Using ChatGPT to Augment Software Engineering Chatbots Datasets

Khaled Badran

A Thesis

in

The Department

of

Computer Science & Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Software Engineering) at

Concordia University

Montréal, Québec, Canada

December 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Khaled Badran**

Entitled: **Using ChatGPT to Augment Software Engineering Chatbots Datasets**

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

Master of Applied Science (Software 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. Jinqiu Yang

_____ Examiner
Dr. Jinqiu Yang

_____ Examiner
Dr. Nikolaos Tsantalis

_____ Supervisor
Dr. Emad Shihab

Approved by _____
Joey Paquet, Chair
Department of Computer Science & Software Engineering

_____ 2023

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Using ChatGPT to Augment Software Engineering Chatbots Datasets

Khaled Badran

Chatbots are envisioned to bring about a significant shift in the realm of Software Engineering (SE), enabling practitioners to engage in conversations and interact with various services using natural language. At the heart of each chatbot is a Natural Language Understanding (NLU) component that enables the chatbots to comprehend the user's queries. However, the NLU requires extensive, high-quality training data (examples) to accurately interpret user queries. Prior work shows that the creation and augmentation of SE datasets are resource-intensive and time-consuming. To address this gap, we explore the potential of using ChatGPT to augment the SE chatbot training dataset. Specifically, we evaluate the impact of retraining the NLU on ChatGPT's augmented dataset on the NLU's performance using four widely used SE datasets. Moreover, we assess the syntactic and semantic aspects of the generated examples compared to human-written examples. Additionally, we conduct an ablation study to investigate the impact of each component in the prompt on the NLU's performance and the diversity of the generated examples. The results show that ChatGPT significantly improves the NLU's performance, with F1-score improvements ranging from 3.9% to 11.6%. Moreover, we find that ChatGPT-generated examples exhibit syntactic diversity while maintaining consistent semantics (2.2% on average) across all datasets. Additionally, the results indicate that including a few human-written examples and a description of the intent's objective in the prompt impacts the quality of the generated examples. Finally, we provide implications for practitioners and researchers of SE chatbots.

Related Submitted Publications

The following submitted publications are related to this thesis.

- **Khaled Badran**, Ahmad Abdellatif, and Emad Shihab. "Using ChatGPT to Augment Software Engineering Chatbots Datasets". Under Submission to the 2024 International Conference on Mining Software Repositories (MSR'24).

Statement of Originality

I, Khaled Badran, hereby declare that I am the sole author of this thesis. All ideas and inventions attributed to others have been properly referenced. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Dedication

To my parents and siblings

Acknowledgments

I would like to express my gratitude to Professor Emad Shihab, whose guidance and support have been invaluable to me. I've grown tremendously by learning from your knowledge and wisdom. Working alongside you has been both an honour and a privilege.

To Dr. Ahmad Abdellatif, who's always been on my side during this journey, thank you from the bottom of my heart. We have shared countless hours of work together and created some of the most wonderful memories I have, and for that, I'm forever grateful.

To my family, whose love and encouragement have been my constant support. I owe you an immeasurable debt of gratitude. I feel very lucky for your presence in my life at every moment. I hope to continue growing, so I can make every one of you truly proud.

Lastly, to my close friends, who have filled my life with joy and stood by me in times of need, I treasure our moments together beyond words. Thank you for everything.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software chatbots perform various software engineering (SE) tasks, from answering technical questions Abdellatif, Badran, and Shihab (2020); Xu, Xing, Xia, and Lo (2017) and refactoring code Wyrich and Bogner (2019), to eliciting user feedback and assisting newcomers to a project Serrano Alves, Wiese, Chaves, and Steinmacher (2022). Practitioners can easily interact with chatbots using natural language, making them quite useful for handling tedious tasks. At the core of each chatbot lies a natural language understanding component (NLU) that enables the chatbot to comprehend the queries posed by users. In other words, the NLU extracts structural information (the user's intent) from unstructured text (user query).

To enhance the NLU's ability to extract the intent from the user's query, chatbot practitioners provide the NLU with training examples. These examples capture the various ways users might phrase the same question. For instance, the training examples: "How many commits do we have?" and "Give me the commit count" have the same intent (i.e., ask about the number of commits) but different syntax. Prior work shows that training the NLU with high-quality training examples (i.e., training data) improves its performance in correctly understanding user queries Abdellatif, Badran, Costa, and Shihab (2021), ultimately enhancing user satisfaction with the chatbot.

However, chatbot developers often have to manually craft and augment these high-quality training examples. This process is not only time-consuming but also resource-intensive. In fact, previous studies indicate that collecting and augmenting training data for the NLU is one of the most difficult tasks when building a chatbot Abdellatif, Badran, and Shihab (2020); Abdellatif, Costa, Badran,

Abdelkareem, and Shihab (2020); Dominic, Houser, Steinmacher, Ritter, and Rodeghero (2020a). Such challenges lead to a scarcity of high-quality training data, with concrete consequences in real life. For instance, Dominic et al. Dominic et al. (2020a) reported that the lack of training data limited their chatbot's performance. Similarly, the developers of the MSRBot faced difficulties in providing accurate answers to certain user queries due to the scarcity of training examples Abdellatif, Badran, and Shihab (2020). In light of the struggles associated with augmenting training examples, and their impact on the progress of SE chatbots, it is crucial to support chatbot developers in this task.

Recently, OpenAI released ChatGPT, which has revolutionized many fields, including SE. ChatGPT is a Large Language Model (LLM) that has been trained using Reinforcement Learning from Human Feedback (RLHF) Wu et al. (2023), helping it set new benchmarks in several NLP challenges, including the generation of human-like text OpenAI (2023a). Within the SE domain, recent studies have showcased the effectiveness of ChatGPT in various tasks, such as code generation Yetiştiren, Özsoy, Ayerdem, and Tüzün (2023), unit test generation Yuan et al. (2023), bug reproduction Joshi et al. (2022), and program repair Cao, Li, Wen, and chi Cheung (2023). This highlights ChatGPT's ability to grasp a wide range of SE contexts and their unique terminologies (e.g., 'bugs' referring to an 'issue'). Moreover, ChatGPT excels in few-shot learning scenarios OpenAI (2023a), which means that it can provide a valuable response even with limited or no prior examples. Given these qualities, ChatGPT presents as a promising solution to assist developers in augmenting their training datasets. However, its effectiveness in this specific task remains unexplored.

In this thesis, we present the first work exploring the potential of using ChatGPT to augment SE chatbot training datasets and empirically evaluating its performance on this task. To achieve this, we design a prompt comprising four components: 'Intent Name', 'Intent Definition', 'Intent Examples', and 'Task Boundaries' (conditions to ensure that ChatGPT generates diverse output) to guide ChatGPT in generating additional training examples. Next, we assess the contribution of adding the generated examples to (i.e., augmenting) the initial dataset to the NLU's performance. We use four SE datasets in our evaluation: 1) Repository, focusing on project-related data exploration (e.g., "Which developer fixed more bugs in TopicPartitionTest file"), 2) Stack Overflow, encompassing software development questions commonly asked and answered by practitioners (e.g., "How to

load data from Excel file to SQL database?"), 3) Ask Ubuntu, containing the most popular questions from the Ubuntu Q&A community on Stack Exchange (e.g., "How to upgrade Ubuntu 9.10 to 12.10 via terminal?"), and 4) MSA, which covers Microservices environment questions (e.g., "I want to know the health data for my server."). Additionally, we examine the quality of the generated examples both in terms of their similarity to human-written ones, and whether they preserve the intent used in the prompt. Moreover, we investigate the impact of each prompt component on the generated examples by ChatGPT.

1.1 Thesis Overview

The remainder of the thesis is organized as follows. Chapter 2 reviews literature and studies that are related to our area of research, including chatbots in software engineering, data augmentation, and the application of ChatGPT in this field. Chapter 3 provides an overview of chatbots and explains related concepts used throughout this thesis. Chapter 4 outlines the methodology employed in our work, including the selection of datasets, the process of prompt engineering for ChatGPT, and the use of the ChatGPT model in conjunction with the Rasa NLU platform. Chapter 5 presents our main results in three parts:

- Chapter 5.1 attempts to answer our first research question **"Can ChatGPT's generated examples improve the NLU's performance?"** We find that ChatGPT generates, on average, 10 new examples per intent. Training the NLU using the augmented examples significantly improves its performance, with improvement ranging from 3.9% to 11.6% in F1-score.

- Chapter 5.2 addresses the question **"How well do the generated examples reflect real-world examples?"** Here we find that ChatGPT's generated examples show syntactic differences when compared to human-written ones. Specifically, examples from the Stack Overflow dataset stand out as the most distinct, with a BLEU-1 score of 28.5%, while those from the Repository dataset closely mirror human examples, with a BLEU-1 score of 48.9%. Nonetheless, 92.2% of generated examples preserve the intent on average. This highlights ChatGPT's ability to preserve the intent while introducing syntactic diversity.

- Chapter 5.3 tackles the research question **"What factors influence the performance of ChatGPT when generating examples?"** We find that the 'Intent Definition' and 'Intent Examples' prompt components are important to generate useful examples that improve the NLU's performance. Additionally, including 'Intent Examples' in the prompt leads to a higher similarity between ChatGPT's generated examples and human-written ones.

Chapter 6 dives deeper into our findings, examining the effect of different temperature settings on the NLU's performance and exploring the impact of using only ChatGPT-generated examples for training the NLU. This chapter concludes with a discussion of the broader implications of our research for practitioners and researchers. Chapter 7 discusses potential limitations and threats to validity in this thesis, while Chapter 8 concludes the thesis and presents future research directions.

1.2 Thesis Contributions

This thesis makes the following contributions to practitioners and the research community:

- To the best of our knowledge, this is the first work that investigates the use of ChatGPT for augmenting SE chatbot training datasets.
- We conduct an empirical evaluation across four distinct SE datasets, offering insights into ChatGPT's generative capabilities in diverse contexts for the SE domain.
- We provide actionable recommendations to SE chatbot practitioners aiming to harness ChatGPT in their workflows.
- We make our replication package publicly available to enable replication and encourage further studies by the chatbot community Zenodo (2023).

Chapter 2

Related Works

Since the goal of this thesis is to explore the potential of using ChatGPT in augmenting training datasets for SE chatbots, we discuss the related work in three areas: work that studies chatbots in the SE domain, work that focuses on dataset augmentation, and work that leverages ChatGPT to perform various SE tasks.

2.1 Dataset Augmentation

Many researchers evaluate approaches that augment datasets for text classification Amin-Nejad, Ive, and Velupillai (2020); S. Y. Feng, Gangal, Kang, Mitamura, and Hovy (2020); Imran, Jain, Chatterjee, and Damevski (2023); Marivate and Sefara (2020); Rizos, Hemker, and Schuller (2019); Wei and Zou (2019). For example, S. Y. Feng et al. (2020) evaluated different approaches, such as random insertion and semantic text exchange, to augment text data using Yelp Reviews dataset. The results demonstrated that augmentation methods improve GPT-2's performance in various aspects of text generation. For instance, the Type-Token Ratio (TTR) score improved from 0.7173 to 0.7420 after augmentation.

