Goal Model Generation from Large-Scale User Reviews: A Method for Requirements Extraction and Visualization

Submitted to

Graduate School of Information Science and Technology

Osaka University

January 2025

Shuaicai REN

Abstract

With the rapid advancement of technology, the scale of application markets is expanding, leading to an increasing diversity of applications (apps). In this highly competitive environment, developers must accurately grasp user requirements and timeously implement unique features to gain an advantage over competitors. Consequently, user reviews submitted on app platforms are gradually becoming a crucial resource to help developers understand user requirements and optimize their products.

User reviews include suggestions for improving existing features and requirements for new ones while also revealing the actual usage scenarios, reasons for use, and the users' requirements, thereby, guiding developers toward new product iterations. However, as the number of users increases, the volume of reviews grows significantly, with only a small portion containing information about user requirements. This makes it exceedingly difficult for developers to analyze and filter requirements manually from so many reviews. In this context, extracting useful information efficiently is a significant challenge.

Several studies, each with its own limitations, have proposed automating the extraction of relevant data from user reviews. Some failed to visualize the relationships between the extracted requirements, hindering developers from identifying priorities and conflicting requirements. Others cannot handle large volumes of reviews, rendering them impractical for development. To overcome these problems, this dissertation proposes a method to analyze large-scale reviews and visualize their requirements. It extracts requirements from reviews and visualizes them by generating goal models. A goal model is a type of requirements model that treats requirements as goals to be achieved, aiding developers in understanding the hierarchical relationships and dependencies between these goals. Additionally, by utilizing unsupervised learning and clustering algorithms, the proposed method extracts requirements from a multitude of reviews, making it applicable for development. Furthermore, to ensure the generated goal models are both accurate and easy to understand, large language models (LLMs) are utilized.

The goal model generation method proposed in this dissertation consists of three components: the Latent Dirichlet Allocation (LDA) topic model, the distance-based clustering method, and the goal model generation method based on LLMs. This method adopts a top-down approach, starting with a root goal and progressively generating a complete goal model layer by layer. The root goal is associated with all user reviews, which are then refined step by step to generate more specific sub-goals. In refining the parent goals and generating sub-goals, the method selects the refinement method based on the number of reviews associated with the parent goal. Specifically, the LDA topic model, distance-based methods, and LLMs each excel at handling different scales of review sets. Therefore, when the number of reviews for a parent goal is large, the LDA topic model or clustering methods is used to identify potential sub-goals. Conversely, when the number of reviews is small or requires more nuanced semantic understanding, LLM-based methods can be used to provide more precise goal generation.

This study undertook four experiments. The first demonstrated changes in the LDA topic model and the distance-based clustering method's accuracy in goal generation as the volume of reviews increased. The second experiment confirmed that the goal models generated by the proposed method had accuracy superior to the existing method. The third experiment demonstrated that the proposed method accurately extracts requirements from user reviews. The fourth experiment demonstrated the practicality of the goal model generated by the proposed method.

In summary, this dissertation presents a goal model generation method that automatically extracts requirements and generates goal models from user reviews by combining the LDA topic model, distance-based clustering method, and LLMs. This method will help developers better understand user requirements, thereby optimizing the app development and iteration process.

List of Major Publications

(1) Shuaicai, Ren, Hiroyuki Nakagawa, and Tatsuhiro Tsuchiya, An Automated Goal Labeling Method Based on User Reviews, In *the 32nd International Conference on Software Engineering & Knowledge Engineering (SEKE 2020)*, pp. 141–146, July 2020.

(2) Shuaicai, Ren, Hiroyuki Nakagawa, and Tatsuhiro Tsuchiya, Hierarchical User Review Clustering Based on Multiple Sub-goal Generation, In *14th International Joint Conference on Knowledge-Based Software Engineering (JCKBSE 2022)*, pp. 207–219, August 2022.

(3) Shuaicai, Ren, Hiroyuki Nakagawa, and Tatsuhiro Tsuchiya, Goal Model Structuring Based on Semantic Correlation of User Reviews, In *Intelligent Decision Technologies*, vol. 16, no. 4, pp. 737–748, December 2022.

(4) Shuaicai, Ren, Hiroyuki Nakagawa, and Tatsuhiro Tsuchiya, Exploring the Potential of LLM for Review-driven Goal Model Generation, In *IEICE Technical Report*, pp. 109–116, September 2023.

(5) Shuaicai, Ren, Hiroyuki Nakagawa, and Tatsuhiro Tsuchiya, Harnessing LLM Conversations for Goal Model Generation from User Reviews, In *16th International Conference on Agents and Artificial Intelligence (ICAART 2024)*, pp. 385–392, February 2024.

(6) Shuaicai, Ren, Hiroyuki Nakagawa, and Tatsuhiro Tsuchiya, Combining Prompts with Examples to Enhance LLM-Based Requirement Elicitation, In *The 48th IEEE International Conference on Computers, Software, and Applications (COMPSAC 2024)*, pp. 1376–1381, July 2024.

ACKNOWLEDGMENTS

Throughout this research journey, I have been privileged to receive support from many individuals whose contributions have been invaluable. First and foremost, I express my deepest gratitude to my advisor, Professor Hiroyuki Nakagawa, whose guidance, encouragement, and insights have been a constant source of inspiration and support. Without his patience and expertise, this work would not have been possible.

I am also immensely grateful to Professor Tatsuhiro Tsuchiya from my research lab, whose mentorship and support have guided me throughout this journey. His expertise and encouragement have greatly contributed to my growth as a researcher.

I am profoundly grateful to the members of my dissertation committee, Professor Shinji Kusumoto and Associate Professor Jun Shiomi, whose constructive feedback and valuable comments helped shape this dissertation's final form.

Special thanks go to my colleagues in the Tsuchiya Laboratory, especially Assistant Professor Junjun Zheng and my senior Hao Jin, for their valuable discussions that greatly enriched this research.

Last, I am eternally grateful to my family, whose unwavering support and understanding helped me navigate challenging times. Their belief in me has been a constant source of strength and motivation.

