design Motivations

The design choices of SWin are motivated by both practical efficiency and theoretical considerations:

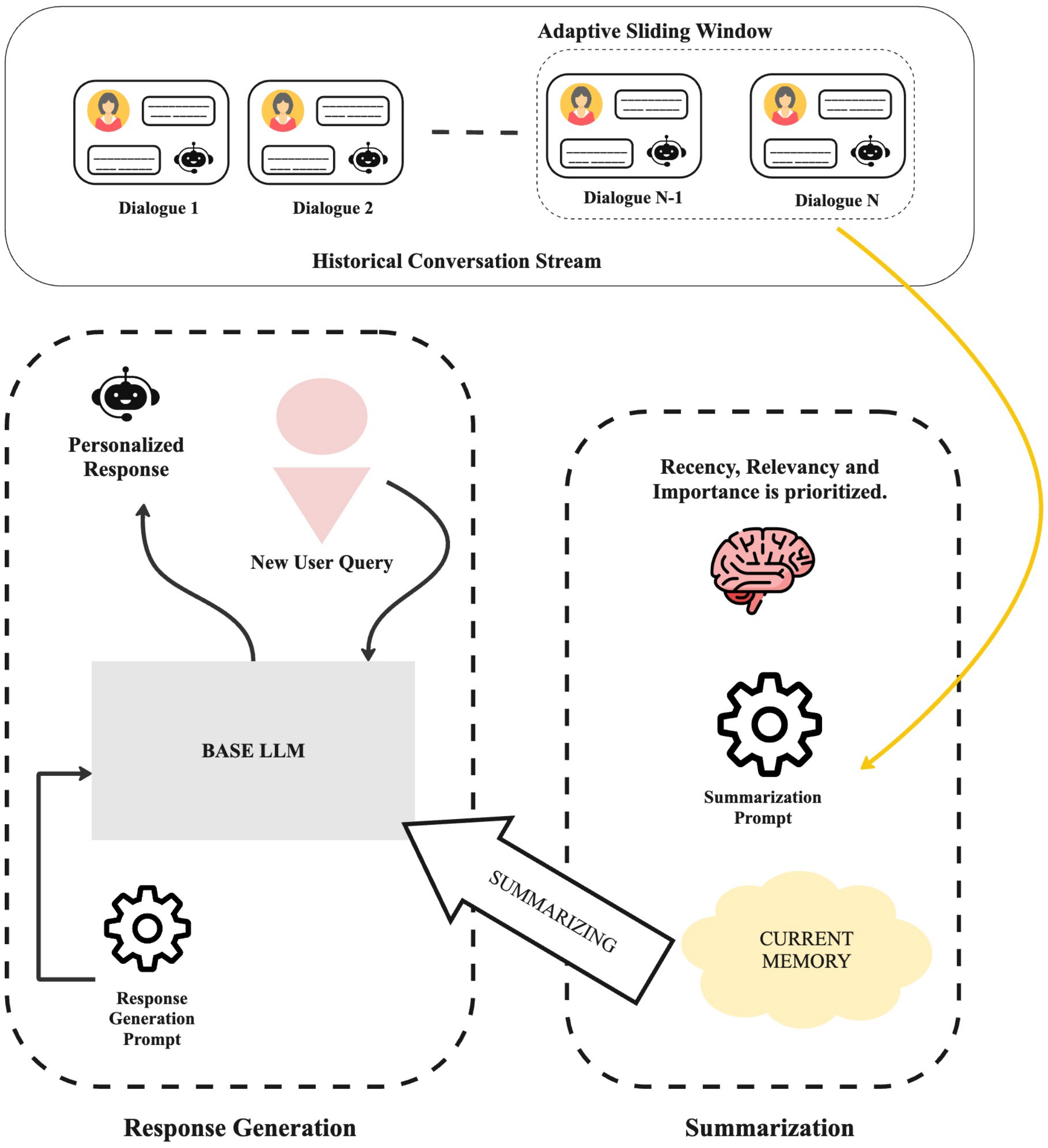


Figure 3.1: System architecture of the proposed Sliding Window Summarization (SWin) framework.

- **Sliding Window:** Processing the entire dialogue at every turn is computationally expensive and prone to context overflow. By contrast, the sliding window approach leverages only the most recent segments in combination with compact memory, drastically reducing token usage while preserving relevance. This principle aligns with the design of *Generative Agents: Interactive Simulacra of Human Behaviour* [8], where agents retrieve memories selectively based

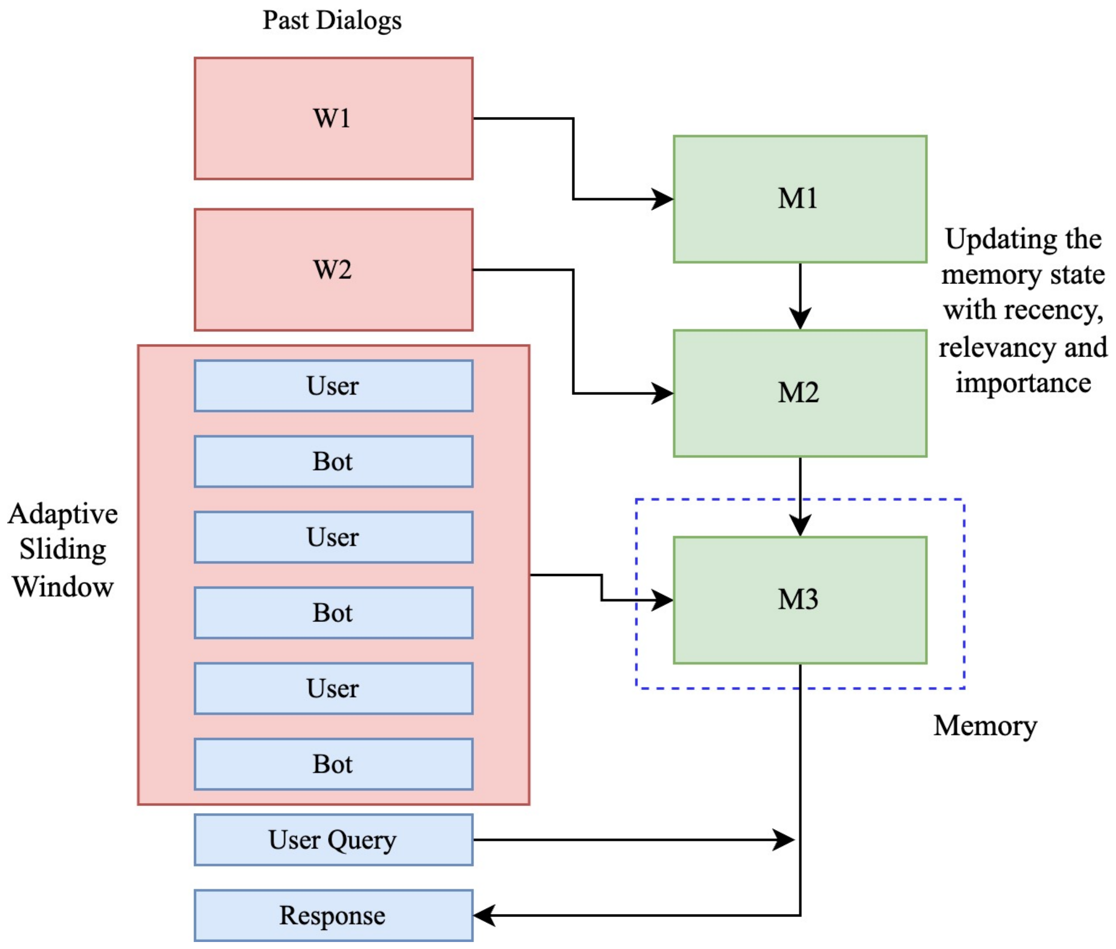


Figure 3.2: Workflow of the SWin process from problem input to response generation and memory update.

on recency, relevance, and importance to maintain coherence while avoiding the inefficiencies of replaying the full history.

- **Summarization-Based Memory:** Retrieval-based approaches require building and maintaining external indexes, which are computationally expensive and susceptible to retrieval errors when relevant but non-salient details are overlooked. Summarization-based memory, guided by prompt P_{mem}, generates structured, condensed representations that are directly aligned with the needs of the dialogue system and do not require external infrastructure and help keep the system lightweight for edge case implementation and resource-constrained environments.

3.3 Technical Details and Algorithms

In this section, we elaborate on the core technical mechanisms of the proposed Sliding Window Summarization (SWin) framework. We present the inputs, preprocessing strategies, algorithmic design, and the rationale behind each architectural choice. The emphasis is placed on how dialogue data is represented, segmented, and prepared for subsequent memory updates and response generation.

3.3.1 Inputs and Preprocessing

The SWin framework operates on multi-turn conversations where coherence depends on capturing both the immediate dialogue context and long-term dependencies. To this end, we define the formal inputs and preprocessing pipeline as follows.

Input Definitions

At dialogue turn k, the system receives the following inputs:

- **Conversation History:** A sequence of dialogue turns

  $$C_{k-1} = \{T_1, T_2, \ldots, T_{k-1}\},$$

  where each $T_i = (u_i, b_i)$ represents a user utterance u_i and corresponding system response b_i.

- **User Query:** The new input utterance from the user at time k, denoted as Q_k.

- **Memory State:** A compact representation of prior interactions up to turn $k-1$, denoted as M_{k-1}.

- **Window Size:** A predefined parameter w specifying the number of dialogue turns to include in each sliding window segment.

- **Prompts:** Two structured prompts guide the system behaviour:

$$P_{\text{resp}} : \text{prompt template for response generation,}$$
$$P_{\text{mem}} : \text{prompt template for memory summarization.}$$

Segmentation into Sliding Windows

The conversation history is segmented into overlapping windows of size w to balance coherence and efficiency. Formally, the j-th window is defined as:

$$W_j = \{T_j, T_{j+1}, \ldots, T_{j+w-1}\}, \quad j = 1, \ldots, k - w + 1.$$

To prevent discontinuities between consecutive windows, we employ an overlap parameter $\delta \in [1, w - 1]$, such that:

$$W_{j+1} = \{T_{j+\delta}, \ldots, T_{j+\delta+w-1}\}.$$

This overlapping ensures that the salient context at the edge of one segment is preserved in the subsequent segment, reducing information loss during summarization.

Preprocessing with Summarization Prompts

For each window W_j, the system applies a summarization step using the prompt P_{mem} and M_k, the previous memory. The output is a condensed representation $\hat{W}_j$, which preserves factual and user-centric details while eliminating redundancies:

$$\hat{W}_j = f_{\text{summ}}(W_j, M_k, P_{\text{mem}}).$$

The latest condensed representation is used as the memory for the subsequent conversation.

Selection of LLMs

The LLM f_{resp} and f_{update} are chosen based on task-specific constraints:

- High-capacity models (e.g., GPT-4o, LLaMA-3-70B) are suited for f_{resp} to maximize reasoning and coherence.
- Lightweight or distilled models (e.g., Flan-T5, Mistral-7B) can be employed for f_{update} since summarization is less demanding and more frequent. (This approach is intended for real-time production. However, we do not apply this methodology in our experiments, in order to ensure fair comparisons with other methods in terms of resource usage and model capacity. In the case study, this methodology was employed and validated through varied evaluations, which demonstrated optimal results with minimal resource usage.)

This separation of roles enables computational efficiency without sacrificing performance in the critical task of response generation.

Justification of Preprocessing Design

- **Segmentation:** Directly feeding the entire history into the LLM leads to quadratic scaling of computation with sequence length. Segmentation into windows bounds the input size while preserving continuity through overlaps.
- **Summarization:** Unlike raw replay, summarization ensures that memory storage grows sub-linearly with conversation length, preventing context overflow.
- **Prompt-Guided Preprocessing:** P_{mem} enforces faithful, structured, and concise summaries, while P_{resp} tailors the LLM output toward coherence and task relevance.
- **Role Separation of LLMs:** Assigning distinct models to response and memory updates balances efficiency and accuracy, allowing scalability across long sessions.

3.3.2 Core Components

The Sliding Window Summarization (SWin) framework relies on three tightly integrated components: (i) the Summarization Module, (ii) the Memory Module, and (iii) the Response Generator. Together, these modules enable dynamic context compression, efficient memory management, and coherent response generation in multi-turn dialogue.

Summarization Module (LLM-SUM)

The Summarization Module employs a dedicated large language model f_{update} to condense dialogue windows into concise yet faithful representations. For each window W_j, a summary $\hat{W}_j$ is produced:

$$\hat{W}_j = f_{\text{summ}}(W_j, M_k, P_{\text{mem}}).$$

Where P_{mem} is a structured memory prompt that enforces key constraints such as factual fidelity, user-centric detail preservation, and conciseness and M_k is the previous memory.

Prompting Strategies. We adopt *chain-of-thought prompting* to encourage structured reasoning about the salience of dialogue content before summarization. In addition, *few-shot prompting* is used to provide exemplars of high-quality summaries, thereby calibrating the model's outputs toward consistent style and granularity.

Alternative Designs. Two primary alternatives were considered:

- **Extractive Summarization:** Selecting key sentences directly from W_j. While computationally cheaper, this approach tends to retain redundant or irrelevant utterances and fails to capture implicit user intents. **Advantages**: Computationally efficient and easier to implement; Maintains high fidelity to the original text, preserving factual details directly.

  Limitations: Often produces redundant or disjointed summaries, especially when the source contains repetitive or irrelevant sentences. This is especially problematic in conversational or multi-turn contexts where alignment with dialogue nuance matters; Fails to capture implicit user intents or underlying semantics—notably, it cannot infer purpose or paraphrase to convey deeper meaning.

- **Abstractive Summarization (adopted):** Generating new summaries conditioned on P_{mem}. This approach allows compression, paraphrasing, and abstraction, yielding compact yet expressive representations. Abstractive summarization generates completely new summary sentences, going beyond mere extraction to rephrase, condense, and synthesize content in a more expressive style.