Amin-Nejad et al. (2020) evaluated the use of Transformer models for augmenting medical datasets, particularly datasets related to discharge summaries of patients. The authors experiment with state-of-the-art Transformer models, including GPT-2, to generate new training data and evaluate their effectiveness in enhancing two clinical NLP tasks: unplanned readmission prediction

and phenotype classification. The study demonstrates that the generated data, while not as high in quality as the original, can significantly improve performance in specific tasks like readmission prediction, especially when used with models like BioBERT that are pre-trained on biomedical texts.

Imran et al. (2023) tackled the challenge of data scarcity related to emotion recognition tasks in GitHub issues and pull requests. The authors focused on enhancing the performance of SE-specific emotion classifiers by employing data augmentation techniques. They propose and evaluate three different augmentation strategies (Unconstrained, Lexicon-based, and Polarity-based) to generate augmented training data. These strategies aim to enhance the emotional polarity of text while introducing more diverse syntax. The study demonstrates an average improvement of 9.3% in micro F1-Score across three existing emotion classification tools when trained with the Polarity-based augmentation strategy.

Sharifirad, Jafarpour, and Matwin (2018) proposed an approach that generates new training examples by replacing words in the dataset with their synonyms and definitions using ConceptNet and Wikidata. They evaluated their approach to the sexiest tweet dataset using Long Short-Term Memory (LSTM) Networks and Convolutional Neural Networks (CNNs). The results show that augmenting data by replacing all words in the dataset improves the models' classification.

Marivate and Sefara (2020) investigated the impact of different text augmentation methods on short text classification. The authors employed three datasets, including social media and formal news articles, to study the effects of four augmentation approaches, namely, WordNet-based synonym, Word2vec-based, Round Trip Translation, and mixup augmentation. They found that the mixup strategy leads to improved performance across all text-based augmentations and reduces overfitting in deep learning models.

Rizos et al. (2019) developed novel data augmentation techniques for short text classification, particularly for hate speech detection. The study proposed three augmentation methods: synonym replacement based on word embedding vector closeness, warping of word tokens along the padded sequence, and class-conditional recurrent neural language generation. They evaluated their approach using several deep-learning architectures and hate speech databases. Their results indicated that the proposed augmentation techniques significantly improve hate speech detection performance. The most notable improvements were observed in multi-class hate speech detection,

where their proposed framework outperformed the baseline models, achieving a 5.7% increase in Macro F1-score.

The previous studies focused on comparing different augmentation approaches for text classifications. Our work differs from and complements the prior work in two ways. First, we concentrate on augmenting training datasets for SE chatbots. Second, the main goal of this thesis is to explore the potential of using ChatGPT to augment SE datasets. Thus, it helps chatbot practitioners by providing easy accessibility and not requiring any pretraining, distinguishing it from other approaches suggested in prior work. This distinction is particularly significant given the scarcity of SE benchmarks for intent classification. Finally, our work complements the studies augmenting text and contributes to that body of research by providing a thorough evaluation of ChatGPT's performance in augmenting SE datasets.

2.2 ChatGPT for Software Engineering

Recently, several studies have explored the potential of using ChatGPT for various Software Engineering tasks Ahmed et al. (2023); Ebert and Louridas (2023); Y. Feng et al. (2023); Maddigan and Susnjak (2023); Ozkaya (2023). Ahmed et al. (2023) conducted a large-scale study at Microsoft exploring the usefulness of large language models (like ChatGPT) in the context of incident management for cloud services. The researchers leveraged several LLMs from OpenAI to root-cause and suggest mitigation steps on more than 40,000 cloud incidents at Microsoft. The incident owners were then asked to evaluate the usefulness of the responses generated by the LLMs. The findings reveal that the majority (over 70%) of incident owners find the LLM response to be useful for root-causing and mitigating cloud-related incidents.

Y. Feng et al. (2023) investigated the code generation performance of ChatGPT using a crowdsourcing approach, where they analyzed 316K relevant tweets and 3.2K Reddit posts. The results highlight that users leverage ChatGPT to generate code in various programming languages, with Python being the most popular. Using LDA topic modelling, the researchers report 17 distinct topics where ChatGPT code generation ability is applied, including debugging, testing, and interview preparation. Finally, the study highlights that the attention on ChatGPT remained consistent over

time, indicating that users find ChatGPT to be helpful for code generation.

Maddigan and Susnjak (2023) proposed an end-to-end approach called NL2VIS, which converts natural language examples into visualizations. More specifically, NL2VIS leverages Large Language Models (e.g., ChatGPT) to generate Python scripts that visualize data based on the user's queries. The results show that NL2VIS outperforms state-of-the-art tools that use symbolic NLP and deep learning techniques to visualize data.

J. Zhang, Chen, Niu, Wang, and Liu (2023) conducted an empirical evaluation of ChatGPT's performance in retrieving requirements information from SE documents. The study focused on zero-shot settings, where ChatGPT was not pre-trained or fine-tuned for specific requirements engineering (RE) tasks. The evaluation framework designed by the authors includes a combination of two popular information retrieval (IR) tasks and two common types of artifacts used in RE (e.g., software requirements specification). The results show that ChatGPT's performance in classifying requirements statements matches that of the baseline with an average F Measure of 0.84, indicating a strong capability in understanding and categorizing requirements. However, when extracting features from app descriptions, ChatGPT exceeded the baseline's average F Measure, scoring 0.75 against the baseline's 0.54, demonstrating a significant improvement in identifying key features from app descriptions.

Sun et al. (2023) evaluated the performance of ChatGPT in automatic code summarization tasks compared to state-of-the-art code summarization models. The authors used a Python dataset from the CodeSearchNet corpus to conduct this evaluation. The findings indicate that although ChatGPT is capable of performing code summarization and generating detailed comments, it falls short in BLEU and ROUGE-L metrics compared to leading models. For example, ChatGPT achieved a BLEU score of 10.28 and a ROUGE-L score of 20.81, whereas CodeT5 scored 20.0 and 37.7 in these metrics, respectively. However, in terms of the METEOR score, ChatGPT's performance (14.4) is nearly on par with other models, like CodeT5, which scored 14.7. The paper highlights ChatGPT's limitations and opportunities for improvement in generating concise, relevant comments for code summarization in SE.

Li et al. (2023) explored the effectiveness of ChatGPT in identifying failure-inducing test cases in software programs. Initially, the study reported that ChatGPT was only 28.8% successful in locating the correct failure-inducing test cases for buggy programs. The low performance was attributed to ChatGPT's struggle in distinguishing certain nuances that often lead to bugs, such as subtle code differences. To address this weakness, the authors introduced a novel approach called 'Differential Prompting'. This approach aims to help ChatGPT with the task of identifying failure-inducing test cases by dividing it into three sub-tasks: program intention inference, program generation, and differential testing. The study then compared this approach to conventional ChatGPT prompting and the state-of-the-art Python unit test generation tool (i.e., PYNGUIN). The results reveal that 'Differential Prompting' significantly outperforms other techniques in finding failure-inducing test cases. For instance, when applied to QuixBugs programs, 'Differential Prompting' achieved a success rate of 75.0%, outperforming the best baseline by 2.6 times. Furthermore, for Codeforces programs, the success rate of 'Differential Prompting' was 66.7%, which is 4 times higher than the baseline.

Our work both differs and complements prior studies in several ways. First, we explore the application of ChatGPT to augment the datasets of SE chatbots. To the best of our knowledge, this is the first work to employ ChatGPT for this domain. Furthermore, our work contributes to the body of work on ChatGPT's generative abilities in SE by providing insights into the performance of ChatGPT in various SE contexts (e.g., repository-related questions, etc). Additionally, our findings shed light on some challenges that need to be addressed by the community.

2.3 Software Engineering Chatbots

A number of studies propose chatbots to support practitioners in their development tasks Abdellatif, Badran, and Shihab (2020); Serrano Alves et al. (2022); Wolfinger, Fotrousi, and Maalej (2022); N. Zhang et al. (2022).

Wolfinger et al. (2022) developed a chatbot designed to elicit contextual information from user feedback on software applications. This chatbot, built using the Rasa NLU platform, aims to address the challenge of collecting valuable, informative, and actionable feedback from user responses,

which are often vague. Lastly, this chatbot significantly streamlines the feedback process by generating structured reports for development teams, thereby enhancing the review and prioritization of user feedback. The chatbot can elicit 13 different contextual information items from user conversations, significantly reducing the time and resources needed to review user feedback. In a case study with a large German company, the chatbot accurately detected 96.8% of contextual information, compared to only 31.3% in traditional surveys. Moreover, the manual effort required for extracting or verifying this information with the chatbot was about four times less than that required for surveys.

Serrano Alves et al. (2022) proposed a chatbot that assists newcomers in selecting appropriate tasks in Open Source Software (OSS) projects. The chatbot filters the available tasks to match the users' skills. To evaluate the chatbot, the study involved 40 participants and compared the chatbot with the traditional GitHub issue tracker interface. The results highlighted that users found the chatbot to be easier to use than the GitHub issue tracker, with a mean perceived ease of use score of 4.55 for the chatbot compared to 4.18 for the GitHub issue tracker. Furthermore, users indicated that the chatbot was particularly beneficial for newcomers and less experienced contributors.

Saini, Mussbacher, Guo, and Kienzle (2020) developed a chatbot, called DoMoBOT, that generates domain models from problem descriptions in natural language. Moreover, it allows modellers to interact with the bot for swift updates to the extracted domain models.

N. Zhang et al. (2022) developed Chatbot4QR, a chatbot that assists practitioners in refining their search queries on Stack Overflow to retrieve more relevant questions to the user. More specifically, the chatbot detects missing technical details in queries and interacts with users to help clarify these missing details. This work evaluated Chatbot4QR through six user studies with 25 participants. The results indicated that Chatbot4QR responds quickly to participants, with an average response time of 1.3 seconds. Moreover, for 48% to 88% of the given tasks, interacting with Chatbot4QR led to more desired results compared to using standard search engines, such as Google and Stack Overflow.

Abdellatif, Badran, and Shihab (2020) developed a chatbot, called MSRBot, that answers software repository-related questions. The proposed chatbot was designed using five components: user

interaction, entity recognizer, intent extractor, knowledge base, and response generator. The effectiveness of the bot was validated through a case study involving 12 participants. The study reported that the MSRBot significantly outperformed manual methods in terms of helping the participant complete the tasks (i.e., task completion rate). The chatbot also helped the participants to complete the tasks faster (i.e., shorter completion time). The majority of the participants (90%) found the MSRBot chatbot to be useful or very useful.

Xu et al. (2017) introduced AnswerBot, which assists software developers in finding relevant information. More specifically, AnswerBot automatically generates summaries of answers to developers' technical questions from Q&A sites like Stack Overflow. The chatbot was designed to retrieve relevant questions, select useful answer paragraphs, and generate diverse answer summaries. To validate the effectiveness of their approach, the authors conducted user studies using a dataset of Java-related questions and their corresponding answers. The results demonstrated AnswerBot's ability to provide relevant, useful, and diverse summaries, with mean relevance, usefulness, and diversity scores of 3.450, 3.720, and 3.830 respectively, outperforming conventional search methods.