Thank you all for your invaluable support and encouragement, which made this accomplishment possible.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background

In contemporary society, mobile apps play an essential role in our everyday activities. Platforms like the App Store and Google Play are more than just places to download apps; they are spaces where users and developers interact. These platforms invite users to submit reviews, their requirements for new features, and report bugs, which are crucial for developers to gain insights and improve their products [47] [40] [55] [37]. The reviews help developers spot trends and common issues, effectively guiding their updates. By analyzing these invaluable review data, developers can better understand user requirements, offer insights into user preferences, desirable features, and areas for improvement [48] [26] [63]. In the requirements-analysis domain, the goal model stands out as one of the most frequently employed models because it can be employed to analyze the requirements extracted from user reviews.

Compared to directly analyzing user reviews, creating goal models helps ensure that user reviews match the app's goals. Connecting reviews with goals can facilitate understanding which goals users care about and whether the new features they want may conflict with current goals. This helps developers identify which user requirements are most urgent and helps them make better updates.

While there are numerous advantages to employing a goal model for analyzing user reviews, the manual construction of a goal model presents a significant challenge. This challenge mainly arises from the vast number of reviews, with only a small proportion containing requirements or bug reports [36] [48] [10]. Consequently, the process of manually reading and summarizing reviews becomes a labor-intensive and time-consuming task. The rapid growth of apps and the quick spread of app stores have led to a significant increase in user reviews. For popular apps, there might be thousands or even millions of reviews. Consequently, it becomes unfeasible for developers to check each review manually to extract informative reviews. Furthermore, a significant portion of user reviews may contain general feedback, opinions, or discussions unrelated to specific requirements. This makes it harder to extract requirements from the reviews.

1.2 Requirements Analysis

Requirements analysis is a crucial phase in software development where the requirements, expectations, and constraints of users and stakeholders are identified and documented. This process ensures that the software product being developed aligns with the business objectives and meets the users' requirements. It involves understanding the functional and non-functional requirements that the system must fulfill, often serving as the foundation for the design and development phases.

The primary benefit of requirements analysis is that it helps prevent costly mistakes later in the development process. By thoroughly analyzing requirements early on, teams can identify potential issues, ambiguities, and inconsistencies, thereby reducing the risk of project delays, budget overruns, and unsatisfactory outcomes. Furthermore, it facilitates better communication between stakeholders and developers, ensuring that everyone has a clear understanding of what the project aims to achieve.

Several models are commonly used in requirements analysis to capture and visualize the requirements effectively: Goal Models are used to capture high-level goals and break them down into sub-goals. They help ensure that all system functionalities are aligned with business objectives. Use Case Diagrams illustrate the interactions between users and the system, highlighting the system's functionalities. Entity-Relationship Diagrams (ERDs) are used to model the data and relationships within the system, essential for database design. Data Flow Diagrams (DFDs) represent the flow of information within the system, showing how data moves between processes and data stores. State Diagrams depict the different states of the system and the transitions between these states, which is useful for understanding the system's behavior.

Requirements analysis relies on particular materials and tools to gather, document, and analyze requirements: Requirements Documentation includes formal documents like Software Requirements Specifications (SRS) that detail the system's functional and non-functional requirements. App Descriptions, for mobile or web applications, from app platforms (e.g., The App Store, Google Play) can provide valuable insights into user requirements and expectations. Specialized Languages, like the Unified Modeling Language (UML), are often used to create visual models that help understand and communicate requirements.

By systematically analyzing requirements using these models and materials, teams can ensure that the final product is well-aligned with stakeholder expectations and is technically feasible.

1.3 The Dissertation: Its Contribution

This dissertation presents a method for requirement analysis using user reviews. This method extracts requirements from reviews and generates goal models, enabling the visualization of requirements and the interrelationships among these requirements. The proposed method utilizes LDA topic modeling, the distance-based clustering method, and LLMs. The method dynamically selects which goal model generation method should be appropriate based on the number of reviews to ensure that the generated goal models are accurate. Introducing the LDA topic modeling and distance-based clustering method allows the proposed method to leverage a large volume of reviews for goal model generation. Additionally, introducing LLMs helps elucidate the generated goal models for developers.

The experimental results demonstrate that the proposed method outperforms the existing methods' accuracy and closely aligns with manually created goal models.

1.4 The Dissertation: An Overview

The remainder of the dissertation is organized as follows:

Chapter 2 introduces the study's essential concepts and methods. The chapter begins by examining the characteristics of user reviews. Then, this chapter introduces related research on analyzing requirements from reviews. Finally, the chapter provides an introduction to the goal model, including its concepts and key components.

Chapter 3 outlines the proposed method's workflow. It begins by describing the top-down method for refining goals and generating goal models, starting from the root and progressing to leaf goals. The chapter introduces the three key components: the LDA topic model, the distance-based clustering method, and the LLM method. It highlights their functions and the dynamic selection process based on review volume. Finally, the overall workflow is presented, illustrating how the methods are applied in practice when handling numerous reviews.

Chapter 4 discusses how to generate goal models using the LDA topic model and the process of selecting the appropriate number of topics.

Chapter 5 introduces a distance-based clustering method, which utilizes Ward's method to cluster reviews and treats clusters as goals. The parent-child relationships among goals are determined based on differences in the distance of clusters.

Chapter 6 discusses the use of LLMs. The first LLM-based method generates goal models by clustering a given set of reviews such that each cluster represents a requirement. This method allows for the association of each generated goal with its relevant reviews. The second method provides supplementary information about the requirement represented by a goal, thereby aiding the requirements analysis.

Chapter 7 evaluates the accuracy of the proposed methods in generating models. First, the LDA and the distance-based clustering method's accuracy are assessed as the number of reviews varies. Next, the proposed methods and the existing method are compared regarding goal model generation accuracy. Following this, the requirement coverage rate of the generated goals is analyzed. Finally, the practicality of the generated goal model is demonstrated.

Chapter 8 discusses the research findings and the characteristics of the proposed method, while also addressing potential limitations.

Chapter 9 summarizes related work.

Chapter 10 concludes this dissertation.