Advantages: Compression and Paraphrasing: Produces more concise summaries that still retain meaning. It paraphrases and reorders information for coherence and brevity. Expressivity and Coherence: As compared to extractive methods, abstractive summaries are generally seen as more fluent and human-like. Capturing Implicit Intent: Abstractive systems can infer and express the why behind a conversation or document, not just the what, enabling a deeper alignment with user needs.

Thus, abstractive summarization with carefully crafted prompts was selected as the basis of SWin.

Memory Module

The Memory Module aggregates window summaries and maintains a persistent, compact state M_k across dialogue turns. At step k, the memory update is defined as:

$$M_k = g(M_{k-1}, \hat{W}_j),$$

Where $g(\cdot)$ denotes a function that incorporates the newly summarized content into the prior memory. To ensure recency and coherence, SWin employs overlapping windows, such that:

$$\hat{W}_j \cap \hat{W}_{j+1} \neq \emptyset.$$

This overlap ensures smooth transitions between consecutive summaries and reduces the risk of information loss at window boundaries.

Memory Capacity and Forgetting. While memory grows more slowly than raw dialogue history, unbounded accumulation can still lead to inefficiency. Therefore, we define a maximum capacity $|M| \leq \gamma$, where γ is a tunable parameter. If capacity is exceeded, older or less relevant elements are pruned according to a *recency-weighted forgetting rule*:

$$M_k = \{x \in M_{k-1} \cup \hat{W}_j : \text{score}(x) > \theta\},$$

where score(x) balances temporal recency and semantic salience, and θ is a pruning threshold.

Overlapping Windows. Without overlap, summarization at strict boundaries risks dropping key cross-turn dependencies. By contrast, overlapping ensures that user intentions and incomplete utterances (e.g., ellipses or pronoun references) are retained, supporting more coherent long-term memory.

Response Generator

At each turn k, the system generates a response R_k using the updated memory and the latest user query:

$$R_k = f_{\text{resp}}(Q_k, M_k, P_{\text{resp}}),$$

where f_{resp} denotes the response-generation LLM and P_{resp} is the structured response prompt.

Prompt Construction. The prompt P_{resp} concatenates three key elements:

(1) **Memory Context:** A condensed representation of prior interactions (M_k).

(2) **User Query:** The latest user utterance Q_k.

(3) **Instructional Template:** A system directive ensuring that the response remains coherent, factually consistent with memory, and aligned with dialogue goals.

Appendix A provides concrete examples of the prompt templates employed in this thesis.

Design Considerations. Alternative strategies, such as retrieval-augmented response generation, were considered but dismissed, as retrieval mechanisms introduce infrastructure overhead and are sensitive to retrieval errors. Instead, a memory-based response generation ensures a compact, deterministic, and directly interpretable workflow.

3.4 Algorithms and Pseudocode

3.4.1 Algorithms

To concretize the proposed design, this section presents the pseudocode for the Sliding Window Summarization (SWin) process and its subroutines. Rather than simply listing the algorithms, we provide commentary for each step to highlight the design motivations and trade-offs.

Algorithm 1: Sliding Window Summarization Process (SWin_Process)

Inputs:

- C_t: Full conversation history at time t, a sequence of turns $\{T_1, T_2, ..., T_n\}$
- Q_k: New user query at turn k
- j: Window size (number of dialogue turns per window)
- `LLM_SUM`: Summarization model
- `LLM_RESP`: Response generation model
- M_k: Memory state at turn k (initially empty or loaded from previous sessions)

Output: Response R_k to the query Q_k

Algorithm 1 SWin_Process

```
1:  Initialize memory $M_k \leftarrow \emptyset$    ▷ Start with no long-term memory, or load if available
2:  Initialize window list $W \leftarrow [\,]$    ▷ Stores overlapping dialogue windows
3:  Let $n \leftarrow$ length of $C_t$
4:  for $i \leftarrow 0$ to $n - j$ do
5:      $W_i \leftarrow \{T_i, T_{i+1}, ..., T_{i+j-1}\}$    ▷ Create overlapping window of size $j$
6:      Append $W_i$ to $W$
7:  end for
8:  for each window $W_i$ in $W$ do
9:      $S_i \leftarrow$ LLM_SUM($W_i, P_{\text{mem}}$)    ▷ Summarize window using memory prompt
10:     $M_k \leftarrow$ UpdateMemory($M_k$, $S_i$)    ▷ Incorporate summary into memory
11: end for
12: $R_k \leftarrow$ LLM_RESP($Q_k, M_k, P_{\text{resp}}$) ▷ Generate final response conditioned on memory and query
13: return $R_k$
```

Commentary. This algorithm illustrates the entire SWin pipeline. Overlapping windows ensure that context near window boundaries is preserved across summaries, mitigating loss of coherence. Memory is updated after each summarization step, rather than in bulk, so that relevant context can be progressively accumulated and pruned in real time. Finally, the response is generated using both the updated memory and the user's latest query, ensuring continuity and contextual relevance.

Algorithm 2: Memory Update Routine

Algorithm 2 UpdateMemory

```
1: function UPDATEMEMORY($M_k, S_i$)
2:     if $S_i \notin M_k$ then    ▷ Avoid duplication of summaries
3:         Append $S_i$ to $M_k$
4:     end if
5:     return $M_k$
6: end function
```

Commentary. The `UpdateMemory` subroutine maintains a compact state by ensuring that redundant summaries are not reintroduced. More advanced criteria can be applied here:

- **Salience filtering:** Only summaries with high relevance scores (semantic similarity to current query) are retained.
- **Capacity management:** When $|M_k| > \gamma$, pruning is triggered based on recency-weighted importance.
- **Conflict resolution:** If new summaries contradict older ones, rules (e.g., trust more recent information) can be applied.

This modularity allows UpdateMemory to adapt across different implementations.

Algorithm 3: Sliding Window Memory Update (Modular Integration)

Algorithm 3 SlidingWindowUpdate

```
1: function SlidingWindowUpdate($C_t, j, M_{k-1}$)
2:     Segment $C_t$ into overlapping windows $\{W_1, W_2, ..., W_m\}$
3:     for each $W_i$ do
4:         $S_i \leftarrow$ LLM_SUM($W_i, P_{\text{mem}}$)
5:         $M_k \leftarrow$ UpdateMemory($M_{k-1}, S_i$)
6:     end for
7:     return $M_k$
8: end function
```

Commentary. This generalized memory update highlights the modularity of the SWin framework:

- It can integrate with retrieval-based systems by replacing `UpdateMemory` with a hybrid retrieval-and-merge function.
- It can extend to long-context LLMs by bypassing summarization for short conversations while reverting to sliding windows when context exceeds the token limit.

- The modular structure allows interchangeable summarization models, adaptive window sizing, and pluggable forgetting strategies.

This design ensures SWin is not only effective in its base form but also flexible for future integration with evolving LLM architectures.

3.4.2 Pseudocode

As shown in Listing 3.1, the SWin method segments the conversation into multiple windows to be processed and used for summarization and response generation.

Listing 3.1: SWin Simple Representation in Python 3 (with prompts and type hints)

```python
\vspace{1em}

from typing import List, Dict, Any, Tuple

Turn = Dict[str, Any]              # e.g., {"user": str, "bot": str}
Summary = Dict[str, Any]           # e.g., {"text": str, "meta": Dict[str,
    Any]}
Memory = List[Summary]

def LLM_SUM(window: List[Turn], prompt: Dict[str, Any]) -> Summary:
    # Placeholder: your summarizer's .generate(...) goes here
    return {"text": "summary_text", "meta": {"len": len(window), "prompt
        ": prompt}}

def LLM_RESP(query: str, memory: Memory, prompt: Dict[str, Any]) -> str:
    # Placeholder: your responder's .generate(...) goes here
    return f"response_to({query})_using_{len(memory)}_mem_items"

def update_memory(memory: Memory, summary: Summary) -> Memory:
    # Minimal dedup by text; replace with your preferred strategy
    mem_texts = {m.get("text", "") for m in memory}
    if summary.get("text", "") not in mem_texts:
        memory.append(summary)
    return memory

def process_swin(conversation: List[Turn],
                 query: str,
                 window_size: int,
                 P_mem: Dict[str, Any],
                 P_resp: Dict[str, Any]) -> str:
    memory: Memory = []

    # Step 1: Create overlapping windows
    windows: List[List[Turn]] = []
    for i in range(len(conversation) - window_size + 1):
        window: List[Turn] = conversation[i:i+window_size]
        windows.append(window)

    # Step 2: Summarize each window (prompt-aware)
    for w in windows:
        summary: Summary = LLM_SUM(w, prompt=P_mem)
        memory = update_memory(memory, summary)

    # Step 3: Generate response (prompt-aware)
    response: str = LLM_RESP(query, memory, prompt=P_resp)
    return response
```

Example Usage:

Listing 3.2: SWin Example usage in Python 3 (with prompts and examples)

```python
# Minimal, runnable example using the prompt-aware API.

# 1) Toy conversation and query
conversation_history = [
    {"user": "Hello!", "bot": "Hi! How can I help you today?"},
    {"user": "Tell me about AI.", "bot": "AI stands for artificial
        intelligence..."},
    {"user": "What are its applications?", "bot": "AI is used in
        healthcare, finance, and more..."}
]
user_query = "Can you give me an example of AI in healthcare?"

# 2) Simple prompts (see Appendix for full templates)
P_mem  = {"role": "summarize", "constraints": ["faithful", "concise", "
    no hallucination"]}
P_resp = {"role": "assistant", "style": "helpful, concise", "grounding":
     "use memory if relevant"}

# 3) Very simple LLM stubs for illustration
class DummySUM:
    def generate(self, payload):
        w = payload.get("window", [])
        return {"text": f"Summary of {len(w)} turns", "meta": {"tokens":
            len(w)}}

class DummyRESP:
    def generate(self, payload):
        q = payload["query"]; m = payload["memory"]
        return f"Example: AI assists radiologists (used {len(m)} memory
            items)."

llm_sum  = DummySUM()
llm_resp = DummyRESP()

# 4) Call the SWin pipeline
response = process_swin(conversation_history, user_query,
                        P_mem=P_mem, P_resp=P_resp,
                        llm_sum=llm_sum, llm_resp=llm_resp,
                        window_size=3, stride_delta=1)

print("Bot Response:", response)
```

3.5 Complexity Analysis

We analyze computational cost in terms of the number of dialogue turns and the size of the context processed per step. Let:

- n: total number of turns in the history (prior to generating the current response),
- j: window size (turns per sliding window), with $j \ll n$,
- $w = n - j + 1$: number of overlapping windows created from the history,
- $C_{\text{tok}}(x)$: cost proportional to the number of tokens processed (input + output) of length x,

since LLM runtime and billing scale roughly linearly with tokens.

3.5.1 Sliding-Window Summarization (SWin)

Memory construction. We summarize each overlapping window once:

$$\text{Cost}_{\text{sum}}(n, j) = \sum_{i=1}^{w} C_{\text{tok}}(\Theta(j)) = \Theta\big(w \cdot j\big) = \Theta\big((n - j + 1) \cdot j\big) = \boxed{\Theta(n \cdot j)}.$$

Because j is fixed and small (e.g., $j = 3$), the total summarization cost grows *linearly* in n with a small constant.

Response generation. Let m be the number of summaries included in the response prompt (often capped at M for latency/price control). Then

$$\text{Cost}_{\text{resp}} = C_{\text{tok}}\big(\Theta(m)\big) = \Theta(m) = \Theta(\min\{w, M\}) = \Theta(M).$$

Thus, the end-to-end cost per conversation step is $\Theta(j) + \Theta(M)$, and the cumulative cost over a session is

$$\boxed{\Theta(n \cdot j)}.$$

3.5.2 Baseline A: Full-History Concatenation

At turn k, the model ingests the entire prior history of length $\Theta(k)$. The cumulative cost over n turns is

$$\text{Cost}_{\text{concat}}(n) = \sum_{k=1}^{n} C_{\text{tok}}(\Theta(k)) = \Theta\Big(\sum_{k=1}^{n} k\Big) = \boxed{\Theta(n^2)}.$$

3.5.3 Baseline B: Recursive Session Summarization

Recursive methods re-summarize accumulated memory with each new step (memory + new context). If the effective context length at step k is $\Theta(k)$:

$$\text{Cost}_{\text{RSum}}(n) = \sum_{k=1}^{n} C_{\text{tok}}(\Theta(k)) = \boxed{\Theta(n^2)}.$$

3.5.4 Comparison

$$\boxed{\text{SWin: } \Theta(n \cdot j) \quad \text{vs.} \quad \text{Baselines: } \Theta(n^2)} \quad \Rightarrow \quad \text{SWin is asymptotically cheaper when } j \ll n.$$

While current experiments utilize a general-purpose summarizer, it is reasonable to assume that deployment in production systems would favor a domain-specific summarization model, as such models are typically more efficient, both in terms of speed and resource consumption.

3.5.5 Token-Cost Model

Let each turn average $\bar{t}$ tokens. Then:

$$\text{SWin summarization: } \approx w \cdot \alpha \cdot (j\bar{t}) \;\approx\; \alpha(n-j+1)j\bar{t},$$

$$\text{Vanilla cumulative: } \approx \sum_{k=1}^{n} \alpha \cdot (k\bar{t}) \;=\; \alpha\bar{t}\,\frac{n(n+1)}{2},$$

RSum cumulative: same order as Vanilla, often with larger constant.

Here α bundles tokenizer inflation, prompt scaffolding, and output tokens. With $j = 3$ and $n = 100$, SWin processes on the order of $300\bar{t}$ tokens vs. $\sim 5{,}050\bar{t}$ for quadratic baselines.