The growing interest in developing chatbots to support developers in their daily tasks inspires our work, with the goal of assisting practitioners in augmenting datasets to train their chatbots. We believe that our work highlights the usefulness of using ChatGPT in augmenting SE training datasets. Thus, chatbot practitioners can focus on the core functionalities of their chatbots rather than augmenting the training dataset.

Chapter 3

Background

3.1 NLUs and Training Data

A vital component to any chatbot is the Natural Language Understanding (NLU) component. The NLU takes the user's input, identifies the intent behind it, and then guides the chatbot to the next steps. For example, if a developer asks, "How many commits were introduced in the last week?", the NLU's job is to determine that the user is inquiring about the number of commits in a specific period. Once this intent is understood, the chatbot can provide an appropriate response or solution. The ability of the NLU to accurately interpret the intent behind user's message relies heavily on the quality of the training dataset. In other words, the better the training examples, the better the NLU becomes at classifying the user's intent.

Figure 3.1 illustrates a typical training dataset for an NLU. The dataset consists of various intents, each accompanied by multiple examples. These examples reflect the diverse ways a user might express a particular intent. For instance, to train the NLU to identify the intent 'GetNumberOfCommitsByDate', the dataset includes training examples such as "Determine the number of commits that happened in the past 30 days" and "Number of commits last year". These examples both ask about the same intent ('GetNumberOfCommitsByDate') but differ syntactically. The variety in the training data allows the NLU to capture the intent behind the user's message, regardless of how it's phrased.

To obtain training datasets for chatbots, practitioners often utilize publicly available datasets

for their chatbot domain. Additionally, where data is scarce, chatbot developers often resort to crafting the training examples manually, a process that consumes much of their time and effort. Lastly, chatbot developers can augment their training dataset using examples from authentic user interactions with the chatbot.

training_data.yml

```yaml
- intent: GetNumberOfCommitsByDate
  examples: |
    - Determine the number of commits that happened in the past 30 days
    - How many commits do I have between 27/5/2018 - 31/5/2018?
    - Number of commits last year

- intent: FixingCommitsForBugTicket
  examples: |
    - Show fix of KAFKA-7354
    - Highlight the changes that resolved KAFKA-1859
    - Fetch me a list of commits that removed the problem KAFKA-1922
```

Figure 3.1: NLU Training Data.

3.2 ChatGPT

ChatGPT is a large language model from the GPT (Generative Pre-trained Transformer) family of models, developed by OpenAI. This model has shown excellent performance in the domain of text generation OpenAI (2023a). ChatGPT's proficiency lies in its ability to generate coherent, contextually relevant text, making it a promising tool for diverse applications.

ChatGPT employs a transformer architecture, which means it uses layers of attention mechanisms to capture contextual information from input data over various distances. Nonetheless, what sets this model apart is its training methodology. It was trained using Reinforcement Learning from Human Feedback (RLHF). In this approach, initial model outputs are ranked by humans, and the model then fine-tunes itself based on this feedback to produce better results. This method ensures that the model generates outputs that are closely aligned with human preferences, making it more effective and context-aware.

OpenAI facilitates interactions with ChatGPT through both an API and a UI, providing users

with flexibility in their mode of engagement. However, it's worth noting that ChatGPT's output is not deterministic, meaning that repeated queries may result in different responses.

Within the software engineering landscape, ChatGPT has also been established as an important tool that can help practitioners in various tasks such as code and test generation. In this thesis, we aim to further explore ChatGPT's effectiveness in generating training examples for SE chatbots.

Chapter 4

Methodology

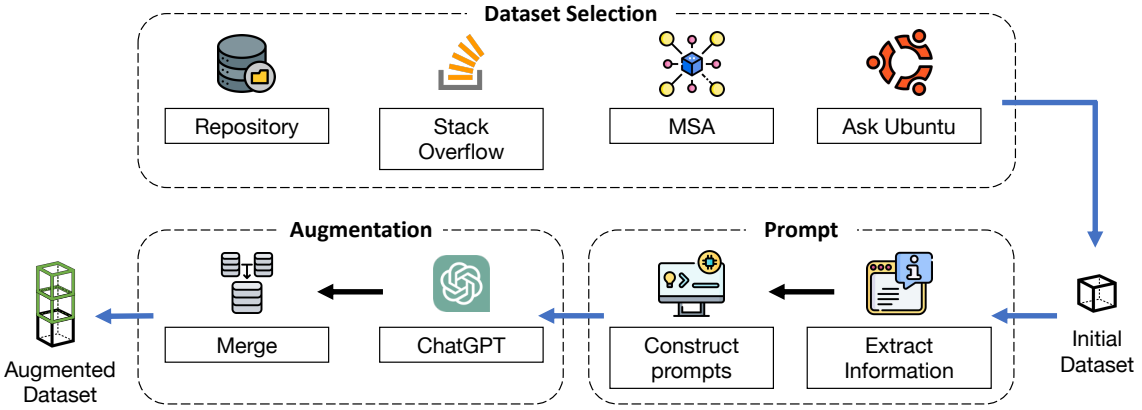

Figure 4.1: Overview of the methodology. Only one dataset (e.g., Repository) is selected for augmentation at a time.

The goal of this thesis is to empirically explore ChatGPT's potential in augmenting SE chatbots' training datasets. Figure 4.1 presents an overview of the case study we conduct in this thesis. We start by selecting four human-written datasets for SE chatbots. From here, we choose one dataset to augment at a time. We derive essential information from the dataset (e.g., intent definition) to formulate prompts for ChatGPT. These prompts are designed to guide ChatGPT in producing new training examples that complement the existing dataset. We then merge these ChatGPT-generated examples with the initial (human-written) dataset. Lastly, we evaluate the effectiveness of this

augmentation process based on the performance gains observed on an NLU platform. This section outlines the SE datasets employed in our evaluation, details the specific prompt we send to ChatGPT, and explains the experimental settings.

4.1 Dataset

To assess the potential of ChatGPT in augmenting SE datasets, we choose four datasets, namely, Repository, Stack Overflow, Ask Ubuntu, and MSA that have been used in prior work Abdellatif et al. (2021); Abdellatif, Badran, and Shihab (2020); Braun, Hernandez Mendez, Matthes, and Langen (2017); Lin, Ma, and Huang (2020). Furthermore, the selected datasets represent questions that cover various SE tasks, such as working with software repositories or managing microservices. Table 4.1 presents the intents, their corresponding definitions, and the number of training examples associated with each intent in the four datasets. In what follows, we provide a description of each dataset:

- **Repository:** This corpus is used to train the MSRBot Abdellatif, Badran, and Shihab (2020), a chatbot designed to answer developers' questions about their software repositories, such as "What commits resolved the bug ticket KAFKA 1521" from the 'FixingCommitsForBugTicket' intent. The dataset contains both the examples written by the MSRBot developers, and queries posed by users to evaluate the MSRBot. In total, the dataset contains ten intents and 398 training examples.

- **Stack Overflow (SOF):** This dataset captures a range of software development questions that developers ask. Ye et al. (2016) collected questions posted under the most popular tags on Stack Overflow. Subsequently, Abdellatif et al. (2021) labelled these questions (used later as training examples) and categorized them into different intents, one of which is 'FacingError'. This intent encompasses use cases where developers encounter exceptions and are seeking solutions (e.g., "PHP mysqli query returns empty error message"). The StackOverflow dataset comprises five intents and 215 examples.

- **Ask Ubuntu:** Braun et al. (2017) collected some of the most popular questions from the

Table 4.1: Intent names, definitions and number of examples.

Dataset	Intent	Definition	Examples
Repository	BuggyCommitsByDate	Present the buggy commit(s) which happened during a specific time period.	79
	IntroducedBugsByCommit	Identify the bugs that are introduced because of certain commits.	61
	MostFileContainsBugs	Determine the most buggy files in the repository to refactor them.	50
	FixingCommitsForBugTicket	Identify the commit(s) which fix a specific bug.	42
	FixingCommitsIntroducedBugs	Identify the fixing commits that introduce bugs at a particular time	39
	GetNumberOfCommitByDate	Identify the number of commits that were pushed during a specific time period.	32
	GetDevelopersWho-HasExperianceToFixBug	Identify the developer(s) who have experience in fixing bugs related to a specific file.	29
	MostDeveloperHasOpenedBugs	Determine the overloaded developer(s) with the highest number of unresolved bugs.	24
	GetCommitsInfoAbout-SpecificClass	View details about the changes that occurred on a file.	22
	GetCommitBodyByDatePeriod	Present the commit information (e.g., commit message) at a specific time.	20
SOF	LookingForCodeSample	Looking for information related to implementation (e.g., code snippets).	132
	UsingMethodImproperly	An improper use of a method is causing unexpected behaviour.	51
	LookingForBestPractice	Looking for the recommended (best) practice, approach or solution for a problem.	12
	FacingError	Facing an error or a failure in a program, mostly in the form of an error message.	10
	PassingData	Passing data between different frameworks or method calls.	10

Dataset	Intent	Definition	Examples
Ask Ubuntu	SoftwareRecommendation	Questions seeking suggestions for specific software or tools that can be used in Ubuntu to perform certain tasks or meet specific needs.	57
	MakeUpdate	Questions related to upgrading or updating software or versions in Ubuntu, specifically regarding the decision to upgrade or steps involved in upgrading.	47
	ShutdownComputer	Questions about the proper methods and techniques to shut down the computer in Ubuntu, including considerations for different scenarios and user sessions.	27
	SetupPrinter	Questions about configuring and setting up printers in Ubuntu, including troubleshooting printer recognition, wireless setup, and instructions for specific printer models.	23
MSA	ServiceInfo	Browsing a service's information.	12
	ServiceUsingInfo	Tracking the usage of a service.	14
	ServiceApiList	Reading the API documentation for a service.	11
	ServiceEnv	Viewing t+he environmental settings of a service.	8
	ServiceHealth	Checking the service health status.	10
	ServiceDependencyGraph	Viewing the service dependency graph for all Microservices in a project.	10
	ServiceOnly	Setting and viewing the service for a Microservice project	12
	LastBuildFail	Understanding the reason behind the build failure for a service.	6

Ubuntu Q&A community on Stack Exchange, a popular online discussion forum. Then, they annotated the intent of those questions through Amazon Mechanical Turk. An instance of a training example from this dataset is "How to upgrade Ubuntu 9.10 to 12.10 via terminal?" from the 'MakeUpdate' intent. This dataset includes 154 examples, split into four intents (e.g., 'Make Update'). We excluded the 'Other' intent from our evaluation due to the insufficient number of examples (i.e., three) for our evaluation.

- **MSA:** Lin et al. (2020) introduced the MSA dataset, used to train the MSABot, a chatbot to assist with developing Microservice architectures. Practitioners can ask the MSABot about their microservice environment settings (i.e., 'ServiceEnv') using a query like "Tell me the server's environment setting". This dataset consists of eight intents captured using 83 training examples.