CHAPTER 2

PRELIMINARIES

2.1 User Reviews

In recent years, extensive research has highlighted the potential value embedded within user reviews, particularly in uncovering user requirements. These reviews contain insights that can guide product development and improve user experiences [15, 21, 28, 33]. However, the overwhelming quantity of user reviews presents a significant challenge. Table 2.1 shows the number of user reviews on the North American App Store from Q1 2021 to Q2 2022, with data sourced from Statista [9]. In the game category alone, on average, over 8,000 reviews are uploaded to the platform every day. While user reviews are abundant, studies indicate that only approximately 30% to 35% of these reviews contain information that is relevant to user requirements [10, 42, 51]. This low signal-to-noise ratio makes it difficult for developers to sift through the data manually and identify requirements. To address this issue, automated analysis techniques have increasingly been employed. However, the unique characteristics of user reviews complicate this process.

One major issue is the **diversity and lack of structure** in user reviews. Unlike the standardized and organized nature of requirement documents, user reviews can range from

Table 2.1: Number of reviews posted on the Apple App Store from users in the United States from the 1st quarter of 2021 to the 2nd quarter of 2022, by category.

Characteristic	Q1 2021	Q2 2021	Q3 2021	Q4 2021	Q1 2022	Q2 2022
Books	31,279	37,379	22,579	18,899	19,662	23,538
Business	50,211	53,567	42,454	47,036	45,921	44,045
Developer Tools	789	916	805	683	696	743
Education	70,690	56,615	57,712	53,589	55,346	54,291
Entertainment	225,971	176,012	142,252	138,737	147,812	150,397
Finance	358,653	183,578	156,701	154,739	150,956	126,627
Food & Drink	81,158	77,116	67,494	59,228	61,798	59,180
Games	852,931	770,262	674,578	706,429	668,326	731,830
Graphics & Design	16,216	16,456	16,078	15,569	15,200	17,572
Health & Fitness	136,805	101,746	91,377	84,555	94,917	101,945
Lifestyle	140,267	127,924	112,262	108,354	95,826	99,178
Medical	22,980	18,989	22,374	23,931	25,694	24,077
Music	62,614	51,705	45,240	50,247	53,484	46,441

brief, one-sentence comments to detailed, multi-paragraph feedback. The variation makes it difficult to group similar content automatically and effectively. For example, two reviews may both discuss similar issues with a product, but one may do so concisely while the other provides a detailed narrative, leading to challenges in clustering them accurately.

Errors and language variants further complicate the analysis. User reviews often contain spelling mistakes, grammatical errors, and the use of regional dialects or informal language. These factors introduce noise into the data, making it harder to parse the content accurately.

Semantic ambiguity in user reviews also poses a challenge. Reviews often contain words or phrases that are ambiguous or have multiple meanings. Different users may use the same words to mean different things, (homonymy) or different words to express the same concept (synonymy). This ambiguity can confuse automated analysis tools, leading to incorrect interpretations. In contrast, requirement documents tend to use precise language, as they are usually drafted by professionals who carefully choose their words to avoid confusion.

Context dependence is another area where user reviews differ markedly from requirement documents. User reviews often lack sufficient context, as they may refer to specific experiences or issues without providing the necessary background information. This lack of context can make it difficult for automated analysis tools to understand fully the intent or meaning behind a review. Conversely, requirement documents are usually comprehensive, including the necessary context and logical relationships that make them easier to interpret.

Another factor contributing to the complexity of analyzing user reviews is that **different requirements can be reflected by changes in just a few key words**. For example, consider these two reviews:

(1) I want to chat with foreigners, but LINE doesn't have a translator.

(2) I want to chat with foreigners, but LINE has regional restrictions.

While both reviews express a requirement to communicate with foreign users, the specific issues highlighted are entirely different—one focuses on the lack of a translation feature, while the other points out regional restrictions. The similarities and differences in wording reflect partially distinct user requirements and such nuances can be difficult for automated analysis tools to capture and distinguish accurately. This variability adds another layer of complexity since small changes in language can signify substantially different user requirements, which may be easily overlooked in automated analysis.

Finally, **noise in the data** is a common problem in user reviews. Reviews often contain a significant amount of irrelevant information or noise, such as advertisements, emojis, or repeated content. This noise can interfere with the analysis, making it difficult to extract requirements. Requirement documents, however, are usually curated and filtered to ensure that the information they contain is relevant and high-quality, with minimal noise.

The diversity and lack of structure, the prevalence of errors and language variants, semantic ambiguity, context dependency, subtle differences in key wording, and the presence of noise in user reviews all contribute to the inherent challenges and reduced accuracy of automated analysis. These factors starkly contrast with the standardized and curated nature of requirement documents. Consequently, analyzing user reviews requires adaptable approaches to capture the nuances accurately and extract meaningful insights from the often messy and unstructured data they present.

Notably, the problems mentioned above are found in user reviews for business-to-consumer (B2C) apps. This dissertation focuses on analyzing these reviews and extracting requirements from them. Although business-to-business (B2B) apps also include user reviews, they differ in characteristics from B2C reviews. For example, B2B reviews tend to be more complex, more comprehensive, and involve more industry-specific terminology and expertise. Therefore, handling and analyzing B2B reviews may require different methods and techniques. Currently, this research is limited to B2C reviews. Future work may explore how to extend the research methods to B2B reviews, but this exceeds the present study's scope.

2.2 Leveraging Reviews for Requirement Analysis

Recognizing that valuable insights can be found in user reviews, numerous studies have focused on analyzing and utilizing this resource to improve requirements engineering and app development. These studies emphasize that user feedback can provide critical information about desired features, usability issues, and areas for improvement. By systematically extracting, categorizing, and prioritizing information from reviews, researchers have developed various frameworks and tools to support developers in understanding user needs and planning effective updates.