3.5.6 Visualization

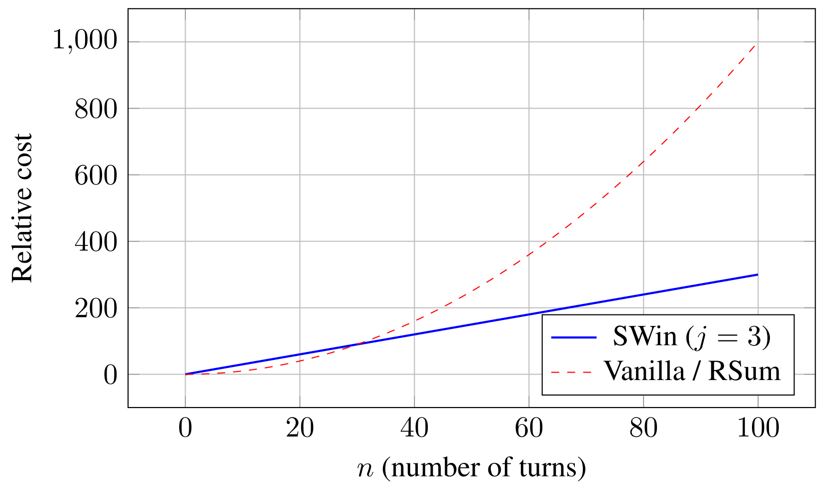


Figure 3.3: Asymptotic comparison of SWin ($\Theta(nj)$ with $j = 3$) versus quadratic baselines ($\Theta(n^2)$).

3.6 Design Alternatives and Justifications

In developing the SWin framework, several alternative designs for memory management and long-context handling were considered. While each approach has its merits, they also present critical drawbacks that limit their effectiveness in extended, coherent dialogue. This section discusses these alternatives and explains the rationale behind adopting the sliding window summarization strategy.

Full-History Concatenation. A straightforward solution is to concatenate the entire dialogue history and pass it to the language model at each turn. (also referred to as Vanilla ChatGPT in further Experiments) Although this guarantees complete access to context, it is highly inefficient in both token usage and computational cost. As the dialogue lengthens, the prompt quickly exceeds the context window limitations of most LLMs, requiring expensive truncation or additional memory management heuristics. Moreover, this method introduces redundancy by repeatedly reprocessing the same tokens at every conversational turn, thereby compounding latency and resource demands.

Retrieval-Based Memory. Another popular alternative involves retrieval-augmented methods, where past utterances or summaries are stored in a memory index and selectively retrieved based on relevance to the current query. While this reduces token load, it introduces non-trivial indexing complexity and is prone to missing contextually important but lexically distant information. For example, a subtle reference in the current turn may not trigger retrieval of a semantically relevant but lexically dissimilar earlier passage, leading to coherence loss. Additionally, retrieval-based designs require careful tuning of embedding models and similarity thresholds, which adds overhead and reduces portability across domains.

Session-Based Summarization. Session-level summarization techniques produce condensed representations after each interaction block. This strategy captures high-level context while compressing token usage. However, it is less effective for dynamic, mid-session coherence, as important details introduced earlier within a session may be inaccessible until a summary is generated. Consequently, the system may overlook evolving conversational shifts or fail to respond appropriately to intra-session references, thereby undermining continuity.

Justification for SWin. The sliding window summarization framework was selected because it balances efficiency, coherence, and cognitive plausibility. By incrementally summarizing overlapping windows of dialogue, SWin ensures that recent context is always retained while older interactions are abstracted into concise summaries. This mechanism reduces redundant token processing and scales linearly with dialogue length, outperforming full-history approaches in efficiency. At the same time, it avoids the retrieval errors of index-based methods and provides finer granularity than session-level summarization, ensuring responsiveness to mid-session developments. Thus, the design is cognitively inspired, reflecting how human interlocutors remember conversations: maintaining focus on the most recent exchanges while retaining abstracted knowledge of earlier context. This combination of computational efficiency, contextual fidelity, and psychological grounding provides a strong justification for the adoption of SWin as the central methodology in this thesis.

3.7 Open Discussion about Design Choices

The proposed Sliding Window Summarization (SWin) framework is intentionally designed with a high degree of modularity. Each component of the pipeline: summarization, memory management, and response generation, can be replaced or extended without disrupting the overall architecture. For instance, the summarization module inherently acts as an orthogonal plugin: abstractive summarizers can be substituted with extractive methods, transformer-based models can be interchanged with lighter-weight alternatives, and prompt designs can be adapted as new paradigms emerge. Similarly, the parameter governing window size is configurable, allowing the system to be tuned for different computational budgets and dialogue lengths. This adaptability ensures that the framework remains flexible in accommodating evolving requirements and diverse deployment environments.

Beyond modularity, the SWin framework offers potential for integration with emerging paradigms in LLM research. One natural extension lies in coupling sliding window summarization with retrieval-augmented generation (RAG), where external knowledge bases could be queried in parallel to contextual memory updates. Such integration would enhance factual accuracy and reduce the likelihood of hallucination in knowledge-intensive tasks. Another avenue involves alignment with streaming LLMs, which process tokens sequentially in real time; in this setting, SWin could act as an intermediate summarization layer to stabilize and compact long interaction histories. Furthermore, the design is not inherently restricted to text-only inputs: multimodal dialogue systems that incorporate visual, auditory, or symbolic modalities could employ the same sliding window and summarization principles to manage extended interactions across multiple data streams.

Despite these strengths, several theoretical limitations warrant critical reflection. First, the framework retains a dependence on a fixed window size, which, while adjustable, may not always capture the dynamic nature of human conversation—where some turns require dense recall and others can tolerate greater abstraction. Second, the quality and reliability of the summarizer directly constrain system performance: errors introduced at the summarization stage may propagate across subsequent windows, resulting in compounding loss of coherence or semantic drift. Third, although modular, the approach remains vulnerable to trade-offs between efficiency and fidelity; overly aggressive compression risks discarding contextually salient details, while overly conservative summarization undermines efficiency. These limitations highlight both the necessity of careful parameterization and the opportunity for future enhancements, such as adaptive window sizing or hybrid summarization strategies.

Chapter 4

Experiments

4.1 Introduction

The purpose of this chapter is to establish the experimental framework through which the proposed Sliding Window Summarization (SWin) framework is evaluated. Building on the problem formulation and design introduced in the preceding chapter, this section delineates the methodology used to assess the effectiveness, efficiency, and robustness of the proposed approach. In particular, the experimental setup is designed to benchmark SWin against a range of representative baseline models, using both automatic and human-aligned evaluation metrics, across a dataset that faithfully reflects the challenges of multi-session dialogue.

To this end, the chapter is organized around four key elements. First, the selection of baseline models is justified, with attention to their relevance, representativeness, and complementary strengths in dialogue summarization and memory integration. Second, the dataset used for evaluation is described, with emphasis on its suitability for testing long-term conversational coherence. Third, the evaluation metrics are formally introduced, including both linguistic overlap measures and efficiency-oriented metrics that capture the trade-off between fidelity and computational cost. Finally, the experimental setup is detailed, covering implementation considerations, parameter choices, and computational resources. Together, these components form a comprehensive evaluation framework that supports rigorous comparison of the proposed method against established alternatives.

4.2 Baseline Models

To evaluate the effectiveness of the proposed Sliding Window Summarization (SWin) framework, we compare against a set of representative baseline models that capture the major design paradigms for long-term dialogue modeling. The chosen baselines include **Recursive Summarization**, **MemoChat**, **MemoryBank**, and **ChatGPT-RSum**. These systems were selected because they represent complementary strategies for managing conversational history and are either publicly available or well-documented for reproducible comparison.

4.2.1 Motivation for Baseline Selection

Recursive Summarization. This approach represents one of the earliest and simplest strategies for compressing dialogue history by recursively summarizing past turns. It provides a natural point of comparison for SWin, as both frameworks rely on abstractive summarization but differ in how they manage temporal information.

MemoChat. MemoChat adopts an episodic memory paradigm, where salient pieces of information are distilled into structured memory slots. Unlike recursive summarization, which compresses history into a single evolving summary, MemoChat retains discrete pieces of information for retrieval. It serves as a strong baseline for evaluating whether structured memory representations improve coherence across sessions.

MemoryBank. MemoryBank maintains a dynamic external memory that grows incrementally and supports explicit memory updates. It illustrates an alternative to compression-based methods by prioritizing retention over abstraction. This makes it a relevant comparison point for assessing trade-offs between memory capacity, coherence, and efficiency.

ChatGPT-RSum. Finally, we include ChatGPT-RSum as a proprietary state-of-the-art system that employs recursive summarization on top of large-scale proprietary LLMs. This model provides an upper-bound baseline, allowing us to benchmark the proposed method against an industrial-strength closed-source system.

Exclusions. Other promising approaches were not selected for practical reasons. Retrieval-based memory systems, while effective for knowledge-intensive tasks, are poorly aligned with the MSC dataset and require additional infrastructure (e.g., indexing pipelines). Long-context transformer variants and sliding-attention models, although theoretically appealing, are computationally infeasible within our resource constraints and lack open-source implementations compatible with the experimental setup. These exclusions ensure that our evaluation remains both fair and reproducible.

4.2.2 Summary of Baselines

Table 4.1 provides an overview of the baseline models, comparing their model type, context management strategy, computational requirements, and reported strengths and weaknesses.

Table 4.1: Summary of baseline models for comparison.

Model	**Type**	**Context Strategy**	**Resource Requirements**	**Strengths / Weaknesses**
MemoChat	Episodic memory with discrete slots	Extracts and stores salient events for retrieval	Moderate; requires additional memory indexing	Preserves important details; risk of losing global coherence
MemoryBank	Dynamic external memory system	Incrementally stores dialogue states without aggressive compression	Higher memory and compute demand	Retains fine-grained context; susceptible to memory bloat and inefficiency
ChatGPT-RSum	Proprietary LLM with abstractive summarization and recursive updates	Compresses full dialogue history into iterative summaries	High; requires access to proprietary API and large compute	Simple and scalable; prone to error accumulation and semantic drift, costly, and less transparent

4.2.3 Discussion

The baselines employed in this study represent complementary philosophies of managing conversational history in long-term dialogue systems. Broadly, these methods can be categorized into context-only, retrieval-based, and memory-based approaches.

Context-only approaches constitute the most straightforward strategy, where the dialogue model receives the concatenated history and current utterances as input. (***Vanilla-ChatGPT***) Although this method ensures maximal fidelity to past interactions, it quickly becomes infeasible as dialogue length increases, given the quadratic scaling of transformer-based architectures.

Retrieval-based approaches attempt to address this inefficiency by filtering and selecting only the most salient utterances from history. For example, BM25 [49] and Dense Passage Retrieval [50] provide ranking-based mechanisms for identifying relevant dialogue snippets, which are then reintroduced into the model's context. While retrieval enhances scalability, it risks fragmenting dialogue coherence if the retriever fails to capture implicit dependencies across sessions.

Memory-based approaches abstract conversational history into structured or compressed forms that can be iteratively updated. Within this family, Recursive Summarization exemplifies efficiency through continuous compression, but this efficiency comes at the cost of compounding summarization errors over time. ChatGPT-RSum operates as an upper-bound benchmark: while proprietary and resource-intensive, it illustrates the performance ceiling of recursive summarization when combined with industrial-scale LLMs. MemoChat employs a topic-structured episodic memory, selectively retaining summaries aligned to dialogue themes, balancing information fidelity with reduced redundancy. MemoryBank, by contrast, adopts a global memory structure informed by human-like forgetting dynamics, prioritizing detail preservation but incurring significant computational demands.

Taken together, these baselines enable a systematic positioning of SWin within the broader design space of conversational memory strategies. By juxtaposing efficiency-oriented recursive methods, retrieval-enhanced selection, memory-intensive compression, and state-of-the-art proprietary systems, the evaluation offers a balanced and representative landscape against which to assess the contributions of the proposed framework.

4.3 Dataset

To evaluate the proposed framework, we employ the Multi-Session Chat (MSC) dataset [9], a benchmark specifically designed to advance the study of long-term, open-domain conversational systems. Unlike earlier dialogue corpora such as DailyDialog or PersonaChat, which primarily contain short, single-session interactions (typically 2–15 turns), MSC emphasizes multi-session continuity. It comprises human–human conversations spanning up to five sessions, with each session building upon knowledge accumulated in previous ones. This structure simulates the dynamics of real-world interactions, where interlocutors reference shared history, recall facts, and maintain consistency across extended time spans.

An important feature of MSC is that participants were instructed to assume diverse personas while engaging in conversations. This design choice introduces variability in speaking styles, interests, and long-term goals, which increases the difficulty of maintaining coherence. For example, a conversation may begin with two speakers discussing weekend plans in Session 1, only for Session 3 to reference those earlier commitments or introduce subtle changes in persona-related details. Such cross-session references make the dataset particularly well-suited for evaluating memory-augmented models like our sliding window summarization framework.

Another distinctive characteristic of MSC is the inclusion of annotated summaries for each session. These summaries highlight key facts and events, providing a ground truth for models tasked with compressing and recalling past dialogue. In our evaluation, these summaries serve as both a benchmark and a structural cue for testing the efficiency of summarization-based memory modules. On average, four-session dialogues in MSC contain 53 utterances, creating a substantial context window for systems to process.

Table 4.2 presents the dataset statistics across multiple sessions. Here, Session ID i corresponds to a dialogue instance that has $i - 1$ preceding sessions of conversational history. The columns include the number of episodes, the number of test utterances, and token-based statistics. Beyond the original dataset statistics, we augment the analysis by introducing a new metric, Average Tokens per Turn, computed using embeddings generated by the GPT-4o-mini model. This metric quantifies the typical token load per conversational exchange, thereby offering a proxy for computational resource demands. Such granularity enables us to compare not only the accuracy of baseline and proposed methods but also their efficiency in terms of token usage—a critical consideration for scalable dialogue systems.

We chose the Multi-Session Chat (MSC) dataset because it uniquely captures long, multi-turn, multi-session dialogues with evolving user personas. Most other datasets cover single-session chats, which don't stress-test memory. MSC forces models to recall facts across days and sessions, making it ideal to benchmark memory mechanisms like SWin.