4.2 Prompt Engineering

Prompt engineering refers to the way of phrasing the question to large language models, and it plays a critical role in the quality of the model's response Reynolds and McDonell (2021). To design the prompt for our evaluation, we follow the best practices outlined in the official OpenAI guidelines for creating prompts OpenAI (2023b). More specifically, OpenAI's guidelines recommend splitting complex tasks into simpler subtasks. Therefore, we split the complex task of augmenting a full dataset into simpler tasks of augmenting one intent at a time. The guidelines also highlight that the prompt can incorporate instructions on the required actions for ChatGPT, contextual information regarding the task, input data to steer the model, and output indicators. Following these instructions, we design our prompt to include the following components:

- **Intent Name**: OpenAI recommends incorporating any important context in the prompt to obtain more relevant outcomes. The first obvious information is the intent name. It acts as a unique identifier for that intent (similar to a variable name in programming) and often captures the goal of the intent (e.g., 'LookingForBestPractice').

- **Intent Definition**: A comprehensive description of the intent's objective, clarifying the examples it is designed to answer. The definition of the intent provides more contextual information to the large language model. In this thesis, we use the intent name and definition as defined in the original studies Abdellatif et al. (2021); Abdellatif, Badran, and Shihab (2020); Braun et al. (2017); Lin et al. (2020) that construct the datasets. Notably, when intent definitions were not available, the first author formulated the definitions based on the intent's training examples. These definitions were subsequently reviewed by the second author for accuracy.

- **Intent Examples**: Training examples for the intent written by the developers of the chatbot. These examples help the model better comprehend the intent and generate similar training data. Prior work refers to the addition of this component as a few-shot scenario Reynolds and McDonell (2021), which involves giving the large language models a sample of the expected output.

- **Task Boundaries**: To ensure that the ChatGPT model generates a diverse set of examples that are relevant to the intent, we add the task parameter component. More specifically, the task boundaries explicitly instruct the model to yield diverse output while preserving the intent. (i.e., "Ensure the generated examples maintain the intent and are diverse.").

Lastly, we follow the recommended practice of specifying the output format by adding the sentence "Return the answer as a numbered list." to the prompt. A template of the complete prompt is presented here:

```
Generate training examples for the intent {Intent_name}. The
    intent is described as follows: {intent_description}. Here are
    some pre-existing training examples for this intent:
     - {training_example_1}
     - {training_example_2}
     - ...

Ensure that the generated examples are diverse, relevant, and
    aligned with the intent description and the provided examples.

Return the answer as a numbered list.
```

4.3 ChatGPT Model

To conduct our evaluation, we chose Open-AI's GPT-3.5-turbo as our ChatGPT model, primarily because of its robustness and proficiency in understanding and generating natural language OpenAI (n.d.). Notably, GPT-3.5-turbo is the model utilized when users interact with ChatGPT through its user interface (UI). This means that our work closely mirrors the experiences and outcomes that practitioners can expect when using ChatGPT through the UI. Furthermore, the model was trained on data up to September 2021, and it has been used in similar work Dong, Jiang, Jin, and Li (2023); Tihanyi et al. (2023). We access the model via its API, as it allows for a higher number of requests. Moreover, using this model is cost-effective, which encourages replication of our work by other researchers. Lastly, we use the default parameters recommended by OpenAI when interacting with the model.

4.4 Rasa NLU platform

There are many NLU platforms that seamlessly integrate with third-party applications, such as Google's Dialogflow and IBM Watson. Among these options, we select Rasa v3.5.4 in our evaluation, which was the latest version available during our evluation. Our decision to employ Rasa stems from its popularity and extensive adoption within both the research and practitioner communities Dominic, Houser, Steinmacher, Ritter, and Rodeghero (2020b); Lin et al. (2020). Rasa stands out as an open-source and free NLU solution that can be easily installed and run locally. This ensures that the Rasa's implementation remains consistent throughout our evaluation, unlike other NLUs. Thus, it facilitates the replication of our work by other researchers. Additionally, prior work shows that Rasa delivers performance comparable to other NLU platforms (e.g., Dialogflow) Abdellatif et al. (2021). In our evaluation, we use the default pipeline recommended by Rasa when training the NLU model.

Chapter 5

Results

In this section, we will discuss the motivation, approach and results of each research question.

5.1 RQ1: Can ChatGPT's generated examples improve the NLU's performance?

Motivation: Prior work shows that crafting a training dataset for SE chatbots is a tedious task that consumes significant resources and requires costly effort Abdellatif et al. (2021); Abdellatif, Badran, and Shihab (2020); Dominic et al. (2020a). This often results in insufficient training data, hindering the chatbot's capability to comprehend users' queries effectively. ChatGPT shows potential in automating a range of SE tasks Ahmed et al. (2023); Ebert and Louridas (2023); Y. Feng et al. (2023); Ozkaya (2023). In this research question (RQ), we set out to examine the potential of using ChatGPT to augment SE training datasets, and how this impacts the performance of the chatbot's NLU. In addition to resource savings, the automated augmentation of training examples lowers the barrier to entry for developing SE chatbots.

Approach: To answer this research question, we conduct a case study that emulates a real-life scenario where chatbot developers have a few initial training examples (i.e., initial dataset) and want to further augment their training dataset to enhance the NLU's performance. Therefore, we randomly select three training examples per intent to obtain an initial dataset. Notably, we reserve any examples not selected for the initial dataset for a separate, held-out test set. Then, we use

Table 5.1: The number of generated examples per intent by ChatGPT.

Dataset	Repository	SOF	Ask Ubuntu	MSA
Min	5.0	5.0	10.0	5.0
Max	20.0	25.0	20.0	20.0
Mean	9.9	10.2	10.2	10.0
Median	10.0	10.0	10.0	10.0

the initial dataset to construct the prompts for each intent, as discussed in Section 4.2. Next, we query ChatGPT using the filled prompts to generate new training examples. Finally, we merge (i.e., augment) the generated examples with the initial dataset to obtain the augmented dataset.

To explore the potential of ChatGPT in augmenting the training datasets for SE chatbots, we investigate the performance gain of the NLU. Specifically, we train the NLU on the augmented dataset and evaluate its performance using the held-out test set, considering the weighted average F1-score as our evaluation metric. Notably, when measuring the contribution of the generated examples to the NLU's performance, we utilize the same test set to evaluate both the initial and augmented examples. Given that ChatGPT responses might vary for the same prompt, we conduct our experiment 50 times to mitigate the influence of this randomness on our findings and report the average of all runs. Additionally, we employ a nonparametric unpaired Mann-Whitney U test to assess the statistical significance of our results. We follow the same evaluation procedure for the four datasets discussed in Section 4.1.

Results: Table 5.1 presents statistics of generated examples per intent for all datasets. The results show that ChatGPT can generate at least 5 and up to 25 new examples for each intent. Chatbot practitioners can benefit from having multiple generated examples, as they can select the most suitable subset to augment their dataset with. Interestingly, the number of examples augmented by ChatGPT remains consistent across the majority of intents and all datasets (a median of 10 examples per intent). This occurs despite not specifying the number of examples to generate in our prompt, which highlights a level of consistency and stability in the generative process of ChatGPT. Nonetheless, the consistent number of examples might also imply potential internal constraints in diversity or creativity, as the model seems to default to a specific pattern of generation.

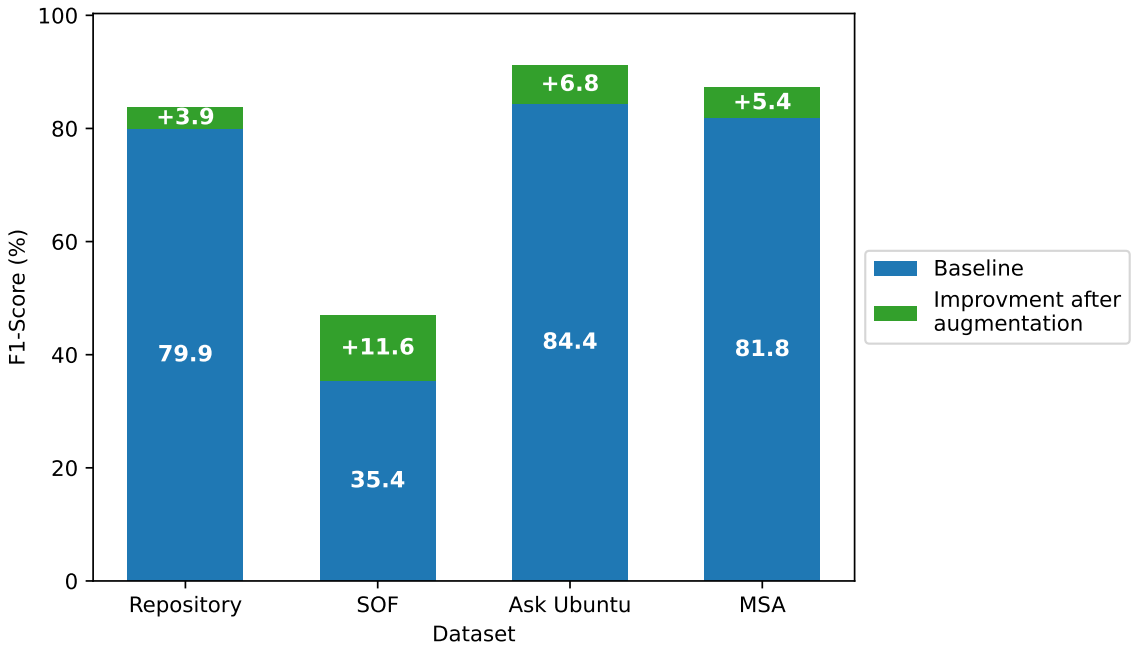

Figure 5.1: Improvement to NLU's performance (F1-score) when using augmented examples.

Table 5.2 presents a sample of the human-written examples (from the initial dataset) and ChatGPT's generated examples, alongside their corresponding intents for all datasets. The examples in the table highlight ChatGPT's ability to generate training examples that are syntactically different from the initial examples. For instance, the intent 'FixingCommitsForBugTicket' from the Repository dataset has the initial example "What commits resolved the bug ticket KAFKA 1521" and ChatGPT generated the example "Tell me which commit(s) resolved the bug ticket PQR-654" for this intent.

Figure 5.1 presents the improvement in the NLU's performance when trained on the augmented examples (depicted in green), contrasting it with the NLU's performance without augmentation. From the figure, we observe that using the augmented examples by ChatGPT improves the NLU's performance across all datasets. The performance gain ranges from 3.9% in the Repository dataset to 11.6% in the Stack Overflow dataset. Moreover, the improvement in the NLU performance across all datasets is statistically significant, as confirmed by a Mann-Whitney U-test ($p - value < 0.05$), which underscores the reliability of the results. Our findings highlight that ChatGPT can generate new examples that contribute to the NLU's performance, reducing the manual efforts required in

creating training examples.

ChatGPT consistently generates an average of 10 examples per intent. Training the NLU using the examples generated by ChatGPT enhances the NLU's performance across all datasets, with performance gains ranging from 3.9% to 11.6%. Our findings underscore the effectiveness of using ChatGPT to augment datasets for SE chatbots.