Requirements engineering is pivotal in software development, as underscored by Bourque et al. [7]. Extensive prior research, as articulated by Begel and Zimmermann [3], demonstrates the criticality of extracting requirements from user reviews for successful requirement analysis. Gu and Kim [22] proposed a review summarization framework named SUR-Miner, which categorizes reviews into five types and uses a pattern-based parser to extract and evaluate software aspects. The summaries are then visualized using interactive diagrams. Feedback from developers showed that 88% found the summaries generated by SUR-Miner to be useful. Similar to this, Scalabrino et al. [54] developed CLAP (Crowd Listener for releAse Planning), a tool that can automatically categorize and cluster user reviews, and prioritize them to plan subsequent app releases. The experiment demonstrated that CLAP exhibits high accuracy in categorizing and clustering reviews, and the prioritizations it recommends are practically valuable in industrial settings. Jiang et al. [30] proposed a new method called SAFER, aimed at assisting developers in identifying and recommending new features by analyzing the descriptions of similar applications. This method first developed a tool that automatically extracts features from app descriptions. Then, by leveraging a topic model, the method identifies similar apps based on the extracted features and API names and uses a feature recommendation algorithm to recommend the features of these similar apps to the target app. In an experimental evaluation involving 533 features from 100 apps, SAFER achieved a Hit@15 score of 78.68%, meaning that in 78.68% of cases, the recommended features for a target app included the correct feature within the top 15 results. This represents an average improvement of 17.23% over the baseline method KNN+. Wu et al. [61] proposed a method named KEFE, which identifies key features highly correlated with app ratings by leveraging app descriptions and user reviews. The method combines natural language processing, machine learning, and regression analysis techniques. It extracts feature descriptions, matches them with relevant user reviews, and builds a regression model to identify features that significantly impact an app's success. Malik et al. [41] proposed a method that uses machine learning to extract app features automatically from user reviews. This method also helps users compare the features of multiple applications based on the sentiments expressed in the relevant reviews. This proposed method can be used to understand the users' preferences for a particular mobile application and reveal why users prefer one application over another.

The outputs of these studies vary in form: SUR-Miner generates categorized summaries for interactive visualization, CLAP provides prioritized lists of reviews for release planning, SAFER recommends features based on similar applications, KEFE identifies key features impacting app ratings, and Malik et al.'s method compares app features based on user sentiment. While each of these outputs provides valuable insights, they lack a comprehensive view of how individual user needs and requirements relate to one another. This limitation restricts developers from fully understanding the interdependencies and hierarchical structure among user requirements, which is essential for developing a cohesive and user-centered application.

In software development, a goal model within a requirements model is essential for illustrating the interrelationships among user needs. To visualize these interconnections, this study employs a goal model that represents the extracted requirements from user reviews. This approach generates a structured goal model that captures individual requirements and reveals how they relate to one another, supporting a more holistic understanding of user needs in the development process.

2.3 Goal Model

A goal model is a structured representation used in requirements engineering to capture and describe the objectives that a system or project needs to achieve. This model provides a clear and hierarchical view of the goals, sub-goals, and the relationships between them, allowing stakeholders to understand the overarching purposes of a system. Goal models include functional goals—what the system is supposed to do—and non-functional goals, such as performance, security, and usability. By mapping out these goals, a goal model serves to align the development process with the strategic aims of an organization or project.

In a goal model, the requirements that must be satisfied are treated as goals. These goals are organized in a hierarchical structure, comprising parent goals and sub-goals. A parent goal represents an abstract objective, while sub-goals are more detailed and specific. When all the sub-goals under a parent goal are achieved, the parent goal is considered satisfied. Figure 2.1 illustrates a portion of the goal model for the chat app LINE. In this model, the abstract goal "Ensure reliable connection" depends on its sub-goals, which refine the parent goal and provide detailed requirements. Once the two sub-goals, "Reconnect dropped calls" and "Switch between Wi-Fi and cellular," are satisfied, the parent goal "Ensure reliable connection" is considered achieved. Numerous notational approaches are employed in the domain of software development for the representation of goal models. These notations are adopted by KAOS [13], i* [62], NFR [43], AGORA [32], and UML Use Case diagrams [18]. The goal model notation presented here and used throughout the dissertation is based on KAOS.

The use of a goal model offers some advantages throughout the requirements engineering process. By defining goals and sub-goals, a goal model helps teams prioritize objectives and identify potential conflicts or dependencies between different goals early

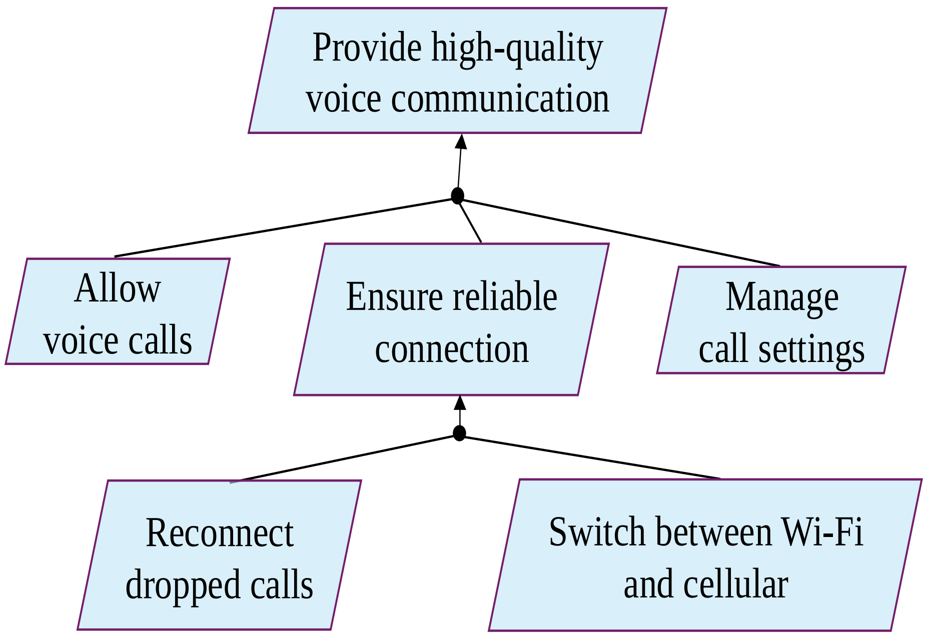

Provide high-quality voice communication
Allow voice calls
Ensure reliable connection
Manage call settings
Reconnect dropped calls
Switch between Wi-Fi and cellular

Figure 2.1: Partial goal model for LINE.

in the development process. This structured approach ensures that all stakeholders—such as business analysts, developers, and project managers—share a common understanding of the project's aims, thereby improving communication and reducing the risk of misunderstandings. When analyzing user reviews, using a goal model constructed from the reviews for analysis offers several advantages compared to simply analyzing the reviews alone.