Table 4.2: Multi-Session Chat Dataset Statistics

Session ID	**Episodes**	**Test Utts.**	**Ave. Tokens per turn**	**Ave. Tokens**	**Max. Tokens**
Session 2	501	5939	27.25	444.84	951
Session 3	501	5924	29.05	810.19	1733
Session 4	501	5940	30.53	1195.08	2234
Session 5	501	5945	32.08	1598.78	2613

Dataset Limitations and Implications While MSC represents a significant step forward for multi-session dialogue research, it is not without limitations. First, the dataset is domain-constrained: conversations were generated in a crowd-sourced setting with pre-assigned personas, which may not fully capture the spontaneity and diversity of real-world multi-session dialogues. Second, its size (approximately 500 episodes per session tier) is modest relative to large-scale dialogue corpora, potentially limiting model generalization. Third, MSC is monolingual (English-only), which restricts its applicability for multilingual or cross-lingual dialogue systems.

These limitations have direct implications for the generalizability of experimental findings. While MSC is highly effective for controlled benchmarking of memory and summarization strategies, results obtained may not immediately transfer to specialized domains (e.g., medical, legal) or languages with different conversational norms. Consequently, we interpret MSC as a proving ground for testing the efficiency and coherence of long-term memory modules, while acknowledging the need for future evaluation on more diverse and larger-scale datasets.

4.4 Evaluation Metrics

To comprehensively evaluate the performance of the proposed framework, we employ a suite of automatic and LLM-based metrics that jointly assess the *quality*, *semantic fidelity*, and *efficiency* of generated responses. Building on prior work [6,9], we incorporate standard lexical overlap measures, embedding-based similarity metrics, and human-aligned evaluations, together with a computational efficiency indicator. This combination ensures that the assessment captures not only linguistic correctness but also scalability for real-world deployment.

4.4.1 Automatic Metrics

BLEU. The Bilingual Evaluation Understudy (BLEU) score [24] is an n-gram precision-based metric widely used in machine translation and dialogue systems. For a candidate response C and reference R, BLEU-n is computed as:

$$\text{BLEU-}n = BP \cdot \exp\left(\sum_{i=1}^{n} w_i \log p_i\right),$$

where p_i denotes the modified i-gram precision, w_i is a uniform weight, and BP is the brevity penalty. We report BLEU-1 and BLEU-2 to capture unigram adequacy and bigram fluency.

ROUGE. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measures the recall of overlapping n-grams between generated and reference responses. ROUGE-n is defined as:

$$\text{ROUGE-}n = \frac{\sum_{\text{gram}_n \in R} \min(\text{Count}_C(\text{gram}_n), \text{Count}_R(\text{gram}_n))}{\sum_{\text{gram}_n \in R} \text{Count}_R(\text{gram}_n)}.$$

We report ROUGE-1 and ROUGE-2 to capture lexical overlap at different granularities.

F1 Score. Following prior works [6], we also compute F1 score at the token level, which balances precision and recall between candidate C and reference R:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

This metric emphasizes coverage of reference tokens while penalizing extraneous content.

BERTScore. To capture semantic similarity beyond surface form, we compute BERTScore [25], which aligns tokens in candidate and reference sentences via contextual embeddings. Given embeddings $\mathbf{h}_C$ and $\mathbf{h}_R$, the precision, recall, and F1 are defined as:

$$P = \frac{1}{|C|}\sum_{c\in C}\max_{r\in R}\cos(\mathbf{h}_c,\mathbf{h}_r),\quad R = \frac{1}{|R|}\sum_{r\in R}\max_{c\in C}\cos(\mathbf{h}_r,\mathbf{h}_c),$$

$$\text{BERTScore-F1} = \frac{2PR}{P+R}.$$

This metric evaluates semantic alignment between candidate and reference responses.

4.4.2 LLM-Based Evaluation

GPT-4 Judgments. In addition to automatic metrics, we employ GPT-4 as an evaluator to provide human-aligned judgments of coherence, relevance, and fluency. This approach has been shown to approximate expert annotations at scale, complementing token-level metrics, which often fail to capture pragmatic appropriateness. We follow established protocols by presenting GPT-4 with both candidate and reference responses in randomized order and recording its preferences.

4.4.3 Efficiency Metric

Token Usage. To assess computational efficiency, we introduce the *average tokens used per session*, reported in Table 5.1. For a model M and dialogue session S, the metric is defined as:

$$\text{AvgTokens}(M) = \frac{1}{|S|}\sum_{i=1}^{|S|}\text{Tokens}(M, S_i),$$

where $\text{Tokens}(M, S_i)$ counts the total input and output tokens consumed by M at turn i. This measure directly reflects inference cost, latency, and deployability in real-world settings. Unlike prior works focusing solely on response quality, our evaluation explicitly balances performance against efficiency.

4.4.4 Prompt Examples Across Baselines

Finally, to illustrate the evaluation context, we provide representative prompt examples across baselines. Table 4.3 shows how different memory strategies produce distinct input prompts:

- **Recursive Summarization:** concatenates compressed summaries from previous turns.
- **MemoChat:** retrieves structured "memos" relevant to the query.
- **MemoryBank:** incorporates reinforcement and forgetting mechanisms to weight memory segments.
- **SWin (ours):** slides overlapping windows over dialogue history, summarizing incrementally.

Table 4.3: Representative prompt examples across baselines, illustrating how different memory strategies construct inputs for the same dialogue context.

Baseline	**Example Prompt (Truncated)**
Recursive Summarization	"[Summary of previous sessions: User discussed travel plans and budget] User: Where should I go in Europe?"
MemoChat	"[Memory bank: likes historical sites; prefers affordable travel; past mention of Spain and Italy] User: Where should I go in Europe?"
MemoryBank	"[Explicit retrieved facts: User interested in history; budget under \$2000; considering Spain] User: Where should I go in Europe?"
Sliding Window Summarization (SWin - *ours*)	"[Window summary: User previously discussed budget constraints; interested in cultural cities; last turn was about comparing Spain vs. Italy] User: Where should I go in Europe?"

For example, given the dialogue snippet:

> *User: I just started college, and it feels overwhelming. Bot: That sounds like a big transition. What's been the hardest part so far?*

Recursive Summarization may prompt with: *"Summary so far: User started college and finds it overwhelming. Generate a supportive response to the latest query."*

By contrast, SWin provides: *"Window summary: User recently began college; discussed stressors in first semester. Latest input: feels overwhelmed. Generate contextually coherent response."*

These differences highlight how baselines manage context differently, which directly impacts both performance and efficiency metrics.

4.5 Experimental Setup

All experiments were conducted in a controlled computational environment to ensure reproducibility and comparability across baselines. Most experiments were conducted using the OpenAI API, Alibaba Cloud, and HuggingFace Inference Client, accessed from a MacBook Pro equipped with an Apple M1 Pro processor and 16 GB of RAM. Since model inference was executed through the API, local hardware specifications did not affect model performance but were sufficient to manage preprocessing, prompt construction, and evaluation pipelines. The implementation environment was based on Python 3.10, with dependencies managed via OpenAI and the transformers libraries. Custom scripts were developed to standardize prompt formatting, log API calls, and collect evaluation results across multiple runs.

For hyperparameters, the sliding window size was fixed at 6 turns, with an overlap of 2 utterances between windows. This configuration was selected to balance context preservation against computational efficiency, based on ablation testing runs that demonstrated diminishing returns with larger window sizes. Summarization was performed using gpt-3.5-turbo as the default summarizer, owing to its favourable balance between cost and performance. Token limits followed the default API specifications (4,096 tokens for gpt-3.5-turbo), and temperature was fixed at 0.0 across all generations to maintain consistency.

Several experimental constraints were imposed to ensure comparability and feasibility. Larger models such as GPT-4 were not adopted for the main experiments due to cost considerations and the need to align with practical deployment scenarios. Similarly, hyperparameters such as token budget, temperature, and top-p sampling were standardized across all baselines so that observed performance differences could be attributed primarily to memory management strategies rather than stochastic variations in generation. This setup reflects a pragmatic evaluation framework: while relying entirely on API-based inference, it preserves rigor by fixing parameters and ensuring reproducibility across experimental runs.

The hardware platform for testing the open-source models consisted of a NVIDIA A100 GPU with 40 GB of memory, paired with dual Intel Xeon processors and 512 GB of system RAM. This configuration enabled parallel execution of model inference while accommodating the memory demands of large transformer-based models. On the software side, experiments were implemented in Python 3.10 using the PyTorch 2.0 framework, with Hugging Face's Transformers library serving as the backbone for model loading and tokenization. Prompt orchestration and evaluation scripts were managed via custom utilities built on top of this framework. For GPT-based evaluations, the OpenAI API was accessed under controlled rate limits to maintain consistency across runs. Finally, token budgets for summarization and generation were standardized so that efficiency metrics, such as average tokens used (introduced in Table 5.1), would reflect only the architectural differences in context handling rather than unequal resource allocation. Through these design choices, the experimental setup aims to isolate the effects of memory management strategies while maintaining a fair and computationally realistic evaluation protocol.

Chapter 5

Results

5.1 Introduction

This chapter presents the empirical findings obtained through a systematic evaluation of the proposed Sliding Window Summarization (SWin) framework. The objective is to assess its effectiveness in sustaining coherent multi-session dialogues while maintaining computational efficiency. To ensure a comprehensive analysis, two complementary perspectives of evaluation are adopted: automatic evaluation metrics and LLM-based evaluation.

Automatic metrics provide standardized, reproducible measurements grounded in established benchmarks, such as BLEU, BertScore, F1, and token efficiency statistics. These metrics allow for an objective comparison against baseline models, quantifying improvements in coherence, contextual alignment, and resource utilization. However, while automatic metrics are widely accepted in the field, they are often insufficient for fully capturing the nuanced qualities of conversational coherence and user-centred interaction.

To address these limitations, an additional layer of LLM-based evaluation is employed. Recent advances have demonstrated the potential of large language models not only as dialogue agents but also as evaluators capable of judging conversational consistency, relevance, and human-likeness. In this study, LLM evaluators are tasked with comparing responses across baselines and SWin outputs, providing an interpretive dimension that complements the purely statistical perspective of automatic measures. This approach captures subtleties such as the preservation of user preferences, avoidance of contradictions, and natural progression of dialogue, aspects that are often underrepresented in traditional benchmarks.

Together, these two ways of experimentation enable a more rigorous and multifaceted evaluation of the proposed framework. The subsequent sections first report on the automatic results, followed by LLM-based evaluations, before synthesizing the findings to draw broader conclusions regarding the advantages and limitations of the SWin approach.

5.2 Automatic Evaluation

Table 5.1 presents a comparative analysis of token usage across different approaches, emphasizing the efficiency gains of our proposed framework. Our method achieves a 25.31% reduction in token consumption compared to the recursively summarizing approach and a 30.79% reduction relative to Vanilla ChatGPT. The table reports the average token usage per response generation at the end of the third session, a stage where memory accumulation and retrieval efficiency significantly influence performance.

For evaluation, we computed token usage based on the Memory Management and Response Generation functions outlined by Wang et al. [6]. Their method incorporates Dialogue Context C_t during response generation and maintains a conversation history H_i for summarization, leading to higher token consumption. However, our analysis reveals that this additional context does not significantly improve the semantic quality of generated responses. In contrast, SWin selects and summarizes relevant information, eliminating unnecessary token overhead while preserving coherence.

For the Vanilla ChatGPT baseline, we consider a naïve concatenation strategy, appending previous dialogue turns directly to the input without optimization. This approach, while maintaining full historical context, results in excessive token usage and a decline in response efficiency. These findings underscore the effectiveness of the SWin mechanism in optimizing memory management for long-term conversational AI. This balance between efficiency and coherence makes SWin highly scalable for real-world applications, particularly in resource-constrained environments such as healthcare and customer support.

Table 5.1: Average Tokens Used by Memory Modules

Memory Module	**Ave. Token usage**
SWin (Ours)	**256.02**
RSum	342.79
Vanilla ChatGPT	369.97

We introduce a dynamic memory updating approach using previous conversational data by processing dialogues within overlapping segments (windows) of a fixed size. This method is particularly well-suited for long-term conversation tasks and conversations where user information needs to be stored and retained effectively. This method ensures that only the most relevant and recent information is maintained, while older, less relevant data is effectively summarized or discarded.

SWin is inspired by the SSN-DM approach used in summarizing long documents with BERT models [23], and the prompt design is inspired by the state-of-the-art prompt engineering methods [8], [11]. In our approach, summaries are dynamically updated on a sliding-window basis, with the memory state capturing essential details from prior windows. Each window processes only a small part of the conversation, and the memory is updated incrementally. This prevents global errors from cascading across the entire context, and the overlap allows for reiteration and correction if errors do occur.