5.2 RQ2: How well do the generated examples reflect real-world examples?

Motivation: NLU providers recommend training the NLU on examples that exhibit syntactic diversity while possessing similar semantics Dialogflow (n.d.). Therefore, in this RQ, we first measure the similarity between human-written examples and those generated by ChatGPT. This analysis will examine whether ChatGPT simply duplicates the examples written by humans, or does in fact generate new examples. Moreover, we measure whether the generated examples preserve (or deviate from) the intent from the prompt. This analysis aims to give chatbot practitioners insight into the quality of output they can expect when using ChatGPT's generated examples.

Approach: To evaluate the similarity between the examples generated by ChatGPT and those in the initial training dataset (i.e., crafted by humans), we apply two similarity metrics on the output of RQ1. These metrics have been used in prior work Ahmed et al. (2023); Mastropaolo et al. (2021), and are defined as follows:

(1) **Bilingual Evaluation Understudy (BLEU):** BLEU metric measures the similarity between machine-generated and human-written text by counting the matching n-grams between them. The resulting BLEU score ranges between 0% and 100%. A lower BLEU value suggests that the generated examples are syntactically distinct from the initial (i.e., human-written) examples. In contrast, higher BLEU values indicate greater similarity between the generated and human-written examples. Prior work in SE often uses the BLEU-4 variant Ahmed et al. (2023); Mastropaolo et al. (2021); however, some examples in our datasets have less than four

Table 5.2: A sample of human-written and ChatGPT-generated examples.

Dataset	**Intent**	**Human-Written Examples**	**ChatGPT-Generated Examples**
Repository	FixingCommitsForBugTicket	What commits resolved the bug ticket KAFKA 1521	- Tell me which commit(s) resolved the bug ticket PQR-654 - Which commits addressed the issue STU-321? - List all the commits that resolved the bug ticket ABC-123
SOF	LookingForCodeSample	I need to get Facebook photo stream into my application using their Facebook id javascript	- How can I implement a login system in Python using Flask and SQLAlchemy? - Is there a way to implement a pagination system in AngularJS for displaying large datasets? - How can I validate an email address using regular expressions in C#? Any code snippets available?
Ask Ubuntu	MakeUpdate	How to upgrade Ubuntu 9.10 to 12.10 via terminal?	- Can you provide a step-by-step guide on upgrading Ubuntu Server from version 18.04 to 20.04? - How can I upgrade from Ubuntu 20.04 to Ubuntu 20.10 using the command line? - Can you guide me on upgrading Ubuntu 16.04 LTS to the latest version?
MSA	ServiceEnv	Tell me the server's environment setting.	- What is the current environment setup of the server? - I would like to know the environment setting of the server. - Can you provide me with the server's environment details?

tokens (e.g., "API list" training example from the 'ServiceApiList' intent), which presents a limitation when computing this variant. Thus, we compute four variants of the BLEU metric, namely, BLEU-1, BLEU-2, BLEU-3, and BLEU-4.

(2) **Recall-Oriented Understudy for Gisting Evaluation (ROUGE):** Measures the overlap of words, phrases, or sentences between the machine-generated and a human-written (reference) text and has been used by prior work in SE Ahmed et al. (2023); Mastropaolo et al. (2021). The ROUGE score, like BLEU, ranges from 0% to 100%. A higher percentage indicates a closer resemblance between the generated and initial (human-written) examples. In our evaluation, we use ROUGE-L, which computes the longest common subsequence between the machine-generated and reference text.

First, for each intent, we compute the similarity metric (e.g., BLEU) for each pair of generated and initial examples. Next, we determine the mean similarity score for all pairs, an approach commonly referred to as pairwise mean similarity. Finally, we compute the weighted average similarity for the entire dataset, taking into consideration the number of augmented examples per intent.

in the datasets discussed in Section 4.1.

Intent Preservation. To assess how well the generated examples preserve the intent in the prompt, we randomly sample the generated examples from each dataset with a confidence level of 95% and a confidence interval of 5%. Our random sample size for each dataset yields a total of 1,367 examples: 359 for the Repository, 333 for Stack Overflow, 351 for MSA, and 324 for Ask Ubuntu. The first author annotates the sampled examples to determine if they uphold the intent (i.e., the intent used in the prompt to generate the sampled examples). Then, the second author randomly samples 50 examples from each dataset (totalling 200 examples) to ensure the accuracy of the labels. Both annotators refer to the intent name, definition, and initial examples to understand the purpose of each intent before proceeding with the annotation. In case there is a disagreement, both annotators revisit the examples and discuss them to reach an agreement. Furthermore, to measure the reliability of the annotations, we calculate the Cohen kappa score based on the 200 examples labelled by both annotators. Our analysis yields a Cohen kappa score of 0.63, indicating a substantial agreement between the annotators. Notably, this Cohen kappa score is higher than the

one documented in prior work Abdellatif, Costa, et al. (2020).

Results: Table 5.4 presents the similarity scores between the initial and generated examples for each dataset (measured using the BLEU and ROUGE scores) and the percentages of augmented examples preserving the intent. From the table, we find that the generated examples in the Stack Overflow dataset are the most distinct (i.e., low similarity) compared to the initial ones, with BLEU-1 and ROUGE-L scores of 28.8% and 11.7%, respectively. Among all datasets, the augmented examples in the Repository dataset show the highest similarity to the initial ones, with a BLEU-1 score of 48.7% and ROUGE-L score of 21.7%. The augmented examples in the Ask Ubuntu and MSA datasets reveal moderate similarity with the initial examples, having BLEU-1 scores of 43.8% and 40.1%, respectively. These moderate to low similarity scores highlight ChatGPT's effectiveness in generating syntactically distinct examples, as opposed to replicating the initial human-written examples.

To gain deeper insights into the results, we manually examine the examples generated by ChatGPT. We observe that the scope of the intent considerably influences the diversity of the generated examples. For instance, the examples pertaining to the 'FixingCommitsForBugTicket' intent from the Repository dataset are narrowly focused, as they mainly inquire about fixing commits for specific bugs (e.g., "Tell me which commit(s) resolved the bug ticket PQR-654" and "Which commits addressed the issue STU-321?"). Conversely, the examples of the 'LookingForCodeSample' intent from the Stack Overflow dataset have a wider scope, inquiring about code snippets that implement various functionalities across different programming languages (e.g., "How to create a JS object from scratch using an HTML button?" and "How to manipulate a BLOB in Java?"). In other words, the intents in the Stack Overflow dataset are broader than those in the Repository dataset, leading ChatGPT to generate a more diverse range of examples compared to the initial dataset.

For preserving the intent in the generated examples, the results (except the MSA dataset) show that there is a high degree of intent preservation overall, as shown in Table 5.4. For example, 99% of the generated examples in the Ask Ubuntu dataset preserve the intent. On the other hand, 80% of the generated examples in the MSA dataset are aligned with their intents. Upon closer examination of the generated examples in the MSA dataset, we find that the generated examples that are misaligned with the intent often ask about the 'How' instead of the 'What'. For instance, in the

Table 5.4: Similarity and intent preservation of augmented examples. All scores are percentages.

	Repository	SOF	Ask Ubuntu	MSA
BLEU-1	48.9	28.5	43.8	40.1
BLEU-2	20.5	10.4	16.6	15.8
BLEU-3	9.0	3.9	6.5	6.4
BLEU-4	3.9	1.5	2.5	2.6
ROUGE-L	27.6	11.5	23.3	21.6
Intent preservation	96.7	93.1	99.1	79.8

'ServiceUsingInfo' intent (e.g., "What's the GroceryInventory's using overview?"), the generated example "How can I monitor the usage for the GroceryInventory service?" is misaligned with the intent.

Building on the insights from RQ1, where we found that ChatGPT enhances the NLU's performance, our current results demonstrate that ChatGPT is also capable of generating examples with distinct syntax (i.e., low similarity compared to the initial examples) while preserving the semantics (i.e., intents). This emphasizes the potential of using ChatGPT to augment SE chatbot datasets.

ChatGPT generates a range of examples that have syntactic diversity while possessing similar semantics across all datasets. Moreover, the intent's scope influences the similarity between the initial and augmented examples.

5.3 RQ3: What factors influence the performance of ChatGPT when generating examples?

Motivation: Previous studies show that large language models are sensitive to the input prompts Reynolds and McDonell (2021). In our evaluation, we design a prompt that includes four components; the 'Intent Name', 'Intent Definition', 'Intent Examples', and 'Task Boundaries', as discussed in Section 4.2. In this RQ, we set out to perform an ablation study to investigate the impact of each component in the prompt on the NLU's performance and the diversity of the generated examples. This analysis provides insights to chatbot practitioners on the significance of each component in the prompt when augmenting training datasets for SE chatbots using ChatGPT.

Table 5.5: The NLU's performance and example similarity when removing prompt components. All scores are percentages.

	Repository	SOF	Ask Ubuntu	MSA	Avg
			F1-score		
Full Prompt	83.8	47.0	91.2	87.3	77.3
w/o Intent Name	82.6 (-1.2)	44.1 (-2.9)*	90.6 (-0.6)	86.5 (-0.8)	75.9 (-1.4)
w/o Intent Definition	84.1 (+0.3)	43.7 (-3.3)*	89.7 (-1.5)*	85.5 (-1.8)	75.7 (-1.6)
w/o Intent Examples	83.4 (-0.4)	43.9 (-3.1)*	88.1 (-3.1)*	87.4 (+0.1)	75.7 (-1.6)
w/o Task Boundaries	83.4 (-0.4)	45.8 (-1.2)	91.4 (+0.2)	87.3 (0.0)	77.0 (-0.3)
			BLEU-3		
Full Prompt	9.0	3.9	6.5	6.4	6.4
w/o Intent Name	9.0 (0.0)	4.0 (+0.1)	6.3 (-0.2)	6.2 (-0.2)	6.4 (0.0)
w/o Intent Definition	9.4 (+0.4)*	4.4 (+0.5)*	6.1 (-0.4)*	7.0 (+0.6)*	6.7 (+0.3)
w/o Intent Examples	6.6 (-2.4)*	3.7 (-0.2)*	5.8 (-0.7)*	4.4 (-2.0)*	5.1 (-1.3)*
w/o Task Boundaries	9.7 (+0.7)*	4.0 (+0.1)	6.6 (+0.1)	7.0 (+0.6)*	6.8 (+0.4)

Approach: To study the impact of each prompt component on the NLU's performance and the similarity of the generated examples, we successively remove one component (e.g., 'Intent Name') from the full prompt shown in Figure. Using the modified prompt, we follow the same evaluation process described in RQ1 to measure the impact of the augmented examples on the NLU's performance. We then employ the BLEU and ROUGE metrics from RQ2 to calculate the similarity between the initial and the generated examples. We follow the same process for all datasets discussed in Section 4.1. We use the nonparametric unpaired Mann-Whitney U test to determine the statistical significance of the results when each component is removed, compared to results from the full prompt. While we evaluate all BLEU and ROUGE-L scores, we only showcase BLEU-3 in this RQ. Nonetheless, we include the results for all BLEU and ROUGE-L scores in the appendix A.