Better Identification of Conflicts: A goal model allows for a more structured approach to identifying potential conflicts among user requirements. For example, in the case of a mobile app, one group of users might request more robust security features, while another group might prioritize ease of access and user-friendly design. These requirements can be conflicting, as stronger security measures may make the app less user-friendly. A goal model can help make these conflicts explicit, enabling developers to address them early in the design process by finding a balanced solution that satisfies both groups.

Uncovering the Common Purpose of Multiple Requirements: A goal model helps to group various user requirements under a common parent goal, revealing the underlying purpose that drives these requirements. For instance, if users express the requirement for features like faster loading, simplified navigation, and more intuitive design, these can all be grouped under the parent goal of improving the overall user experience. By organizing these requests under a single parent goal, developers can better understand that the primary objective is to enhance user satisfaction, guiding them to focus their efforts on optimizing the system's usability.

Supporting Developer Trade-offs and Prioritization: When different user requirements are aligned with the same parent goal, a goal model provides a framework for developers to prioritize and make trade-offs. For example, if a software receives a user review requesting enhanced security (e.g., multi-factor authentication) while also emphasizing ease of use (e.g., avoiding frequent password inputs), these can be grouped under the parent goal of improving user experience. The goal model helps developers prioritize which features to address first based on their impact on the primary goal. This structured approach enables the team to allocate resources efficiently, ensuring that the final design better aligns with users' core requirements. By focusing on the parent goals, developers can balance competing requirements, such as implementing robust security measures versus streamlining authentication processes, to deliver a system that meets the most critical user expectations.

Tracking and Tracing: Goal models offer the capability to track and trace the issues and requirements referenced in user reviews. Mapping the reviews to corresponding goals within the model guarantees that each goal's solutions and improvements receive the requisite attention and follow-up. This tracking and tracing mechanism elevates user satisfaction and bolsters trust in the development team.

Due to these advantages, this dissertation presents a method for automatically creating goal models from a given collection of user reviews. This method comprises three components dynamically selected to cluster reviews, enabling the top-down generation of a goal model.

CHAPTER 3

THE PROPOSED METHOD: AN OVERVIEW

This dissertation proposes a method for extracting requirements from user reviews and visualizing these requirements. The method begins with user reviews as its input. From these reviews, the proposed method extracts requirements. These requirements are then treated as goals, serving as the foundation for constructing a goal model. This goal model describes the extracted requirements and visualizes their relationships. The method's output is a goal model designed to help developers understand user requirements.

The following introduces the overall workflow of the proposed method. First, as pre-processing, the words in the reviews are lemmatized and stop words are removed. Stop words include common terms such as "a," "an," and "the," as well as words that do not describe user requirements, such as "app," "like," and "use." Finally, reviews with fewer than five words are removed, since short reviews often lack sufficient information and may provide insufficient context or detail. Afterward, the remaining reviews are used as input for generating goal models with the proposed method. The proposed method starts from the root and progressively refines the goals, generating goal models in a top-down manner. The root is considered the goal containing all reviews, and the reviews within this goal are further refined into multiple review sets, with each review set representing a sub-goal. This refinement continues iteratively until the leaf goals are generated.

There are three refinement methods, which correspond to the proposed method's components: the LDA topic model, the distance-based clustering method, and the LLM method. Among these, the LLM method is the most accurate in generating goal models and produces the most comprehensible models, but it can not handle a large volume of reviews. The LDA topic model and the distance-based clustering method can process a large number of reviews, although their accuracy may vary depending on the volume of reviews. Therefore, the method used for refinement is dynamically selected based on the number of reviews contained within the goal needing refinement.

Generating the goal model proceeds as follows. Initially, when a large number of reviews are included in the root, the LDA topic model is applied to divide the reviews into several topics. These topics are considered as sub-goals of the root. As the goals are refined, the number of reviews contained in each goal decreases. The number of reviews impacts the accuracy of goal generation. When the number of reviews decreases to a certain extent, as shown in Chapter 7, LDA accuracy drops below that of the distance-based clustering method. At this point, the distance-based clustering method continues generating the goal model. This method groups the reviews into clusters, with each cluster considered a goal. Compared the LDA and distance-based clustering methods, LLMs offer more understandable goals. However, due to token limits, generating goal models from a large number of reviews is unfeasible. This study used GPT-4 as the LLM. Generally, when the input exceeds 100 reviews, the token limit is reached, resulting in output rejection. Therefore, GPT-4 was used for refinement when the number of reviews in a goal is fewer than 100.

Algorithm 1 outlines the overall process for goal generation and sub-goal identification. This algorithm takes user reviews as its input, along with three parameters: a threshold for the number of reviews T, a review count limit L, and a reduction rate R. The output is a goal model. First, an empty goal model is initialized (Line 3), and a root goal containing all the input reviews is created (Line 4). The root goal is added to the goal model, marking the beginning of the goal generation (Line 5).

This algorithm processes each goal by recursively evaluating the number of included reviews. If the review count exceeds the threshold T, LDA topic modeling is applied to identify topics within the reviews (Line 9). These topics are treated as sub-goals of the input goal (Lines 10–12). The sub-goals are then recursively processed to ensure all goals are addressed. If the review count is below T but exceeds the review count limit L, distance-based clustering is used to avoid exceeding the LLM's token limit (Lines 14–15). The clustering process generates a sub-tree of goals, which is added to the goal model. Each leaf goal of the sub-tree is then recursively processed to refine the goal hierarchy further (Lines 16–18). For goals with a review count below the limit L, an LLM-based method is used to generate sub-goals (Lines 20–21).

Descriptive labels explaining the intent of the generated goals are automatically provided for all goals. The method for generating labels depends on the method used to generate the goals. For goals generated by the LDA and distance-based clustering methods, a subset of reviews within the goal is randomly selected. These reviews are then summarized by GPT-4, and the resulting summary forms the goal label (Lines 23-25). Each goal generated using GPT-4 (the LLM method) is labeled during goal generation. Finally, the labeled goal model is returned as the output.