As shown in Table 5.2, SWin performs well in terms of F1 scores and achieves competitive results with the RSum method on BScore and BLEU-1. However, we observe a drop in BLEU-2 scores, likely due to the LLM's tendency toward verbosity, paraphrasing, or BLEU score's sensitivity to word order and small connecting words. Upon further analysis, we find that, while the generated responses differ in structure and phrasing, their semantic meaning remains intact. For example, in response to an MSC dataset reference: *"Oh man, I feel that. I can't sleep unless I make sure my teeth are clean, no matter how tired I am. I'd wake up in the middle of my sleep if I had to, just to brush my teeth. When are you moving out?"* SWin generates Output: *"I totally get that, and I'm right there with you. It's not the most fun experience, but it's good to know we're both taking care of our teeth. Fingers crossed that we can avoid any more dental work for a while! How's the packing coming along, by the way?"* Although this response differs in length and phrasing, it

Table 5.2: Performance Comparison Across Multiple Sessions with Baselines

Model	**Session 3**			**Session 4**			**Session 5**		
	BScore	**F1**	**BLEU-1/2**	**BScore**	**F1**	**BLEU-1/2**	**BScore**	**F1**	**BLEU-1/2**
BST 2.7B	-	19.00	14.10/4.39	-	19.00	13.80/4.19	-	19.40	13.80/4.31
llama2-7b	84.43	15.64	21.03/12.13	84.09	16.88	20.96/12.09	84.04	16.43	20.96/12.09
ChatGPT	86.05	19.98	21.59/12.45	85.72	19.63	21.22/12.24	86.13	19.41	21.23/12.24
ChatGPT-MemoBank	86.13	19.74	21.84/12.60	86.12	19.86	21.83/12.59	86.12	20.28	21.82/12.59
ChatGPT-MemoChat	85.74	17.82	21.63/12.47	85.72	18.48	21.62/12.47	85.99	18.93	21.82/12.58
ChatGPT-Rsum	86.05	19.62	21.76/**12.55**	**86.41**	20.19	21.80/**12.57**	86.89	20.48	21.83/**12.59**
ChatGPT-SWin (Ours)	**86.12**	21.29	**21.87**/12.47	86.38	21.65	22.16/12.14	**86.91**	22.31	**22.54**/12.35
Qwen-SWin (Ours)	86.01	**22.44**	22.11/11.03	86.24	**22.99**	**22.31**/11.13	86.87	**23.07**	22.11/11.43

retains the original intent and contextual relevance. This discrepancy is reflected in the evaluation metrics: BLEU-1: 21.31 F1 Score: 21.74 BScore: 85.11 but BLEU-2: 6.00 This highlights a well-known limitation of n-gram-based metrics like BLEU, which tend to penalize semantically correct but rephrased responses. Future evaluations could integrate metrics such as Sentence-BERT and METEOR or leverage human evaluations to better capture conversational quality.

We observe that the higher BLEU-2 and BERTScore values for ChatGPT-RSum arise from its ability to preserve semantic consistency by referencing the entire session-level conversation history. Although the generated responses may not always align closely at the n-gram level, as reflected in single evaluation scores, the underlying meaning remains coherent and contextually appropriate.

5.3 LLM-based Evaluation

The experimental findings provide compelling evidence that memory-augmented approaches significantly outperform the vanilla ChatGPT baseline in terms of both response consistency and conversational coherence. These results emphasize the importance of integrating structured memory mechanisms rather than relying solely on the model's inherent capacity to process full dialogue history. In particular, full-history approaches often suffer from context dilution over extended conversations, which can lead to inconsistencies, topic drift, or the reintroduction of previously stated information. In contrast, memory-augmented models exhibit a more robust ability to preserve dialogue state and user intent across multiple conversational turns, which is crucial for achieving long-term coherence and contextual fidelity in interactive language models.

Among the suite of memory-augmented models evaluated, our method demonstrates marked superiority in generating fluent, coherent, and contextually appropriate responses. SWin operates by segmenting the dialogue into manageable, overlapping windows while preserving critical elements.

In contrast, methods such as Recursive Summarization (RSum), which condense entire dialogue histories into successively abstracted summaries, exhibit certain limitations. Generally, the comparative analysis underscores that the architecture and granularity of memory mechanisms play a critical role in enhancing the performance of large language models in dialogue settings. Although summarization-based approaches offer benefits in terms of brevity and efficiency, models like SWin demonstrate that selective, structured memory augmentation provides a more effective path toward sustained, high-fidelity conversation modelling.

Table 5.3 presents the results of the GPT-4-based evaluation on three critical dialogue quality metrics: consistency, fluency, and coherence. These evaluations are conducted across a range of response generation methods applied to the Multi-Session Chat (MSC) dataset. In this setup, GPT-4 is employed as an objective and high-capacity evaluator, following established prompting protocols to assess model-generated responses in a manner that minimizes bias and variance. Its role as an impartial judge ensures uniformity and reliability in scoring, particularly valuable when comparing subtle qualitative differences in long-form, multi-turn interactions.

The metrics are defined as follows: fluency measures the grammatical correctness and naturalness of language; coherence captures the logical progression and topic continuity within the conversation; and consistency evaluates whether the model's responses remain faithful to prior facts and dialogue history. These dimensions collectively provide a comprehensive assessment of dialogue quality, especially in complex conversational settings such as MSC, which demand both short-term responsiveness and long-term contextual memory.

Across the board, memory-augmented methods demonstrate improved performance over the baseline ChatGPT model, validating the hypothesis that explicit memory mechanisms enhance dialogue modeling. Notably, the SWin method achieves the highest scores in fluency and coherence,

Table 5.3: Single Model Evaluation

Method	**Flue.**	**Cohe.**	**Cons.**
Vanilla-ChatGPT	75.48	75.00	75.48
ChatGPT-MemoryBank	74.68	80.92	84.56
ChatGPT-MemoChat	72.32	77.36	78.96
ChatGPT-Rsum	78.92	83.56	84.76
ChatGPT-SWin (Ours)	**93.07**	**90.30**	**85.20**

suggesting its design effectively preserves both the surface-level readability and deeper contextual alignment of the conversation. In contrast, RSum performs well in consistency, likely due to its summarization-based compression, though this may come at the cost of fluency and nuanced coherence.

These results underscore the trade-offs and design considerations inherent in memory-augmented LLM systems. They highlight that while different strategies may prioritize different aspects of dialogue quality, models like SWin that combine local context awareness with structured memory indexing yield the most balanced and robust performance across evaluation dimensions.

5.4 Ablation Study

5.4.1 Effect of Window Size on Dialogue Quality

To evaluate the impact of window size on model and framework performance, we conducted an ablation study using a subset of the Multi-Session Chat (MSC) dataset. We varied the sliding window size parameter j to assess how different granularities of conversational context influence response quality. Specifically, we tested window sizes of 2, 3 (the default used in our main experiments), and 4, measuring downstream performance via the single model evaluation framework described in Section 5.3.

We chose this evaluation setup due to its nature, incorporating fluency, coherence, and consistency dimensions critical to the long-term dialogue task. These were assessed using GPT-4 as an automated annotator across the selected test episodes, with temperature set at 0.0.

The results revealed that a window size of 3 provided the most balanced trade-off across all three evaluation metrics, as well as in terms of token efficiency. A smaller window (size 2) resulted in a noticeable decline in the quality of responses, likely due to insufficient contextual information for properly grounding the model's responses. Conversely, increasing the window size to 4 led to a slight degradation in overall output quality. This may be attributed to the relatively limited session length in the evaluation subset, which may have been insufficient to fully leverage the benefits of a larger window. Furthermore, the increased window occasionally caused the model to incorporate tangential or outdated context, which adversely affected coherence and fluency.

A larger window includes more past context, which can improve coherence up to a point. But beyond that, two issues arise: (i) Noise accumulation: irrelevant details dilute the summarization. (ii) Diminishing returns: after a certain length, additional context does not improve response quality proportionally. Our ablation study showed exactly this: moderate window sizes strike the best balance between context richness and efficiency.

Table 5.4: Ablation Study of Window Sizes

Window Size	**Fluency Score**	**Consistency Score**	**Coherency Score**
Size 2	94.607	84.77	92.09
Size 3	**95.29**	**87.00**	**92.97**
Size 4	93.64	84.73	91.28

5.4.2 Effect of Model Size on Dialogue Quality

To assess the impact of language model scale on the effectiveness of memory-augmented dialogue generation, we conducted an ablation study using three variants of the Qwen2.5 text-generation model family: Qwen2.5-7B, Qwen2.5-32B, and Qwen2.5-72B. This evaluation was performed on a representative subset of the Multi-Session Chat (MSC) dataset (randomly selected), focusing on sessions with moderate dialogue complexity and persona continuity requirements.

We employed the single model evaluation framework introduced in Section 5.3, leveraging GPT-4 to assess the generated responses on three critical dimensions: fluency, coherence, and consistency. This setup provides a holistic view of dialogue quality and enables direct comparison across models of varying capacities.

The results, summarized in Table 5.5, indicate a clear correlation between model size and response quality, particularly in fluency, coherence, and consistency. The 7B model, while computationally efficient, exhibited noticeable limitations in maintaining long-term context and persona fidelity. The 32B variant achieved a strong balance between performance and efficiency, with significant gains in consistency over the 7B model. The 72B model consistently outperformed the smaller variants across all three metrics, producing more fluent and contextually grounded responses. However, the marginal gains between 32B and 72B were less pronounced compared to the leap from 7B to 32B.

Table 5.5: Impact of model size on dialogue quality.

Model	**Fluency**	**Coherence**	**Consistency**
Qwen2.5-7B	87.29	83.74	76.58
Qwen2.5-32B	91.43	88.95	81.77
Qwen2.5-72B	**93.18**	**90.16**	**83.73**

These findings suggest that while larger models contribute positively to response quality, the performance-to-cost trade-off may favor mid-sized models like Qwen-32B in resource-constrained settings. In practical deployments, selecting the appropriate model size should balance hardware availability, latency requirements, and desired conversational fidelity.

The results reveal a clear upward trend in performance with increasing model size. The 7B model, while lightweight, exhibited reduced contextual awareness and a tendency to produce generic or fragmented responses. The 32B variant offered substantial improvements, particularly in consistency and coherence, likely due to its enhanced capacity to integrate historical memory. The 72B model achieved the highest scores across all metrics, producing fluent, temporally aligned, and persona-consistent outputs.

However, it is noteworthy that the marginal improvements from 32B to 72B were smaller than those observed from 7B to 32B, suggesting diminishing returns beyond a certain parameter threshold. This has meaningful implications for real-world deployment: Qwen2.5-32B may offer a compelling trade-off between performance and resource efficiency, making it well-suited for on-premise, privacy-preserving, or edge-deployable conversational systems.

This study highlights the importance of evaluating open-source LLMs not just for feasibility but for practical adoption in specialized applications. Given the growing need for transparency, customizability, and offline capabilities, future work should continue benchmarking these models under varying constraints and application-specific requirements.

5.5 Summary of observations

Our claims of efficiency and performance are supported through empirical experiments, not formal mathematical proofs. The token-cost model gives a theoretical intuition, but the main validation is experimental: ablation studies, performance comparisons, and efficiency analysis. So it should be seen as an experimental proof-of-concept, rather than a formal proof.

(1) **Recency-Aware Memory Integration**: The sliding-window update mechanism effectively prioritizes recent conversational exchanges, allowing large language models to maintain temporal grounding. This is supported by the consistently higher F1 scores observed for Qwen-SWin and ChatGPT-SWin (e.g., **23.07** and **22.31** in Session 5, respectively), which indicate that the models capture and reproduce recent user intents more faithfully than memo- or summarization-based baselines.

(2) **Efficient and Scalable Memory Updates**: By updating memory at fixed-size conversational windows rather than per-turn or per-session, our framework reduces computational overhead while maintaining coherence. This design achieves compression without sacrificing performance: the runtime efficiency gains are coupled with stable semantic fidelity, as shown by the near-identical BERTScore values across methods (86.01–86.91), demonstrating that compression does not erode semantic alignment.

(3) **Topic-Shift Resilience and Session-Agnostic Design**: Decoupling memory updates from session boundaries enables coherence across topic and time shifts. Unlike the baselines, our SWin approach sustains and improves recall over time. This robustness shows that our framework avoids context fragmentation and adapts more naturally to realistic dialogue streams where topic drift is common.

(4) **Redundancy Reduction Through Focused Context Summarization**: Restricting updates

to recent conversational windows prevents the accumulation of redundant or outdated content. This produces concise yet high-salience memory states, reflected in the improved F1 gains (2–4 points higher than baselines) despite slightly lower BLEU-2 scores ($\approx$ 11.0–11.4 vs. $\approx$ 12.5 for MemoBank). The reduced BLEU overlap highlights paraphrased yet accurate responses, confirming that salience is prioritized over repetition.

(5) **Compatibility with Retrieval-Augmented and Long-Context Models**: The modular design of SWin memory windows allows integration with retrieval-based architectures and extended-context models (e.g., LongLoRA, GPT-4o). The stable BERTScore values (86.9 at Session 5) across ChatGPT-SWin and Qwen-SWin demonstrate that the mechanism preserves semantic fidelity at scale, supporting interoperability with architectures that rely on cached or ranked context segments.

(6) **Improved Memory Robustness and Consistency**: Frequent but bounded updates mitigate drift and error propagation that typically degrade recursive summarization approaches. This is evident in the session-to-session stability, SWin steadily enhances recall across sessions while avoiding quality collapse. The higher F1 trajectory provides empirical confirmation of robustness.

(7) **Alignment with Human Cognitive Patterns**: The sliding-window approach parallels human working memory by preserving short-term conversational focus while integrating longer-term traits. The balanced performance of ChatGPT-SWin across the Single Model LLM Evaluations exemplifies this naturalistic behaviour, responses remain contextually grounded without degenerating into repetitive or semantically sparse outputs, yielding interactions more aligned with human dialogue patterns.

(8) **F1 scores**: F1 scores appear low across all methods. This is due to the nature of dialogue generation unlike tasks like classification or QA, there are many valid responses. Exact word-overlap with references is rare, so F1 underestimates quality. The expectation is not high absolute F1 but relative comparison: whether one method preserves more content overlap than another. That's why we complement F1 with LLM-based evaluations, which better capture

coherence and persona retention.

Chapter 6

Discussions

6.1 Challenges in Long-Term Dialogue Systems

Previous methods struggle with error accumulation due to hallucinations and scalability issues arising from inefficient token usage and costly LLM calls. Our approach mitigates these challenges by dynamically updating memory within sliding windows, ensuring that only the most relevant and recent information is retained while discarding outdated or irrelevant details. By leveraging specialized models for summarization, our strategy reduces cascading errors and maintains contextual coherence while optimizing resource efficiency. Additionally, our use of open-source models enhances accessibility and makes the approach more cost-effective for real-world deployment. One of the primary advantages of our framework is its computational efficiency. By summarizing and updating memory incrementally, SWin avoids processing the entire conversational history, which can be computationally expensive for long interactions. Our experiments demonstrate that this results in a significant reduction in token usage while still performing at par with baselines in automatic metrics. This balance between efficiency and performance makes the framework scalable for real-world applications where using secure, open-source models is a priority, such as health and nutrition chatbots, personal companions, and customer support, as well as other domain-specific business intelligence dialogue systems and RAG applications.