Results:

From Table 5.5, we find that the 'Intent Definition' and 'Intent Examples' components have a significant effect on the NLU's performance (-1.6% in F1-score on average). The impact of removing either of these components is more pronounced in the Stack Overflow and Ask Ubuntu datasets, where removing one of these components leads to a drop in performance of up to -3.3% in F1-score. These findings highlight the relative importance of the 'Intent Definition' and 'Intent Examples' components in generating effective examples that enhance the NLU's performance. One possible

reason for this observation is that these two components are the most informative, offering richer context to ChatGPT and guiding it toward generating more valuable examples. Unsurprisingly, excluding the 'Task Boundaries' and 'Intent Name' components has a negligible impact (except for removing the intent name in the Stack Overflow dataset) on the NLU's performance across datasets. Excluding the intent name leads to a drop in performance of -1.4% on average, while removing the task boundaries component leads to a drop of -0.3% in performance on average. This is likely because the 'Task Boundaries' and 'Intent Name' components offer relatively less context to ChatGPT in comparison to the 'Intent Definition' and 'Intent Examples' components.

From Table 5.5, we find that the 'Intent Definition' and 'Intent Examples' components have a significant effect on the NLU's performance, with an average decrease in F1-score by -1.6%. The impact of removing either of these components is more pronounced in the Stack Overflow and Ask Ubuntu datasets, where the removal of one of these components leads to a drop in performance of up to -3.3% in F1-score. These findings underscore the relative importance of the 'Intent Definition' and 'Intent Examples' components in generating effective examples that enhance the NLU's performance. One possible reason for this is that these two components are the most informative, providing richer context to ChatGPT and guiding it to generate more valuable examples. Unsurprisingly, excluding the 'Task Boundaries' and 'Intent Name' components has a negligible impact on the NLU's performance across datasets, except when removing the 'Intent Name' in the Stack Overflow dataset. Excluding the 'Intent Name' results in an average performance drop of -1.4%, while removing the 'Task Boundaries' component leads to an average drop of -0.3% in performance. This minor impact may be because the 'Task Boundaries' and 'Intent Name' components offer relatively less context to ChatGPT compared to the 'Intent Definition' and 'Intent Examples' components.

When examining the diversity of the generated examples, we find a significant reduction in similarity (in terms of BLEU scores) across all datasets upon the exclusion of 'Intent Examples' as depicted in Table 5.5. In other words, removing the 'Intent Examples' yields generated examples that are less similar to the initial examples. This phenomenon could be attributed to ChatGPT relying significantly on the intent examples when they are included in the prompt. Therefore, ChatGPT generates examples that closely resemble the initial examples, leading to a higher level of similarity between them. Furthermore, excluding the 'Intent Definition' component leads to an increase in

example similarity across all datasets except for Ask Ubuntu. This is likely because removing the 'Intent Definition' makes ChatGPT put more emphasis on the 'Intent Examples' component, leading to generated examples that are more syntactically similar to the 'Intent Examples'. Similar to the performance results, the removal of the 'Task Boundaries' and 'Intent Name' components from the prompt has a minor impact on the BLEU scores of the generated examples.

The 'Intent Definition' and 'Intent Examples' components play a crucial role in generating examples that closely resemble human-written ones and enhance the NLU's performance. Conversely, the exclusion of the 'Task Boundaries' and 'Intent Name' components has a slight impact on both the similarty of the generated examples and the NLU's performance.

Chapter 6

Discussion

In this section, we delve into the evaluation results to gain deeper insights into ChatGPT's temperature sensitivity, along with evaluating the performance of the NLU trained solely on ChatGPT-generated examples. Finally, we provide a set of actionable recommendations to chatbot practitioners and researchers to achieve better NLU's performance using ChatGPT.

6.1 Examining The Impact of Temperature on The NLU's Performance

Table 6.1: F1-Score and Intent Preservation with varying temperature settings. All scores are percentages.

Score	Temperature	0.0	0.5	1	1.5
F1-score	**Repository**	83.9	84.0	83.8	83.3
	SOF	47.3	47.5	47.0	45.1
	Ask Ubuntu	91.1	91.5	91.2	89.9
	MSA	86.9	86.6	87.3	87.1
	Avg	77.3	77.4	77.3	76.4
Intent Preservation	**Repository**	100.0	92.0	96.7	92.0
	SOF	88.0	92.0	93.1	96.0
	Ask Ubuntu	100.0	100.0	99.1	96.0
	MSA	84.0	72.0	79.8	72.0
	Avg	93.0	89.0	92.0	89.0

Prior work highlights that the temperature parameter is an important factor that impacts ChatGPT's response quality Ahmed et al. (2023). Ranging from 0.0 to 2.0, a lower temperature produces more deterministic output, while higher temperatures typically increase the randomness in the response. Selecting the optimal temperature can benefit chatbot developers, as it may result in generating higher quality training examples that improve the NLU's performance. Therefore, in this section, we explore the impact of using different temperature settings on the NLU's performance. To achieve this, we use the same evaluation process described in Section 5.1, while varying the temperature to the values [0.0, 0.5, 1.0, 1.5, and 2.0].

Table 6.1 presents the NLU's performance, measured using the weighted average F1-score, when using different temperature values for ChatGPT. The results show that the NLU's performance remains constant overall when different temperatures are used. In other words, there is no significant impact of using different temperature values on the NLU's performance. To better understand the impact of the temperature on the intent preservation of the generated examples, we manually examined 400 examples (25 examples for each dataset, in each temperature). The results show that there is no significant change in intent preservation per dataset across the different temperatures, as shown in Table 6.1.

Interestingly, we observed that ChatGPT produced outputs that were excessively random and challenging to parse when using a temperature value of 2.0. For example, ChatGPT often generated full paragraphs of text instead of providing a list of training examples. Training the chatbot on paragraphs might cause it to learn to mimic lengthy and formal language, potentially missing the casual and short messages often anticipated in conversational interactions. Therefore, we excluded the results obtained using a temperature value of 2.0 from our analysis.

Based on our findings, we recommend that chatbot practitioners opt for temperatures within the range of [0 - 1.5] when employing ChatGPT to augment their chatbot dataset and avoid excessively high temperature values (e.g., 2.0). Nevertheless, we advise practitioners to explore the optimal temperature setting based on their specific context.

6.2 Training the NLU Using the Generated Examples Only

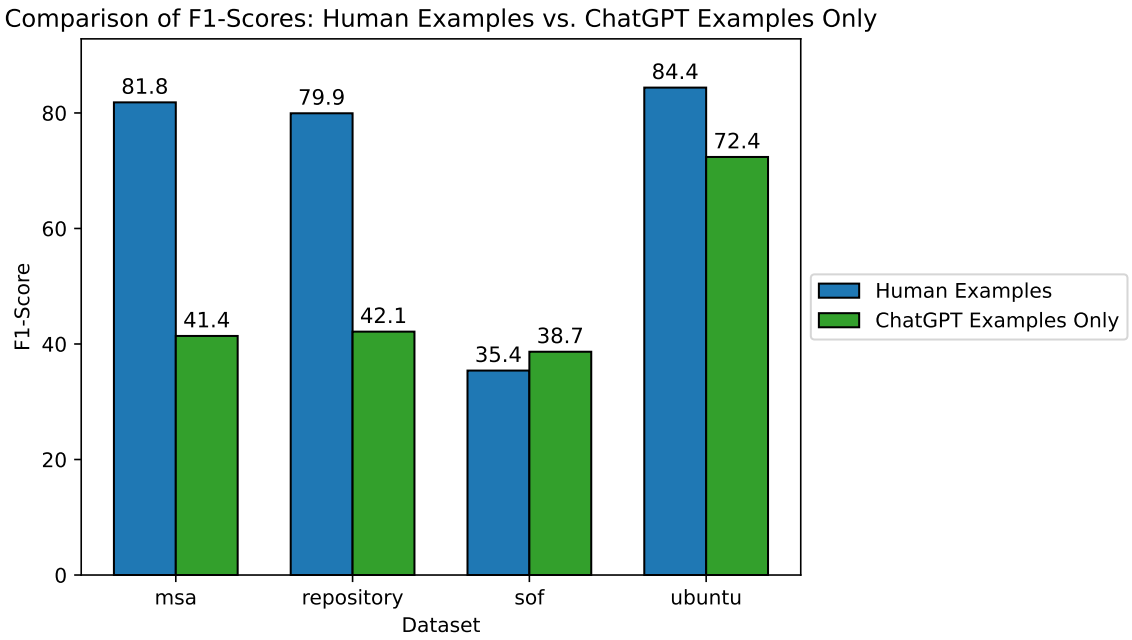

Figure 6.1: The NLU's performance (F1-score) when using generated examples only compared to human-written examples.

Augmenting the initial dataset with ChatGPT's generated examples improves the NLU's performance, as discussed in RQ1. However, it is critical to understand the impact of solely relying on generated examples (without augmenting them to the initial dataset). Therefore, we conduct an analysis mirrors a real-world scenario where chatbot practitioners lack any initial examples, requiring them to create a training dataset from scratch. In this analysis, we adopt a similar methodology described in RQ1, with the distinction that we removed 'Intent Examples' from the prompt. To put the results into perspective, we evaluate the NLU's performance when trained on three initial (i.e., human-written) examples per intent, which serves as the baseline in this analysis. Additionally, we randomly select three generated examples per intent from ChatGPT's output to align with the number of examples in the baseline, ensuring a fair comparison

Figure 6.1 presents the NLU's performance when using ChatGPT's generated examples only, compared to the baseline. Overall, we find the NLU's performance trained solely using ChatGPT's generated examples decrease compared to the baseline. For example, in the Repository dataset, the

NLU's performance drops by about half when using ChatGPT' generated examples. This might be attributed to the specificity inherent to these datasets, which might be difficult for ChatGPT to capture without any examples in the prompt. For instance, in the intent 'GetDevelopersWhoHas-ExperianceToFixBug' from the Repository dataset, the initial examples (e.g., "Who is the top bug fixing developer in relation to SetSchemaMetadata.java") possess specific context, focusing on individual file names. On the other hand, ChatGPT's generated examples, such as "Find me developers who are skilled in fixing bugs related to SQL files" take on a broader scope, focusing on generic programming languages and not files. In contrast, the Ask Ubuntu dataset experiences a moderate drop in performance from 84.4% (baseline) to 72.4%, while the Stack Overflow dataset shows a slight performance increase, moving from 35.4% (baseline) to 38.7%. The relatively higher performance shown by these two datasets when using the generated examples only to train the NLU can be attributed to the broad scope of intents in these datasets, which cover more generic questions related to software development and using the Ubuntu operating system. This indicates ChatGPT's relative familiarity with programming and Ubuntu-related examples, which likely contributes to the model's ability to craft questions related to these topics.

Overall, our findings suggest that relying exclusively on ChatGPT's generated examples tends to yield lower performance compared to using human-written examples, though in some cases, the performance gap is less pronounced. On the other hand, augmenting human-written examples with those generated by ChatGPT generated leads to more optimal performance, as demonstrated in RQ1 (Section 5.1).

6.3 Implications

In the following, we discuss the implications of our results for practitioners and researchers.

6.3.1 Implications For Practitioners

Our results show that including the 'Intent Examples' and 'Intent Definition' in the prompt leads to further improvement in the NLU's performance after augmentation, as shown in RQ3. Although

ChatGPT is trained on a massive amount of data, our results emphasize the importance of providing ChatGPT with the relevant context in the prompt. Contextual details become more critical when working on niche domains, primarily because ChatGPT tends to generalize when context is not provided in the prompt. Practitioners should provide ChatGPT with the specific context of their chatbot in the prompt, as it not only leads to improvement in the NLU's performance but also results in a more relevant output.