Each of these methods is discussed in detail in the later chapters. Chapter 4 introduces the LDA topic model, which serves as a goal-generation method for handling a large number of reviews. This method assigns topics to reviews contained in the goals and treats these topics as sub-goals of the original goal. The LDA topic model is then applied to each sub-goal to generate topics, thereby refining the goals. This chapter also discusses certain of the LDA topic model's limitations, such as the need to predefine the number of topics, and variations precisely and proposes appropriate solutions.

Chapter 5 presents the distance-based clustering method, which is used when the LDA's precision decreases. This method calculates the distance between reviews using Ward's method and generates a dendrogram with its clusters treated as goals.

Chapter 6 presents the LLM method. Due to the input token limitation of LLMs, this method can only be applied to a small number of reviews. However, the goals obtained by the method are highly accurate. This method generates a goal model by repeatedly classifying a given set of reviews into different clusters, each representing a distinct requirement. Additionally, while this method generates a goal model, it generates the goal descriptions by extracting supplementary information about the requirements the goals represented.

Algorithm 1 Goal Generation Overview

```
1: Input: reviews, review threshold $T$, review count limit $L$, reduction rate $R$
2: Output: $goal\_model$
3: $goal\_model \leftarrow$ create_empty_goal_model()    ▷ Initialize an empty goal model
4: $root\_goal \leftarrow$ create_goal(reviews)    ▷ Create the root goal
5: add_goal_to_model($goal\_model$, $root\_goal$)
6: function PROCESS_GOAL($goal\_model$, $goal$)
7:     $review\_count \leftarrow$ get_review_count($goal$)
8:     if $review\_count > T$ then
9:         $sub\_goals \leftarrow$ LDA_Topic_modeling($goal$)
10:        for all $sub\_goal$ in $sub\_goals$ do
11:            add_sub_goal_to_model($goal\_model$, $goal$, $sub\_goal$)
12:            PROCESS_GOAL($goal\_model$, $sub\_goal$)
13:    else if $review\_count > L$ then    ▷ Use distance-clustering to avoid excessive reviews exceeding LLM's token limit
14:        $sub\_tree \leftarrow$ Distance-based_Clustering_for_Goal_Generation($goal$, $L$, $R$)
15:        add_subtree_to_model($goal\_model$, $goal$, $sub\_tree$)
16:        $leaf\_goals \leftarrow$ get_leaf_goals($sub\_tree$)
17:        for all $leaf\_goal$ in $leaf\_goals$ do
18:            PROCESS_GOAL($goal\_model$, $leaf\_goal$)    ▷ Recursively process leaf goals
19:    else
20:        $sub\_tree \leftarrow$ LLM-based_Goal_Model_Generation($goal$)
21:        add_subtree_to_model($goal\_model$, $goal$, $sub\_tree$)
22: PROCESS_GOAL($goal\_model$, $root\_goal$)    ▷ Start processing the root goal
23: for all $goal$ in $goal\_model$ do    ▷ Labeling goals generated by LDA and clustering
24:     if $goal$ is generated by LDA or Distance-based Clustering then
25:         $goal.label \leftarrow$ LLM-based Label Generation($goal.reviews$)
```

CHAPTER 4

GOAL MODEL GENERATION VIA LDA

Topic models are statistical tools used to uncover hidden semantic structures within documents [49]. Topic modeling is an unsupervised method that identifies or extracts topics by recognizing patterns in the data. By analyzing these patterns, including word clusters and word frequency, various topics present in the documents can be identified.

Topic models like Latent Dirichlet Allocation (LDA) [6], PLST [27], and the Pachinko allocation model [34] are widely used for classifying and clustering reviews. In this study, the LDA model was selected, as it is among the most commonly used algorithms for topic modeling [6]. The term "latent" refers to something hidden, unrealized, or not directly observable, while "Dirichlet" relates to the assumption that topics and words in the documents follow a Dirichlet distribution. The word "allocation" refers to the assignment of topics, meaning that each document may encompass multiple topics, with each word being generated by a specific topic. In addition, compared to other topic models, LDA excels in several areas for the following reasons [1]. Firstly, LDA effectively captures the latent topic structure within reviews and performs well in both accuracy and stability. Second, LDA is computationally efficient, often requiring less computation time than other complex topic models, which allows it to provide results speedily when handling large-scale datasets. Additionally, LDA excels regarding interpretability. The topics and word distributions generated by LDA offer a clear thematic structure, making it easier for analysts to understand the components of each topic and their function in the reviews.

4.1 LDA: Goal Modeling

To generate goal models, the LDA model is utilized for analyzing the reviews contained within each goal in a top-down approach. This involves applying LDA to the reviews for a parent goal to refine sub-goals, with each sub-goal corresponding to one of the identified topics. By predefining the number of topics, denoted by K, the model generates K topics, each representing a sub-goal under the parent goal.

During sub-goal generation, LDA assumes that all reviews comprise a mixture of topics, each defined by a specific distribution of words. These reflect the underlying themes present in the reviews. Each review is assigned a probability distribution over the topics based on the word distributions of the topics. Based on this distribution, the most probable topic for a review is considered the representative topic. This probabilistic framework allows LDA to model the hidden thematic structure within the text data.

The LDA-based goal model generation proceeds as follows. LDA is first applied to the root goal, which contains all the reviews, with the number of topics predefined. Each review is then assigned to the topic with the highest probability, and these topics become the sub-goals of the root goal. Since many reviews do not provide useful information, due to their limited vocabulary, these uninformative reviews, often cluster into specific topics. For example, some topics may consist only of praise, while others may contain only criticism. Filtering out these topics helps reduce noise and improve modeling accuracy. Therefore, after applying LDA to model the root goal, all topics are manually checked to identify those that include many reviews lacking useful information. These topics typically contain a large proportion of reviews, with 80% to 90% providing little to no valuable content. Removing such topics helps eliminate noise, resulting in a more accurate goal model. Once these topics are removed, the remaining topics are used to continue generating the goal model.

By applying LDA to the reviews contained in each remaining sub-goal of the root goals, the topics of these sub-goals are generated and treated as sub-goals of those sub-goals. This process is repeated until the number of reviews within the goals decreases to a certain level, at which point the accuracy of the LDA method becomes lower than that of the distance-based clustering method. At this stage, the distance-based clustering method is more suitable for further refining the goals. The relationship between the number of reviews and the accuracy of both methods will be discussed in Chapter 7.