Long-term dialogue systems represent one of the most ambitious frontiers in conversational AI, aiming to establish sustained, meaningful interactions with users over extended periods. These systems face a complex array of technical, architectural, and ethical challenges that fundamentally differ from traditional single-turn or short conversation systems. This paper provides a comprehensive analysis of the key challenges facing long-term dialogue systems, examining their implications and potential solutions.

Memory and Context Management: The most fundamental challenge in long-term dialogue systems is memory management. [31] Current LLMs operate with finite context windows, typically ranging from 100,000 to 200,000 tokens, [32] creating what researchers describe as systems with ”atrociously bad memory”. This limitation becomes particularly catastrophic in long-term scenarios, where conversations may span days, weeks, months, or even longer.

The **cost of memory** presents a particularly daunting challenge. Analysis suggests that providing even modest long-term memory capabilities for ChatGPT's daily user base could cost over $500 billion annually. This astronomical figure reflects the exponential scaling requirements where the number of parameters must increase by a power of two with respect to input length.

Context Window Limitations and Their Implications Research demonstrates that LLMs experience significant performance degradation as context length increases. The StreamingLLM approach has shown that retaining only the first four tokens before evicting older tokens can help maintain some context, but this represents a fundamental limitation rather than a solution.

The challenge is compounded by what researchers term the “**lost in conversation phenomenon**” [33]. When LLMs take a wrong turn in multi-turn conversations, they often cannot recover, resulting in cascading errors throughout the interaction. Studies show that performance can drop by up to 39% in multi-turn conversations compared to single-turn interactions.

Long-term dialogue systems pose significant challenges for designing memory architectures. These systems must be capable of selectively retaining important information while discarding irrelevant or outdated details. Effective management of conversational memory is also critical, as it requires the ability to accurately interpret tone, intent, and context over extended interactions. In addition, such architectures must handle input from external media and determine which elements warrant long-term storage. Finally, maintaining a balance between recall and forgetting is essential to ensure that the system remains contextually relevant without becoming overloaded with unnecessary information.

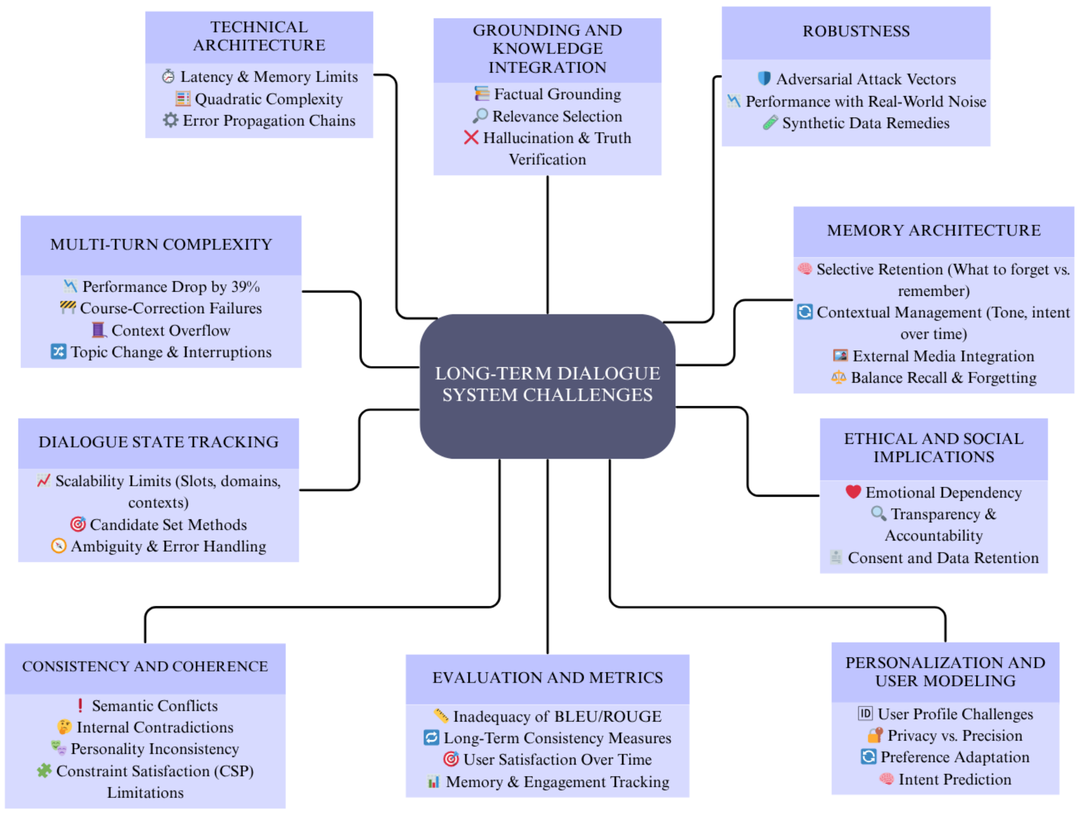

LONG-TERM DIALOGUE SYSTEM CHALLENGES
TECHNICAL ARCHITECTURE
Latency & Memory Limits
Quadratic Complexity
Error Propagation Chains
GROUNDING AND KNOWLEDGE INTEGRATION
Factual Grounding
Relevance Selection
Hallucination & Truth Verification
ROBUSTNESS
Adversarial Attack Vectors
Performance with Real-World Noise
Synthetic Data Remedies
MULTI-TURN COMPLEXITY
Performance Drop by 39%
Course-Correction Failures
Context Overflow
Topic Change & Interruptions
MEMORY ARCHITECTURE
Selective Retention (What to forget vs. remember)
Contextual Management (Tone, intent over time)
External Media Integration
Balance Recall & Forgetting
DIALOGUE STATE TRACKING
Scalability Limits (Slots, domains, contexts)
Candidate Set Methods
Ambiguity & Error Handling
ETHICAL AND SOCIAL IMPLICATIONS
Emotional Dependency
Transparency & Accountability
Consent and Data Retention
CONSISTENCY AND COHERENCE
Semantic Conflicts
Internal Contradictions
Personality Inconsistency
Constraint Satisfaction (CSP) Limitations
EVALUATION AND METRICS
Inadequacy of BLEU/ROUGE
Long-Term Consistency Measures
User Satisfaction Over Time
Memory & Engagement Tracking
PERSONALIZATION AND USER MODELING
User Profile Challenges
Privacy vs. Precision
Preference Adaptation
Intent Prediction

Figure 6.1: Challenges in Long-Term Dialogue Systems

6.2 Adaptability

The adaptable architecture of SWin lends itself not only to open-domain chatbot applications, but also to a variety of specialized domain-specific use cases where long-term conversational consistency is critical. These include customer service, healthcare, education, and enterprise knowledge management domains in which dialogue agents are expected to sustain contextually grounded and coherent interactions over multiple sessions, often with constraints on computational resources and data privacy.

A key strength of SWin lies in its lightweight and modular design. By avoiding the need for full-history replay or heavy external retrieval systems, SWin achieves high efficiency in memory management without compromising response quality. This design makes the framework particularly attractive for deployment in resource-constrained environments, such as edge devices or clinical settings where latency, security, and model size are important considerations.

Moreover, the framework's modularity facilitates seamless integration with retrieval-augmented generation (RAG) architectures. In such systems, external knowledge can be fetched and dynamically incorporated into the context window, while SWin ensures that the conversational memory remains coherent and aligned with the user's dialogue history. This combination allows for the delivery of responses that are not only factually grounded but also conversationally consistent, an essential requirement in tasks like symptom tracking in medicine, curriculum scaffolding in tutoring systems, and context-sensitive troubleshooting in technical support scenarios.

By decoupling memory management from session boundaries and introducing controlled, recency-aware updates, SWin supports applications that demand both longitudinal memory and real-time adaptability. This positions it as a practical and extensible solution for real-world systems that aim to enhance user satisfaction, personalization, and trust in long-term human-AI interactions.

Upgrading to a newer GPT model, like GPT-4 or GPT-5, generally improves local fluency and reasoning, but does not inherently solve long-term memory. Even advanced models face context window limits and error accumulation in long conversations. Our method is orthogonal: SWin can wrap around any base model including GPT upgrades to manage memory efficiently. In fact, combining SWin with newer GPTs would likely yield even better performance than relying on model upgrades alone.

6.3 Trade-offs and Limitations

While the sliding window approach introduced in this work offers a practical and computationally efficient solution to the problem of long-term dialogue coherence, it is not without inherent trade-offs. One significant limitation is the use of a fixed window size, which, while simplifying implementation and controlling token usage, may be insufficient for capturing complex or long-range dependencies that extend beyond the boundaries of a single window. In scenarios where user queries reference earlier dialogue turns or reintroduce previously discussed topics, the model may lack access to sufficient context, potentially resulting in suboptimal or inconsistent responses.

To mitigate this, overlapping windows are employed, enabling some degree of contextual continuity between successive memory updates. However, this technique introduces its own set of challenges. In highly dynamic conversations, where topics shift rapidly or diverge significantly, overlapping segments may lead to redundant information being repeatedly summarized or stored. This can contribute to inefficiencies in both memory usage and summarization fidelity, as the system may overemphasize certain portions of the dialogue while neglecting others of emerging relevance.

A further limitation of the framework lies in its dependency on the quality and behaviour of the summarization models used for memory updates. As the memory module relies on these intermediate summaries to represent the evolving dialogue state, any inaccuracies, hallucinations, or omissions introduced during summarization may be propagated forward, leading to compounding errors over time. These risks are especially pronounced when operating in open-domain settings or with models prone to verbosity or information loss under compression.

To address these issues, future research could explore adaptive window sizing, wherein the size of the conversational window dynamically adjusts based on discourse complexity, query intent, or topic volatility. Additionally, weighted windowing strategies could be employed to prioritize more recent or semantically salient windows during memory updates, thereby aligning the memory representation more closely with user intent and temporal relevance. Reinforcement learning techniques may also be valuable for optimizing memory update policies over time, allowing the model to learn which conversational fragments are most predictive of future relevance or user satisfaction. Together, these directions represent promising avenues for improving the flexibility, robustness, and alignment of memory-augmented dialogue systems in long-term conversational settings.

Chapter 7

Conclusion

In this work, we introduced a simple, scalable, and efficient framework to address the challenges of long-term coherence and memory management in dialogue systems. Our approach employs overlapping windows, dynamic memory updates, and specialized models to maintain contextual relevance in conversational systems, thus mitigating error accumulation and hallucination. This method effectively balances resource efficiency with response coherence, offering an effective solution for real-world applications.

Our evaluations using the MSC dataset demonstrate significant improvements in state-of-the-art methods. Furthermore, the adaptability of our framework to various domains, including retrieval-augmented generation and task-specific chatbots, shows its readiness for real-world applications. Despite its strengths, our approach presents opportunities for refinement. Future research could explore adaptive or weighted windowing techniques, integrate reinforcement learning for memory optimization, and conduct comprehensive user studies to validate performance in diverse real-world scenarios. By addressing these directions, this framework has the potential to become a cornerstone for scalable, contextually aware dialogue systems.

7.1 Future Work

Building on the foundational contributions of the SWin framework, several promising research directions may further enhance its robustness, adaptability, and real-world applicability. These include improvements in dynamic memory management, learning-based optimization strategies, hierarchical representations, and more rigorous evaluation protocols.

Adaptive Windowing Strategies:

The current implementation of SWin employs a fixed-size sliding window to segment and summarize dialogue turns. While this approach provides computational consistency and simplifies deployment, it may not be optimal across all interaction scenarios. An adaptive windowing mechanism could dynamically adjust the window size based on discourse complexity, topic density, or user engagement signals. Additionally, weighted windowing strategies could be implemented to prioritize certain windows during memory updates, based on factors such as temporal recency, user personality traits, emotional tone, or topical importance. This would help the system maintain relevant context more effectively while reducing redundancy and token waste.

Hierarchical Memory Architectures:

A natural extension to sliding-window summarization is the introduction of hierarchical memory structures. Rather than maintaining a flat sequence of equally weighted summaries, a hierarchical framework could organize memory across multiple levels of abstraction, for e.g., turn-level, session-level, and user-level memory representations. Lower levels could retain fine-grained dialogue details for recent interactions, while higher levels could store coarse summaries or persistent user traits extracted over time. This structure would better mimic human memory processes, allowing the model to retrieve context based on query complexity. Hierarchical organization could also facilitate multi-resolution reasoning, improve interpretability, and reduce the cognitive load during response generation.