Our findings from RQ2 reveal that the majority (92.2% on average) of ChatGPT-generated examples successfully retain their intent. Nevertheless, a small fraction of these generated examples do not align with the intent. Chatbot practitioners need to manually review the generated examples to ensure they preserve their intent. Moreover, by reviewing ChatGPT's generated examples, practitioners can be inspired to craft additional examples that hadn't been previously considered. This iterative process, where ChatGPT's output feeds into human ideation, could lead to a richer dataset.

Our results can help chatbot practitioners select the appropriate temperature setting for ChatGPT. Setting the temperature value too low results in more deterministic outcomes OpenAI (2023c), which often leads to nearly identical responses for repeated prompts. This makes it challenging for practitioners to obtain diverse sets of examples. On the other hand, setting the temperature value too high can cause ChatGPT to produce responses that are both lengthy and random, as mentioned in section 6.1. Thus, we recommend that practitioners aim for a moderate temperature setting (e.g., 0.5, 1.0, and 1.5) to balance the variety and specificity of the generated examples. One possible approach is to begin with a lower temperature, given its tendency to generate less random responses. Then, as ChatGPT's generated examples begin to duplicate, to gradually increase the temperature. The greater range of outputs at higher temperatures may not only help in reducing repetition but also inspire practitioners to craft novel examples. Nonetheless, we encourage practitioners to investigate the temperature values that yield the best outcomes for their specific application.

6.3.2 Implications For Researchers

Our findings show a consistent pattern in ChatGPT's output, producing an average of ten examples, although there is no explicit number indicated in the prompt. These results open new research directions related to the number of generated examples and their quality. For example, researchers

can assess how specifying a desired number of examples in the prompt impacts both the quality of examples and NLU's performance after augmentation. Furthermore, we plan (and encourage researchers) to study the relationship between the prompt's components (e.g., 'Intent Definition' and 'Intent Examples') and the number of generated examples. Identifying the optimal number of examples to include in the prompt is another avenue for future research. It's crucial to strike a balance between providing sufficient context (i.e., enough examples) and overloading the model with information, which might lead to less relevant responses. Thus, there is a need for approaches to select which examples to include in the prompt. One approach could involve choosing examples that are maximally different in their syntax, thereby encouraging the model to generate diverse responses. This diversity might result in broader coverage of possible phrasings or structures, which in turn could lead to more robust training datasets for chatbot development.

Although ChatGPT-generated examples improve the NLU's performance, we observed that not all generated examples align with the intent, making manual review necessary. This indicates a need for post-processing techniques to vet and refine ChatGPT's output. Such techniques could automatically filter out generated examples that are misaligned with the intent, ensuring that only relevant examples are retained. Moreover, they can filter out similar examples to ensure that the remaining output is diverse. By implementing these techniques, developers can save the time and effort otherwise spent on inspecting the generated examples.

Chapter 7

Threats to Validity

In this section, we discuss the threats to internal, construct, and external validity of this thesis.

7.1 Internal Validity

Internal validity concerns factors that could have influenced our results. We manually label a sample of ChatGPT's generated examples to assess intent preservation. Manual labelling could introduce bias due to its subjectivity. To mitigate this threat, we had multiple annotators label a sample of the example and measured the interrater-agreement using the Cohen-Kappa test. We found substantial agreement (Kappa = 0.63) among the annotators. Another threat to internal validity is that ChatGPT is based on a transformer-based architecture, which introduces randomness in its responses. Thus, it might bias the results and conclusions in this thesis. To alleviate this threat, we repeat our evaluation 50 times and report the average of the results to minimize the impact of randomness.

7.2 Construct Validity

Construct validity considers the relationship between theory and observation, in case the measured variables do not measure the actual factors. We followed OpenAI's guidelines and best practices to construct the prompt to generate new training examples OpenAI (2023b). However, using

different templates might yield different output. We purposely decided to resort to OpenAI's documentation to evaluate ChatGPT's performance of a user that follows the online guidelines. Furthermore, we plan to explore the impact of using different prompts on ChatGPT's output in future research.

7.3 External Validity

Threats to external validity concern the generalizability of our findings. To assess the influence of ChatGPT-generated examples on the NLU's performance, we conducted a case study utilizing the Rasa platform. However, our findings may not necessarily apply to other NLUs (e.g., IBM Watson). Rasa is a widely-adopted open-source NLU tool, and it has consistently exhibited performance comparable to other NLUs Abdellatif et al. (2021). Furthermore, Rasa is frequently employed by practitioners in the development of software engineering chatbots Lin et al. (2020). One additional benefit of selecting Rasa is that it makes our work reproducible, as Rasa's implementation stays consistent. In our evaluation, we used four SE datasets; however, our results might not generalize to other datasets. Nonetheless, these datasets represent a variety of tasks from the SE domain, such as questions related to a project's repository or development-related questions. Moreover, these datasets have been used in similar studies to evaluate the NLU's performance Abdellatif et al. (2021); Braun et al. (2017).

Chapter 8

Conclusion, Contributions, and Future Work

This chapter presents the conclusion of the thesis, and discusses areas for future work.

8.1 Conclusion

Chatbots are gaining momentum within the SE domain, successfully performing various SE tasks, such as code refactoring and answering technical questions, which results in resource savings. However, prior work shows a scarcity of training data, which hinders the chatbots' ability to comprehend users' questions. Recently, ChatGPT has demonstrated its effectiveness in numerous SE tasks. In this thesis, we explore ChatGPT's potential in augmenting four SE datasets. More specifically, we design a prompt comprising four different components (e.g., 'Intent Definition' and 'Intent Examples') to ask ChatGPT to generate new training examples. Then, we evaluate the impact of using ChatGPT-generated examples on the NLU's performance. We find that ChatGPT generates training examples that improve the NLU's performance, ranging from 3.9% to 11.6%. Subsequently, we examined the similarity between ChatGPT's examples and human-written ones and assessed whether ChatGPT's examples preserve the intent. Our findings reveal that while ChatGPT's generated examples have syntactical differences from those written by humans, they largely

(92.2% on average) preserve the intent. Furthermore, we observe that varying the temperature settings in ChatGPT has minimal impact on the NLU's performance. However, higher temperatures can produce excessively random outputs. Our results show that ChatGPT can be a valuable tool for chatbot practitioners looking to augment their datasets, especially during the initial stages of chatbot development when there are limited training examples available.

8.2 Contributions

This thesis makes the following contributions to practitioners and the research community:

- To the best of our knowledge, this is the first work that investigates the use of ChatGPT for augmenting SE chatbot training datasets.
- We conduct an empirical evaluation across four distinct SE datasets, offering insights into ChatGPT's generative capabilities in diverse contexts for the SE domain.
- We provide actionable recommendations to SE chatbot practitioners aiming to harness ChatGPT in their workflows.
- We make our replication package publicly available to enable replication and encourage further studies by the chatbot community Zenodo (2023).

8.3 Future Work

Our work opens the door for future work in various areas:

8.3.1 Optimize the Augmentation Process with Post-processing Techniques

Using ChatGPT to augment SE chatbot datasets has the potential to streamline the chatbot development process; however, manual review of the generated examples is still necessary. Future work could focus on refining ChatGPT's output by developing post-processing techniques to filter out any undesirable examples, especially those with misaligned intents. This approach aims to enhance the efficiency and accuracy of the augmentation process, potentially leading to a significant

reduction in the time and effort required by developers.

8.3.2 Explore the Effectiveness of Advanced Models like GPT-4

Our investigation into ChatGPT's performance focused on the GPT-3.5-turbo model. However, with the introduction of more advanced models like GPT-4 by OpenAI, there is a potential that using these advanced models could lead to a more effective augmentation process. More specifically, these models may have a better understanding of the provided context and may generate more relevant output. In turn, this might lead to better NLU performance after augmentation and a higher quality of generated examples. Considering that more advanced models tend to be costlier, future studies could compare the effectiveness of different models against their associated costs.

8.3.3 Evaluate the Impact of Different Prompt Templates

While this thesis has explored the augmentation of SE datasets using a specific prompt template, the impact of varying the prompt template on NLU's performance and diversity of the generated examples remains an open question. Future research could systematically analyze different prompt phrasings and their impact on NLU performance. Additionally, investigating the optimal number of human-written examples to include in the prompt, which may yield better results, is another promising avenue. Such studies would offer valuable insights into how prompt design influences the effectiveness of ChatGPT in the augmentation process.

8.3.4 Extend the Assessment Across Diverse SE Datasets

Our findings in this thesis have demonstrated the potential of ChatGPT in augmenting SE chatbot datasets. This opens up opportunities for future research to explore the use of ChatGPT in augmenting datasets for other SE applications, such as requirement analysis. Priority could be given to areas with text-based datasets and where data scarcity is a significant challenge.

Appendix A

Appendix-A

In Chapter 5.3, we presented the results of removing individual components from the prompt with regard to the F1-score and BLEU-3 metrics. In this appendix, we add detailed results from all metrics, including BLEU-1 through BLEU-4 and ROUGE-L.

Table A.1: The NLU's performance and example similarity when removing prompt components. All scores are percentages.

Scenario	**Repository**	**SOF**	**Ask Ubuntu**	**MSA**	**Avg**
F1-score					
Full Prompt	83.8	47.0	91.2	87.3	77.3
w/o Intent Name	82.6	44.1	90.6	86.5	75.9
w/o Intent Definition	84.1	43.7	89.7	85.5	75.7
w/o Intent Examples	83.4	43.9	88.1	87.4	75.7
w/o Task Boundaries	83.4	45.8	91.4	87.3	77.0
BLEU-1					
Full Prompt	48.9	28.5	43.8	40.1	40.3
w/o Intent Name	49.0	28.6	43.3	39.9	40.2
w/o Intent Definition	50.2	30.4	42.0	41.6	41.1
w/o Intent Examples	43.0	27.8	41.5	31.5	36.0
w/o Task Boundaries	50.1	28.5	44.1	42.3	41.3
BLEU-2					
Full Prompt	20.5	10.4	16.6	15.8	15.8
w/o Intent Name	20.5	10.5	16.3	15.5	15.7
w/o Intent Definition	21.3	11.4	15.7	16.8	16.3
w/o Intent Examples	16.6	10.0	15.3	11.6	13.4
w/o Task Boundaries	21.4	10.5	16.8	16.9	16.4

Scenario	Repository	SOF	Ask Ubuntu	MSA	Avg
BLEU-3					
Full Prompt	9.0	3.9	6.5	6.4	6.4
w/o Intent Name	9.0	4.0	6.3	6.2	6.4
w/o Intent Definition	9.4	4.4	6.1	7.0	6.7
w/o Intent Examples	6.6	3.7	5.8	4.4	5.1
w/o Task Boundaries	9.7	4.0	6.6	7.0	6.8
BLEU-4					
Full Prompt	3.9	1.5	2.5	2.6	2.6
w/o Intent Name	4.0	1.5	2.4	2.4	2.6
w/o Intent Definition	4.2	1.7	2.3	2.9	2.8
w/o Intent Examples	2.6	1.4	2.2	1.7	2.0
w/o Task Boundaries	4.4	1.5	2.6	2.9	2.9
ROUGE-L					
Full Prompt	27.6	11.5	23.3	21.6	21.0
w/o Intent Name	27.5	11.6	22.4	21.3	20.7
w/o Intent Definition	29.1	13.3	21.4	23.3	21.8
w/o Intent Examples	20.5	9.6	19.4	13.9	15.9
w/o Task Boundaries	29.1	11.5	23.6	23.6	22.0

References

Abdellatif, A., Badran, K., Costa, D., & Shihab, E. (2021). A comparison of natural language understanding platforms for chatbots in software engineering. *IEEE Transactions on Software Engineering (TSE)*, 1-1. doi: 10.1109/TSE.2021.3078384

Abdellatif, A., Badran, K., & Shihab, E. (2020). Msrbot: Using bots to answer questions from software repositories. *Empirical Software Engineering (EMSE)*, *25*, 1834-1863.