4.2 LDA: Limitations and Solutions

Despite its utility, the LDA topic model has some notable drawbacks that can limit its effectiveness in goal modeling.

4.2.1 Predefining the Number of Topics

A limitation of the LDA model is having to predefine the number of topics. Reviewing all the reviews manually to determine the optimal number of topics is impractical and inefficient. To address this issue, this study utilized a coherence score. The coherence score can be used to assess the interpretability of a generated topic by measuring the co-occurrence of the most probable words in the reviews associated with that topic. For instance, a topic containing words like "chat," "voice," "app," "friend," and "send" is easily understood as relating to a chat application, leading to a high coherence score. Conversely, a topic with words such as "review," "undated," "receive," "teacher," and "clients" may be more challenging to interpret, resulting in a lower coherence score. Among the several methods available to calculate coherence scores, this study adopted the CV metric due to its extreme accuracy [53]. The CV metric is also the default in the Gensim topic coherence pipeline module [50].

The coherence score calculation works as follows. Suppose that LDA has already been applied and that a predefined number of topics have been obtained. This means that for each topic, the set of keywords with probabilities and the set of reviews have been determined. In the proposed method's current implementation, the coherence score is calculated for each topic by providing as input to GenSim, the top-N keywords and the topic reviews. This produces in a coherence score for each topic.

When processing the reviews associated with a goal, LDA topic modeling and coherence score calculation are repeatedly performed by varying the number of topics from small to large. Finally, the optimal number of topics is selected based on the coherence score. Figure 4.1 shows the changes in coherence score during LDA topic modeling on 1000 reviews for LINE. Starting with 5 topics and in increments of 5 increasing to 40 topics, coherence scores were computed at each increment. For example, when the number of topics was set to 5, LDA generated 5 topics using the 1000 reviews from LINE. The coherence score for each of these 5 topics was then calculated individually. The average of the coherence scores for these 5 topics represents the coherence score for this number of topics. When the gain in the coherence score becomes marginal or negative, the process is stopped and the final number of topics is decided. Selecting the number of topics with a high coherence score is likely to yield subgoals that correspond to individual requirements.

Note that the number of requirements can vary depending on the number of reviews and the app's characteristics. A larger number of reviews often indicates a broader range of requirements, necessitating more topics. Additionally, the genre of the app can influence the number of requirements. For example, with 1,000 reviews, the requirements for LINE may cover areas such as video calls, group chats, translation, and stickers. In contrast, for YouTube, the requirements are mainly focused on ads, thus needing fewer topics. Selecting the number of topics using the coherence score produces appropriate subgoals regardless of the apps' characteristics.

By repeatedly analyzing the reviews using LDA topic modeling, the upper part of the goal model, including, for example, the root node and its subgoals, is generated.

Algorithm 2 outlines the process of applying LDA topic modeling to generate sub-goals from a given goal. The algorithm takes goal G, which contains a set of associated reviews, as the input and outputs a list of sub-goals. First, the LDA model is applied to the reviews within G with an increasing number of topics set to 5, 10, 15, and so on (Line 3). The resulting topics are generated (Line 4). Next, coherence scores for these topics are calculated (Line 5). If the average coherence score does not increase, the topics and number of topics from the previous step are restored (Lines 6–8), and the loop is terminated (Line 9). An empty list is initialized to store the generated sub-goals (Line 10). For each identified topic, a sub-goal is created, and reviews related to the topic are assigned to the corresponding sub-goal (Lines 11-14). These sub-goals are added to the list of sub-goals.

If the input goal G is the root goal, the algorithm evaluates the reviews within each sub-goal. Sub-goals containing mostly uninformative reviews are removed to improve the relevance and clarity of the goal model (Lines 16-18). Finally, the algorithm returns the list of generated sub-goals.

4.2.2 LDA: Performance Instability

The LDA topic model's performance can be unstable, particularly when dealing with varying amounts of reviews. As the number of reviews fluctuates, the precision of the

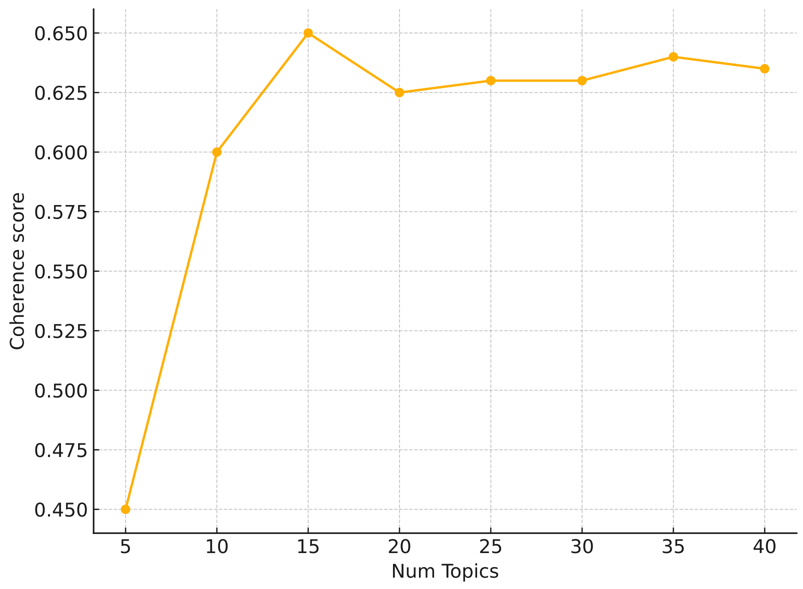


Figure 4.1: Choosing the optimal number of topics with coherence scores.

topic modeling results can vary. This variability can lead to less reliable outcomes, particularly when the number of reviews is limited. As explained in the previous chapter, to address this limitation and improve the precision of the proposed goal modeling method, the proposed method uses two other methods, namely, the distance-based clustering method and the LLM-based method. These help to reduce the LDA's instability by providing alternatives when dealing with smaller datasets.