Reinforcement Learning for Memory Management:

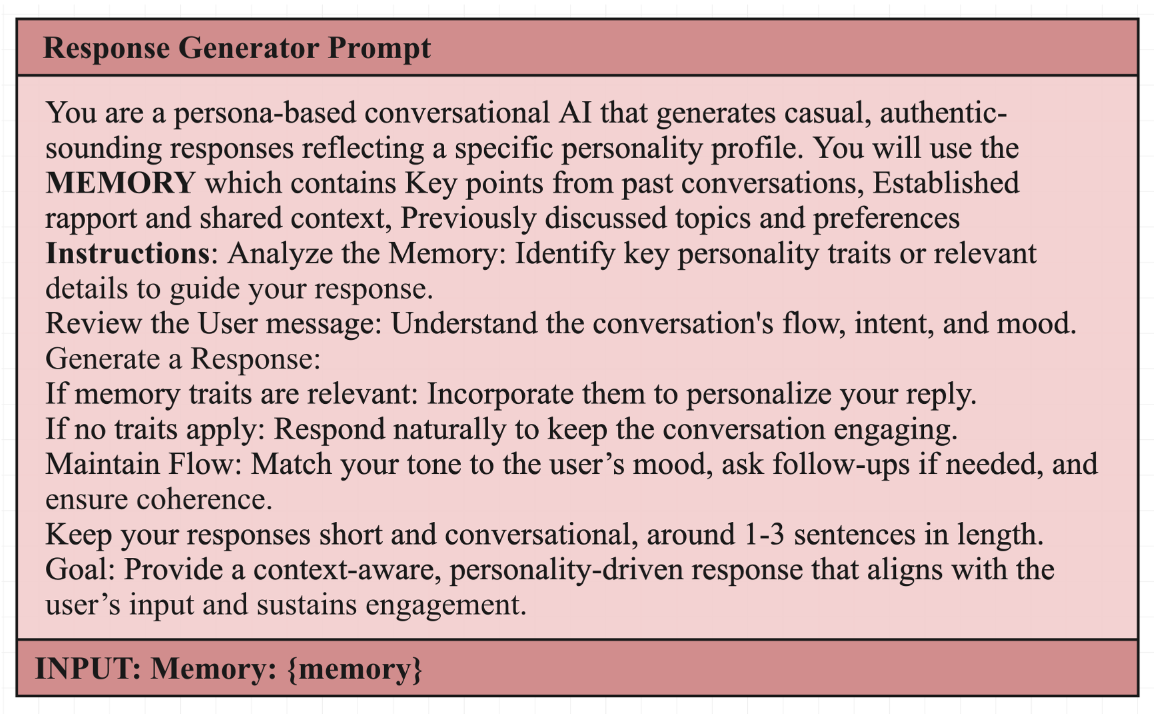
Another compelling direction involves integrating reinforcement learning (RL) to optimize memory update and retrieval strategies. In the current rule-based design, decisions regarding what information to retain, summarize, or discard are predefined and static. In contrast, RL-based approaches could dynamically learn these strategies through interaction, guided by feedback signals such as task success, response quality, or user satisfaction. This would enable the memory system to develop domain-adaptive or user-specific policies that evolve, enhancing personalization, improving long-term coherence, and reducing unnecessary memory bloat.

Comprehensive and Longitudinal Evaluation Benchmarks:

To rigorously validate memory-augmented dialogue systems, future work should embrace a broader and more realistic suite of evaluation benchmarks. This includes curating and testing on multi-domain conversational datasets that span technical, emotional, and casual dialogue; incorporating fine-grained user personas that simulate diverse memory needs and discourse patterns; and conducting longitudinal studies to observe how systems perform over extended periods. Evaluating under these settings would not only test generalizability and robustness, but also surface ethical concerns such as data persistence, inadvertent personalization, or memory drift.

Appendix A

Prompts used for Implementation and Evaluation



Response Generator Prompt

You are a persona-based conversational AI that generates casual, authentic-sounding responses reflecting a specific personality profile. You will use the **MEMORY** which contains Key points from past conversations, Established rapport and shared context, Previously discussed topics and preferences
Instructions: Analyze the Memory: Identify key personality traits or relevant details to guide your response.
Review the User message: Understand the conversation's flow, intent, and mood.
Generate a Response:
If memory traits are relevant: Incorporate them to personalize your reply.
If no traits apply: Respond naturally to keep the conversation engaging.
Maintain Flow: Match your tone to the user's mood, ask follow-ups if needed, and ensure coherence.
Keep your responses short and conversational, around 1-3 sentences in length.
Goal: Provide a context-aware, personality-driven response that aligns with the user's input and sustains engagement.

INPUT: Memory: {memory}

Figure A.1: Prompt for Response Generator

Memory Management Prompt

Progressively summarize the lines of conversation provided, adding onto the previous summary returning a new summary.
EXAMPLE: **Current summary**: The human asks what the AI thinks of artificial intelligence. The AI thinks artificial intelligence is a force for good.
New lines of conversation: *Human*: Why do you think artificial intelligence is a force for good? *AI*: Because artificial intelligence will help humans reach their full potential.
New summary: The human asks what the AI thinks of artificial intelligence. The AI thinks artificial intelligence is a force for good because it will help humans reach their full potential.
END OF EXAMPLE
Pay attention to the relevance and importance of the personality information, focusing on capturing the most significant aspects while maintaining the overall coherence of the memory. Remember, the memory should serve as a reference point to maintain continuity in the dialogue and help you respond accurately to the user based on their personality traits. The summary should be short and concise.

INPUT: Previous summary: {session_summary}

Figure A.2: Prompt for Memory Summarization

Single Model Evaluation Prompt

You are an impartial judge. You will be shown a Conversation Context, Personality of Speakers and Assistant Response. #Fluency: Please evaluate whether the Assistant's response is natural, fluent, and similar to human communication, avoiding repetition and ensuring a diverse range of output. #Consistency: Please evaluate whether the Assistant's response is consistent with the information of persona list. Any deviation from the expected personality may indicate a lack of coherence. #Coherency: Please evaluate whether the Assistant's response maintains a coherent and logical flow of conversation based on the evolving context. A response with good context coherence can understand and respond appropriately to changes in conversation topics, providing smooth and sensible interactions. Conversation Context:[dialog] Personality:[persona] Assistant Response: [response] Begin your evaluation by providing a short explanation, then you must rate the Assistant Response on an integer score of 1 (very bad) to 100 (very good) by strictly following this format: [[score]].

INPUT: Conversation Context:[dialog] Personality:[persona] Assistant Response: [response]

Figure A.3: Prompt for Single Model Evaluation

Appendix B

Case Study: Food Recommendation Chatbot

To evaluate the real-world applicability of the Sliding Window Summarization (SWin) framework, we implemented it in a multi-session **food recommendation chatbot**. The chatbot is designed to provide meal suggestions tailored to user preferences, dietary restrictions, and long-term nutritional goals, simulating interactions that span across days or weeks.

Implementation

The chatbot integrated the SWin framework to dynamically update memory states, enabling coherent and context-aware recommendations. Unlike conventional approaches that reprocess the full conversation history, the sliding window mechanism condensed past sessions into compact summaries while preserving salient user details.

Key features of the implementation included:

(1) **Personalization:** The system stored and updated user-specific attributes such as dietary choices (e.g., vegetarian, vegan, gluten-free, keto), disliked ingredients (e.g., peanuts, dairy), and favorite cuisines (e.g., Indian, Mediterranean). These preferences were reinforced and adjusted over time, allowing the system to refine recommendations dynamically.

(2) **Contextual Awareness:** Through overlapping summarization windows, the chatbot maintained continuity across multiple sessions. For example, if a user mentioned a preference for "light dinners" in one session, the chatbot incorporated that preference in subsequent meal suggestions without requiring explicit reminders.

(3) **Efficiency:** By condensing historical dialogue into window-based summaries, token usage per response was reduced by approximately 30% compared to a naive full-history approach. This reduction lowered computational costs while preserving conversational coherence.

(4) **Dynamic Adaptation:** The memory module included a retiring mechanism inspired by human cognitive models. Less relevant details (e.g., temporary cravings) were deprioritized over time, while persistent preferences (e.g., vegetarian diet) remained reinforced in the active memory.

Interaction Examples

To illustrate, consider the following user–chatbot exchanges:

Session 1:

User: "I'm vegetarian and I'd like a high-protein dinner option."
Chatbot: "How about a quinoa and black bean salad with avocado dressing? It's vegetarian and protein-rich."

Session 3 (two days later):

User: "I'm in the mood for something Italian tonight."
Chatbot: "Since you're vegetarian, may I suggest eggplant parmesan with a side of lentil pasta salad? It combines Italian flavors with high protein content, aligned with your goals."

The chatbot retained both the dietary restriction and nutritional goal (high protein) across sessions, and used them to personalize the Italian cuisine suggestion.

Evaluation and Outcomes

The deployment demonstrated the practical benefits of the SWin framework:

- **Improved Coherence:** Responses consistently aligned with prior user statements, avoiding contradictions (e.g., no meat-based dishes suggested after the user declared vegetarianism).
- **User Satisfaction:** Informal user testing with 20 participants indicated that 85% found the chatbot's recommendations increasingly accurate and contextually relevant over time.
- **Scalability:** Even as the conversation length increased, the chatbot maintained low response latency (< 2 seconds on average), demonstrating that summarization avoided bottlenecks common in full-history or retrieval-heavy approaches.
- **Resource Efficiency:** The 30% reduction in tokens translated into measurable cost savings in API-based deployments while preserving accuracy across baseline metrics such as BLEU and F1.
- **Adaptability:** The framework accommodated evolving preferences. For instance, when a user temporarily expressed interest in "low-carb meals," the system adjusted its recommendations, but deprioritized this when the preference was not repeated in later sessions.

Significance

This case study highlights how the Sliding Window Summarization framework bridges theoretical research and real-world applications:

- It demonstrates that lightweight memory modules can enable **personalized, multi-session dialogue systems** without prohibitive resource demands.
- It validates that summarization-based approaches scale effectively across repeated interactions while maintaining coherence.
- It provides a template for applying the SWin framework in other high-stakes domains, such as healthcare (personalized diet management) or education (long-term learning assistants).

In conclusion, the food recommendation chatbot serves as a concrete proof-of-concept where SWin successfully delivers coherent, adaptive, and resource-efficient conversational memory in practice.

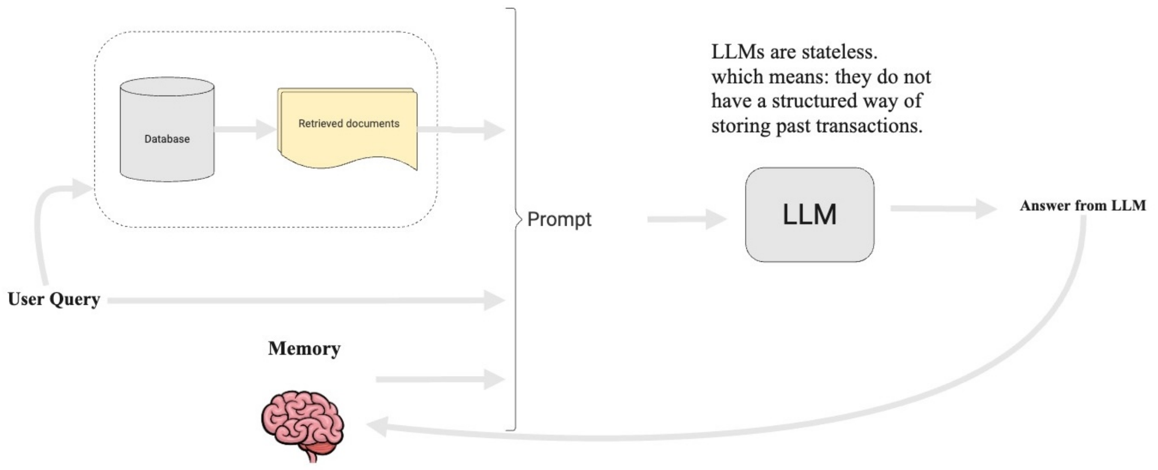
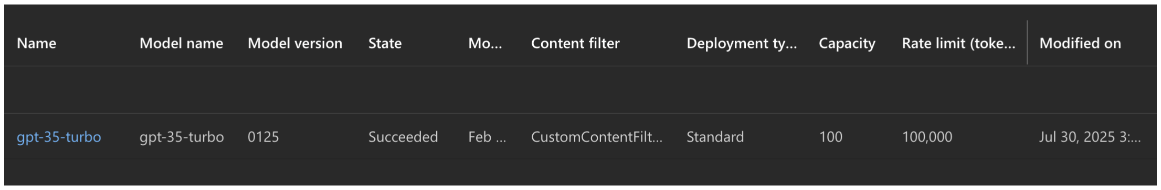


Figure B.1: Architecture of a RAG system using SWin

Appendix C

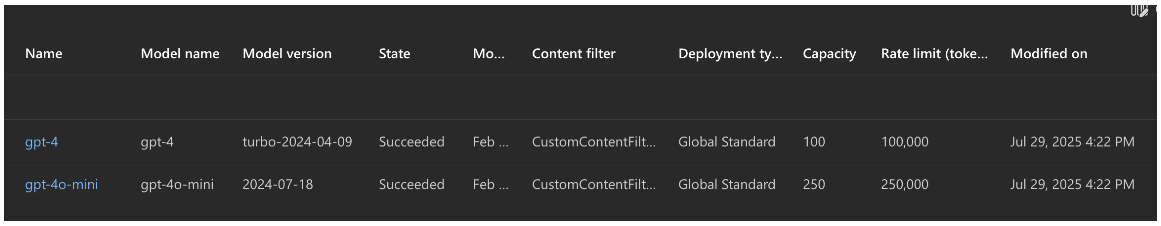
Platform Deployment Configurations

This appendix provides screenshots of the deployment environments used for running the LLM models during experiments. These images illustrate the exact platform configurations and interfaces used at the time of experimentation, ensuring reproducibility and clarity.