Abdellatif, A., Costa, D. E., Badran, K., Abdelkareem, R., & Shihab, E. (2020). Challenges in chatbot development: A study of stack overflow posts. In *Proceedings of the 17th international conference on mining software repositories (msr'20)* (p. To Appear).

Ahmed, T., Ghosh, S., Bansal, C., Zimmermann, T., Zhang, X., & Rajmohan, S. (2023). Recommending root-cause and mitigation steps for cloud incidents using large language models. In *Proceedings of the 45th international conference on software engineering* (p. 1737–1749). IEEE Press. Retrieved from https://doi.org/10.1109/ICSE48619.2023.00149 doi: 10.1109/ICSE48619.2023.00149

Amin-Nejad, A., Ive, J., & Velupillai, S. (2020, May). Exploring transformer text generation for medical dataset augmentation. In *Proceedings of the 12th language resources and evaluation conference* (pp. 4699–4708). Marseille, France: European Language Resources Association.

Braun, D., Hernandez Mendez, A., Matthes, F., & Langen, M. (2017, August). Evaluating natural language understanding services for conversational question answering systems. In *Proceedings of the 18th annual SIGdial meeting on discourse and dialogue* (pp. 174–185). Saarbrücken, Germany: Association for Computational Linguistics. doi: 10.18653/v1/W17-5522

Cao, J., Li, M., Wen, M., & chi Cheung, S. (2023). *A study on prompt design, advantages and limitations of chatgpt for deep learning program repair.*

Dialogflow, G. (n.d.). *General agent design best practices.* Retrieved from https://cloud.google.com/dialogflow/es/docs/agents-design

Dominic, J., Houser, J., Steinmacher, I., Ritter, C., & Rodeghero, P. (2020a). Conversational bot for newcomers onboarding to open source projects. In *Proceedings of the ieee/acm 42nd international conference on software engineering workshops* (p. 46–50). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3387940.3391534

Dominic, J., Houser, J., Steinmacher, I., Ritter, C., & Rodeghero, P. (2020b). Conversational bot for newcomers onboarding to open source projects. In *Proceedings of the ieee/acm 42nd international conference on software engineering workshops* (p. 46–50). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3387940.3391534 doi: 10.1145/3387940.3391534

Dong, Y., Jiang, X., Jin, Z., & Li, G. (2023). *Self-collaboration code generation via chatgpt.*

Ebert, C., & Louridas, P. (2023). Generative ai for software practitioners. *IEEE Software*, *40*(4), 30-38. doi: 10.1109/MS.2023.3265877

Feng, S. Y., Gangal, V., Kang, D., Mitamura, T., & Hovy, E. (2020, November). GenAug: Data augmentation for finetuning text generators. In *Proceedings of deep learning inside out (deelio): The first workshop on knowledge extraction and integration for deep learning architectures* (pp. 29–42). Online: Association for Computational Linguistics. doi: 10.18653/v1/2020.deelio-1.4

Feng, Y., Vanam, S., Cherukupally, M., Zheng, W., Qiu, M., & Chen, H. (2023). Investigating code generation performance of chatgpt with crowdsourcing social data. In *2023 ieee 47th annual computers, software, and applications conference (compsac)* (p. 876-885). doi: 10.1109/COMPSAC57700.2023.00117

Imran, M. M., Jain, Y., Chatterjee, P., & Damevski, K. (2023). Data augmentation for improving emotion recognition in software engineering communication. In *Proceedings of the 37th ieee/acm international conference on automated software engineering.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/

3551349.3556925 doi: 10.1145/3551349.3556925

Joshi, H., Cambronero, J. P., Gulwani, S., Le, V., Radicek, I., & Verbruggen, G. (2022). Repair is nearly generation: Multilingual program repair with llms. In *Aaai conference on artificial intelligence*. Retrieved from https://api.semanticscholar.org/CorpusID:251765058

Li, T.-O., Zong, W., Wang, Y., Tian, H., Wang, Y., Cheung, S.-C., & Kramer, J. (2023). Nuances are the key: Unlocking chatgpt to find failure-inducing tests with differential prompting. In *Proceedings of the 38th ieee/acm international conference on automated software engineering*.

Lin, C.-T., Ma, S.-P., & Huang, Y.-W. (2020). Msabot: A chatbot framework for assisting in the development and operation of microservice-based systems. In *Proceedings of the ieee/acm 42nd international conference on software engineering workshops* (p. 36–40). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3387940.3391501

Maddigan, P., & Susnjak, T. (2023). Chat2vis: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. *IEEE Access*, *11*, 45181-45193. doi: 10.1109/ACCESS.2023.3274199

Marivate, V., & Sefara, T. (2020). Improving short text classification through global augmentation methods. In A. Holzinger, P. Kieseberg, A. M. Tjoa, & E. Weippl (Eds.), *Machine learning and knowledge extraction* (pp. 385–399). Cham: Springer International Publishing.

Mastropaolo, A., Scalabrino, S., Cooper, N., Nader Palacio, D., Poshyvanyk, D., Oliveto, R., & Bavota, G. (2021). Studying the usage of text-to-text transfer transformer to support code-related tasks. In *2021 ieee/acm 43rd international conference on software engineering (icse)* (p. 336-347). doi: 10.1109/ICSE43902.2021.00041

OpenAI. (n.d.). *Openai documentation gpt-3.5 model.* Retrieved from https://platform.openai.com/docs/models/gpt-3-5

OpenAI. (2023a). *Gpt-4 technical report.* Retrieved from https://cdn.openai.com/papers/gpt-4.pdf ([Online])

OpenAI. (2023b, 07). Gpt best practices — openai docs.. https://platform.openai.com/docs/guides/gpt-best-practices. ((Accessed on 14/07/2023))

OpenAI. (2023c). *How should i set the temperature parameter?* Retrieved from https://platform.openai.com/docs/guides/gpt/how-should-i-set-the-temperature-parameter

Ozkaya, I. (2023). Application of large language models to software engineering tasks: Opportunities, risks, and implications. *IEEE Software*, *40*(3), 4-8. doi: 10.1109/MS.2023.3248401

Reynolds, L., & McDonell, K. (2021). Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended abstracts of the 2021 chi conference on human factors in computing systems.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3411763.3451760 doi: 10.1145/3411763.3451760

Rizos, G., Hemker, K., & Schuller, B. (2019). Augment to prevent: Short-text data augmentation in deep learning for hate-speech classification. In *Proceedings of the 28th acm international conference on information and knowledge management* (p. 991–1000). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3357384.3358040

Saini, R., Mussbacher, G., Guo, J. L. C., & Kienzle, J. (2020). Domobot: A bot for automated and interactive domain modelling. In *Proceedings of the 23rd acm/ieee international conference on model driven engineering languages and systems: Companion proceedings.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3417990.3421385 doi: 10.1145/3417990.3421385

Serrano Alves, L. P., Wiese, I. S., Chaves, A. P., & Steinmacher, I. (2022). How to find my task? chatbot to assist newcomers in choosing tasks in oss projects. In A. Følstad et al. (Eds.), *Chatbot research and design* (pp. 90–107). Cham: Springer International Publishing.

Sharifirad, S., Jafarpour, B., & Matwin, S. (2018, October). Boosting text classification performance on sexist tweets by text augmentation and text generation using a combination of knowledge graphs. In *Proceedings of the 2nd workshop on abusive language online (ALW2)* (pp. 107–114). Brussels, Belgium: Association for Computational Linguistics. doi: 10.18653/v1/W18-5114

Sun, W., Fang, C., You, Y., Miao, Y., Liu, Y., Li, Y., ... Chen, Z. (2023). *Automatic code summarization via chatgpt: How far are we?*

Tihanyi, N., Bisztray, T., Jain, R., Ferrag, M. A., Cordeiro, L. C., & Mavroeidis, V. (2023). *The formai dataset: Generative ai in software security through the lens of formal verification.*

Wei, J., & Zou, K. (2019, November). EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* (pp. 6382–6388). Hong Kong, China: Association for Computational Linguistics. doi: 10.18653/v1/D19-1670

Wolfinger, R., Fotrousi, F., & Maalej, W. (2022). A chatbot for the elicitation of contextual information from user feedback. In *2022 ieee 30th international requirements engineering conference (re)* (p. 272-273). doi: 10.1109/RE54965.2022.00040

Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., & Tang, Y. (2023). A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, *10*(5), 1122-1136. doi: 10.1109/JAS.2023.123618

Wyrich, M., & Bogner, J. (2019). Towards an autonomous bot for automatic source code refactoring. In *2019 ieee/acm 1st international workshop on bots in software engineering (botse)* (p. 24-28). doi: 10.1109/BotSE.2019.00015

Xu, B., Xing, Z., Xia, X., & Lo, D. (2017). Answerbot: Automated generation of answer summary to developersź technical questions. In *Proceedings of the 32nd ieee/acm international conference on automated software engineering* (p. 706–716). IEEE Press.

Ye, D., Xing, Z., Foo, C. Y., Ang, Z. Q., Li, J., & Kapre, N. (2016). Software-specific named entity recognition in software engineering social content. In *2016 ieee 23rd international conference on software analysis, evolution, and reengineering (saner)* (Vol. 1, p. 90-101). doi: 10.1109/SANER.2016.10

Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. (2023). *Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt.*

Yuan, Z., Lou, Y., Liu, M., Ding, S., Wang, K., Chen, Y., & Peng, X. (2023). *No more manual tests? evaluating and improving chatgpt for unit test generation.*

Zenodo. (2023, November). *Using ChatGPT to Augment Software Engineering Chatbots Dataset.*

Author. Retrieved from https://doi.org/10.5281/zenodo.10153042 doi: 10.5281/zenodo.10153042

Zhang, J., Chen, Y., Niu, N., Wang, Y., & Liu, C. (2023). *Empirical evaluation of chatgpt on requirements information retrieval under zero-shot setting.*

Zhang, N., Huang, Q., Xia, X., Zou, Y., Lo, D., & Xing, Z. (2022). Chatbot4qr: Interactive query refinement for technical question retrieval. *IEEE Transactions on Software Engineering*, *48*(4), 1185-1211. doi: 10.1109/TSE.2020.3016006