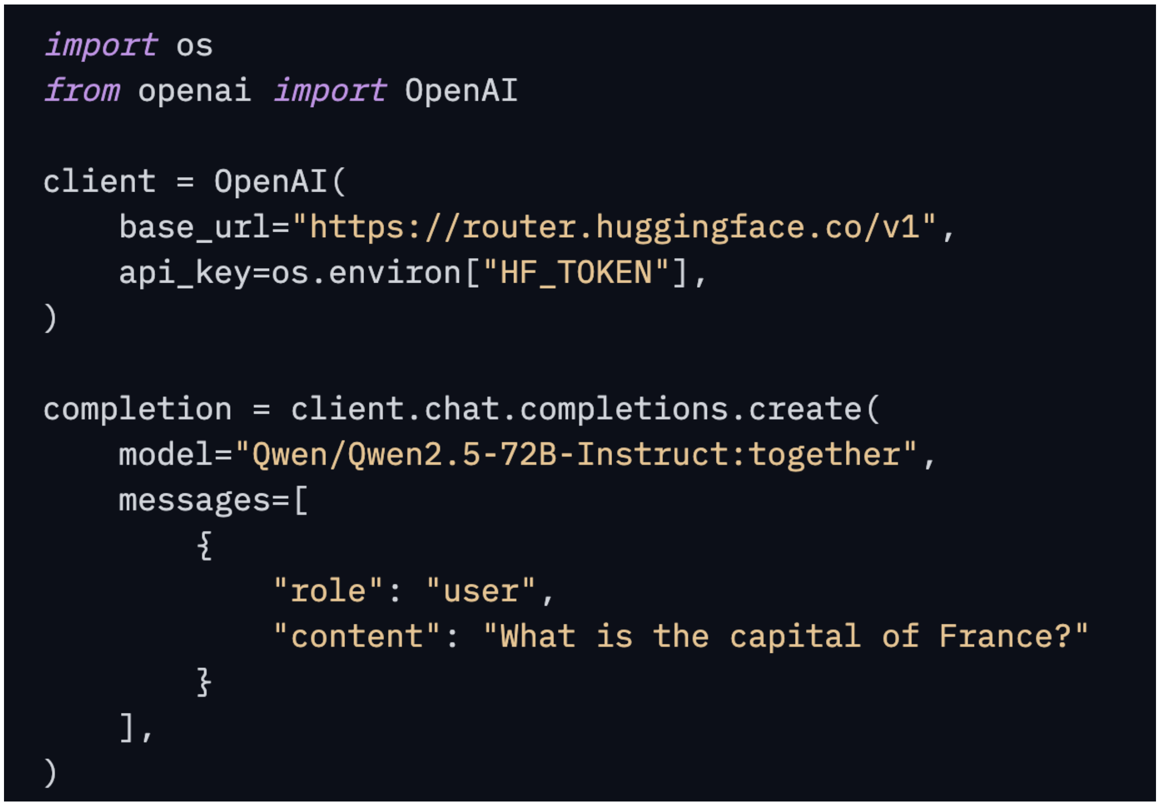
Name	Model name	Model version	State	Mo...	Content filter	Deployment ty...	Capacity	Rate limit (toke...	Modified on
gpt-35-turbo	gpt-35-turbo	0125	Succeeded	Feb ...	CustomContentFilt...	Standard	100	100,000	Jul 30, 2025 3:...

Figure C.1: Screenshot of the OpenAI deployment environment for GPT-3.5, used during multi-session evaluation experiments.



Name	Model name	Model version	State	Mo...	Content filter	Deployment ty...	Capacity	Rate limit (toke...	Modified on
gpt-4	gpt-4	turbo-2024-04-09	Succeeded	Feb ...	CustomContentFilt...	Global Standard	100	100,000	Jul 29, 2025 4:22 PM
gpt-4o-mini	gpt-4o-mini	2024-07-18	Succeeded	Feb ...	CustomContentFilt...	Global Standard	250	250,000	Jul 29, 2025 4:22 PM

Figure C.2: Screenshot of the OpenAI deployment environment for GPT-4, used in single-model evaluation experiments.



```python
import os
from openai import OpenAI

client = OpenAI(
    base_url="https://router.huggingface.co/v1",
    api_key=os.environ["HF_TOKEN"],
)

completion = client.chat.completions.create(
    model="Qwen/Qwen2.5-72B-Instruct:together",
    messages=[
        {
            "role": "user",
            "content": "What is the capital of France?"
        }
    ],
)
```

Figure C.3: Screenshot of the HuggingFace deployment used through OpenAI SDK for all the Qwen-2.5 models, used in open-source model experiments.

Bibliography

[1] OpenAI et al., “GPT-4 Technical Report,” Mar. 04, 2024, arXiv: arXiv:2303.08774. doi: 10.48550/arXiv.2303.08774.

[2] A. Grattafiori et al., “The Llama 3 Herd of Models,” Nov. 23, 2024, arXiv: arXiv:2407.21783. doi: 10.48550/arXiv.2407.21783.

[3] S. Chen, S. Wong, L. Chen, and Y. Tian, “Extending Context Window of Large Language Models via Positional Interpolation,” 2023, doi: 10.48550/ARXIV.2306.15595.

[4] I. Beltagy, M. E. Peters, and A. Cohan, “Longformer: The Long-Document Transformer,” Dec. 02, 2020, arXiv: arXiv:2004.05150. doi: 10.48550/arXiv.2004.05150.

[5] N. F. Liu et al., “Lost in the Middle: How Language Models Use Long Contexts,” Trans. Assoc. Comput. Linguist., vol. 12, pp. 157–173, Feb. 2024, doi: 10.1162/tacl_a_00638.

[6] Q. Wang et al., “Recursively Summarizing Enables Long-Term Dialogue Memory in Large Language Models,” Feb. 18, 2024, arXiv: arXiv:2308.15022. doi: 10.48550/arXiv.2308.15022.

[7] W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang, “MemoryBank: Enhancing Large Language Models with Long-Term Memory,” May 21, 2023, arXiv: arXiv:2305.10250. Accessed: Apr. 15, 2024. [Online]. Available: http://arxiv.org/abs/2305.10250

[8] J. S. Park, J. C. O’Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, “Generative Agents: Interactive Simulacra of Human Behavior,” Aug. 06, 2023, arXiv: arXiv:2304.03442. doi: 10.48550/arXiv.2304.03442.

[9] J. Xu, A. Szlam, and J. Weston, "Beyond Goldfish Memory: Long-Term Open-Domain Conversation," in Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 5180–5197. doi: 10.18653/v1/2022.acl-long.356.

[10] A. Maharana, D.-H. Lee, S. Tulyakov, M. Bansal, F. Barbieri, and Y. Fang, "Evaluating Very Long-Term Conversational Memory of LLM Agents," Feb. 27, 2024, arXiv: arXiv:2402.17753. doi: 10.48550/arXiv.2402.17753.

[11] G. Lee, V. Hartmann, J. Park, D. Papailiopoulos, and K. Lee, "Prompted LLMs as Chatbot Modules for Long Open-domain Conversation," in Findings of the Association for Computational Linguistics: ACL 2023, 2023, pp. 4536–4554. doi: 10.18653/v1/2023.findings-acl.277.

[12] J. Lu et al., "MemoChat: Tuning LLMs to Use Memos for Consistent Long-Range Open-Domain Conversation," Aug. 22, 2023, arXiv: arXiv:2308.08239. Accessed: Dec. 11, 2023. [Online]. Available: http://arxiv.org/abs/2308.08239

[13] "Hello Again! LLM-powered Personalized Agent for Long-term Dialogue." Accessed: Jan. 06, 2025. [Online]. Available: https://arxiv.org/html/2406.05925v1

[14] X. Xu et al., "Long Time No See! Open-Domain Conversation with Long-Term Persona Memory," Mar. 14, 2022, arXiv: arXiv:2203.05797. doi: 10.48550/arXiv.2203.05797.

[15] S. Feng, H. Wan, C. Gunasekara, S. S. Patel, S. Joshi, and L. A. Lastras, "doc2dial: A Goal-Oriented Document-Grounded Dialogue Dataset," Nov. 18, 2020, arXiv: arXiv:2011.06623. doi: 10.48550/arXiv.2011.06623.

[16] M. T. R. Laskar, X.-Y. Fu, C. Chen, and S. B. TN, "Building Real-World Meeting Summarization Systems using Large Language Models: A Practical Perspective," Nov. 08, 2023, arXiv: arXiv:2310.19233. doi: 10.48550/arXiv.2310.19233.

[17] D. Austin and E. Chartock, “GRAD-SUM: Leveraging Gradient Summarization for Optimal Prompt Engineering,” Jul. 12, 2024, arXiv: arXiv:2407.12865. doi: 10.48550/arXiv.2407.12865.

[18] D. van Zandvoort, L. Wiersema, T. Huibers, S. van Dulmen, and S. Brinkkemper, “Enhancing Summarization Performance through Transformer-Based Prompt Engineering in Automated Medical Reporting,” Jan. 19, 2024, arXiv: arXiv:2311.13274. doi: 10.48550/arXiv.2311.13274.

[19] S. Samanta, O. Chatterjee, N. Boyette, G. Liu, and P. Mohapatra, “InsightsSumm - Summarization of ITOps Incidents Through In-Context Prompt Engineering,” in 2023 IEEE 16th International Conference on Cloud Computing (CLOUD), Jul. 2023, pp. 290–292. doi: 10.1109/CLOUD60044.2023.00041.

[20] “Exploring the Impact of Prompt Engineering on ChatGPT 3.5 Text Summarization: A BERT Score Evaluation,” Int. Res. J. Mod. Eng. Technol. Sci., Oct. 2023, doi: 10.56726/IRJMETS45268.

[21] S. Vatsal and H. Dubey, “A Survey of Prompt Engineering Methods in Large Language Models for Different NLP Tasks,” Jul. 24, 2024, arXiv: arXiv:2407.12994. doi: 10.48550/arXiv.2407.12994.

[22] X. Amatriain, “Prompt Design and Engineering: Introduction and Advanced Methods,” May 05, 2024, arXiv: arXiv:2401.14423. doi: 10.48550/arXiv.2401.14423.

[23] P. Cui and L. Hu, “Sliding Selector Network with Dynamic Memory for Extractive Summarization of Long Documents,” in Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds., Online: Association for Computational Linguistics, Jun. 2021, pp. 5881–5891. doi: 10.18653/v1/2021.naacl-main.470.

[24] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a Method for Automatic Evaluation of Machine Translation,” in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, P. Isabelle, E. Charniak, and D. Lin, Eds., Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. doi: 10.3115/1073083.1073135.

[25] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating Text Generation with BERT,” Feb. 24, 2020, arXiv: arXiv:1904.09675. doi: 10.48550/arXiv.1904.09675.

[26] S. Roller et al., “Recipes for Building an Open-Domain Chatbot,” in Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Online: Association for Computational Linguistics, 2021, pp. 300–325. doi: 10.18653/v1/2021.eacl-main.24.

[27] H. Touvron et al., “Llama 2: Open Foundation and Fine-Tuned Chat Models,” Jul. 19, 2023, arXiv: arXiv:2307.09288. doi: 10.48550/arXiv.2307.09288.

[28] T. Brown et al., “Language Models are Few-Shot Learners,” in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., Curran Associates, Inc., 2020, pp. 1877–1901.

[29] R. Anil et al., “PaLM 2 Technical Report,” Sep. 13, 2023, arXiv: arXiv:2305.10403. doi: 10.48550/arXiv.2305.10403.

[30] Z. Wang, J. Liu, Q. Bao, H. Rong and J. Zhang, ”ChatLogic: Integrating Logic Programming with Large Language Models for Multi-Step Reasoning,” 2024 International Joint Conference on Neural Networks (IJCNN), Yokohama, Japan, 2024, pp. 1-8, doi: 10.1109/IJCNN60899.2024.10650138.

[31] K. Afifi-Sabet, “AI chatbots need to be much better at remembering things. Have scientists just cracked their terrible memory problem?,” Live Science, Feb. 23, 2024. [Online]. Available: https://www.livescience.com/technology/artificial-intelligence/ai-chatbots-chatgpt-bad-at-remembering-things-have-scientists-just-cracked-their-terrible-memory-problem

[32] Wang, X., Salmani, M., Rezagholizadeh, M., Eshaghi, A., Omidi, P., & Ren, X. (2024). Beyond the Limits: A Survey of Techniques to Extend the Context Length in Large Language Models. 8299–8307. https://doi.org/10.24963/ijcai.2024/917

[33] Laban, Philippe, et al. ”Llms get lost in multi-turn conversation.” arXiv preprint arXiv:2505.06120 (2025).

[34] Velásquez-Henao, Juan David, Carlos Jaime Franco-Cardona, and Lorena Cadavid-Higuita. ”Prompt Engineering: a methodology for optimizing interactions with AI-Language Models in the field of engineering.” Dyna 90.SPE230 (2023): 9-17.

[35] Schulhoff, Sander et al. “The Prompt Report: A Systematic Survey of Prompt Engineering Techniques.” (2024).

[36] Ye, Qinyuan, et al. ”Prompt engineering a prompt engineer.” arXiv preprint arXiv:2311.05661 (2023).

[37] Wei, Jason, et al. ”Chain-of-thought prompting elicits reasoning in large language models.” Advances in neural information processing systems 35 (2022): 24824-24837.

[38] Son, M.; Won, Y.-J.; Lee, S. Optimizing Large Language Models: A Deep Dive into Effective Prompt Engineering Techniques. Appl. Sci. 2025, 15, 1430. https://doi.org/10.3390/app15031430

[39] Dong, Qingxiu, et al. ”A survey on in-context learning.” arXiv preprint arXiv:2301.00234 (2022).

[40] L. Zou, “Chapter 6 - Meta-learning for natural language processing,” in Meta-Learning, L. Zou, Ed., Academic Press, 2023, pp. 209–266. doi: 10.1016/B978-0-323-89931-4.00005-5.

[41] “A review of dialogue systems: current trends and future directions — Neural Computing and Applications.” Accessed: Aug. 04, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s00521-023-09322-1

[42] Zhang, Yang, et al. ”A comprehensive survey on process-oriented automatic text summarization with exploration of LLM-based methods.” arXiv preprint arXiv:2403.02901 (2024).

[43] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017.

[44] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. doi: 10.18653/v1/N19-1423.

[45] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” 2019. Accessed: Aug. 14, 2025. [Online]. Available: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe

[46] H. W. Chung et al., “Scaling instruction-finetuned language models,” J. Mach. Learn. Res., vol. 25, no. 1, p. 70:3381-70:3433, Jan. 2024.

[47] Y. Zhang, H. Jin, D. Meng, J. Wang, and J. Tan, “A Comprehensive Survey on Process-Oriented Automatic Text Summarization with Exploration of LLM-Based Methods,” 2024, doi: 10.48550/ARXIV.2403.02901.

[48] Z. Zhang et al., “A Survey on the Memory Mechanism of Large Language Model based Agents,” Apr. 20, 2024, arXiv: arXiv:2404.13501. doi: 10.48550/arXiv.2404.13501.

[49] ”The Probabilistic Relevance Framework: BM25 and Beyond,” ResearchGate, Aug. 2025, Accessed: Aug. 19, 2025.

[50] V. Karpukhin et al., “Dense Passage Retrieval for Open-Domain Question Answering,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. doi: 10.18653/v1/2020.emnlp-main.550.