As shown in a later section, experiments revealed that LDA performs more consistently when applied to a larger dataset. Consequently, to ensure accuracy, it was decided to apply the LDA topic model only, when sufficient reviews were available. In the cases of smaller datasets, the distance-based clustering method and LLMs were relied upon to maintain high accuracy in goal modeling. This combined method allowed us to harness the strengths of each method, optimizing performance across different scenarios.

Algorithm 2 LDA Topic Modeling

```
1: **Input:** goal $G$
2: **Output:** $subgoals$
3: **for** $num\_topics \in \{5, 10, 15, \dots\}$ **do**
4:     $topics \leftarrow$ apply_LDA$(G.reviews, num\_topics)$
5:     $coherence\_scores \leftarrow$ calculate_coherence_scores$()$
6:     **if** average coherence score does not increase **then**
7:         $topics \leftarrow previous\ topics$
8:         $num\_topics \leftarrow previous\ num\_topics$
9:         **break**
10: $subgoals \leftarrow$ []                                ▷ Initialize an empty list for subgoals
11: **for** $topic$ **in** $topics$ **do**
12:     Create sub-goal $sub\_goal$                         ▷ Generate sub-goal for the topic
13:     $sub\_goal.reviews \leftarrow$ reviews related to $topic$   ▷ Assign topic-related reviews to
    the sub-goal
14:     Add $sub\_goal$ to $subgoals$
15: **if** goal $G$ is root goal **then**
16:     **for** $sub\_goal$ **in** $subgoals$ **do**
17:         **if** reviews in $sub\_goal$ are mostly uninformative **then**
18:             Delete $sub\_goal$                      ▷ Delete sub-goals that are uninformative
```

CHAPTER 5

DISTANCE-BASED CLUSTERING ALGORITHM

The LDA topic modeling requires the number of topics to be predefined, which poses a challenge for automating the process. Additionally, the model's accuracy can vary depending on the volume of reviews. To address these issues, a distance-based clustering method was introduced. This approach vectorizes reviews and clusters the vectors to form a hierarchical tree structure, which serves as the framework of the goal model. Clusters represent goals, and the hierarchical structure reflects the relationships between sub-goals and parent goals.

Algorithm 3 illustrates a simple method for generating a goal model by clustering reviews. The resulting clusters are considered goals, and their aggregation can be regarded as inverting the refinement process in goal modeling. First, reviews are split into words. Next, the words are added to a dictionary, and a list representing the Bag of Words (BoW) [38] is generated from the dictionary. The BoW represents the words in the reviews and their occurrence frequency. Then, the generated BoW list is stored in a matrix in vector form. For example, suppose the following two one-sentence reviews:

Algorithm 3 Clustering method for Goal Model Generation

```
1: Input: user reviews
2: Output: $goal\_model$
3: for $review$ in $reviews$ do
4:     $sentencelist \leftarrow$ divided sentences in review    ▷ segment reviews with commas, periods, exclamation marks, and question marks
5:     $bows \leftarrow$ bag-of-words (BoW) of $sentencelist$    ▷ generate BoW
6:     for {$word\_id, frequency$} in $bows$ do
7:         $array[review][word\_id] \leftarrow array[review][word\_id] + frequency$
8: $dendrogram \leftarrow$ apply Ward's method to $array$    ▷ generate dendrogram
9: Generate $goal\_model$ from $dendrogram$
```

(1) I cannot delete any documents from my company on the web now.

(2) It does not allow me to delete any documents.

After lemmatization and filtering, the BoW lists are ["delete": 1, "document": 1, "company": 1, "web": 1] and ["delete": 1, "document": 1]. These lists are then stored in a matrix.

Ward's method [59] (employing hierarchical clustering) is applied to the matrix, and the result is the clustering of reviews. These clusters are treated as goals, and through a bottom-up clustering process, they progressively form the goal model. The next section provides a detailed explanation of the clustering process in Ward's method and discusses its limitation.

5.1 Ward's Method

Ward's method, introduced by Joe H. Ward, Jr., is applied to cluster reviews and generates goal models. It is particularly favored in disciplines like linguistics because it can create compact and well-balanced clusters [57]. Ward's method is a type of hierarchical clustering. It starts by treating each individual review as its own cluster and then iteratively merges the closest pair of clusters, continuing this process until all reviews are grouped into a single cluster.

Ward's method implements the steps below. **Initialization:** At the outset, each review is assigned to its own cluster. The squared Euclidean distance between each pair of clusters is then calculated. **Merging Clusters:** In each subsequent step, the two clusters that are closest to each other—i.e., those with the smallest distance—are merged into a single cluster. This process reduces the total number of clusters by one. **Iteration:** The merging process is repeated iteratively. After each merge, the distances between the new cluster and the remaining clusters are recalculated, and the two closest clusters are again merged. This continues until all reviews are consolidated into one cluster.

During the merging process, clusters are combined based on the minimum variance criterion. Specifically, the distance between clusters is determined by the change in the error sum of squares (ESS) resulting from their merger, as expressed in the following equation:

$$\Delta E = \sum_{i \in A \cup B} (x_i - \mu_{A \cup B})^2 - \left(\sum_{i \in A} (x_i - \mu_A)^2 + \sum_{i \in B} (x_i - \mu_B)^2 \right) \tag{5.1}$$

In this formula, x_i represents the feature vector corresponding to review i, μ_A and μ_B are the centroids of clusters A and B, respectively, and $\mu_{A \cup B}$ is the centroid of the combined cluster. ΔE represents the reduction in the error sum of squares (ESS) resulting from merging clusters A and B. A smaller ΔE indicates a more cohesive cluster, as it minimizes the variance within the cluster. Using this criterion, Ward's method iteratively identifies and merges the most compatible clusters. For instance, if two clusters are characterized by similar terms, such as "edit" and "save," merging them reduces ΔE, suggesting that they belong to the same higher-level goal.

Figure 5.1 illustrates the application of Ward's method in clustering 10 reviews. Initially, 10 clusters are created, one for each review, and the distances between them are computed. The closest pair of clusters, {0} and {6}, are first merged to form a new cluster {0, 6}, reducing the total number of clusters to nine. Next, the newly formed cluster {0, 6} is merged with the closest remaining cluster, {8}, resulting in a cluster {0, 6, 8} and reducing the number of clusters to eight. This iterative merging continues until all reviews are combined into a single cluster.

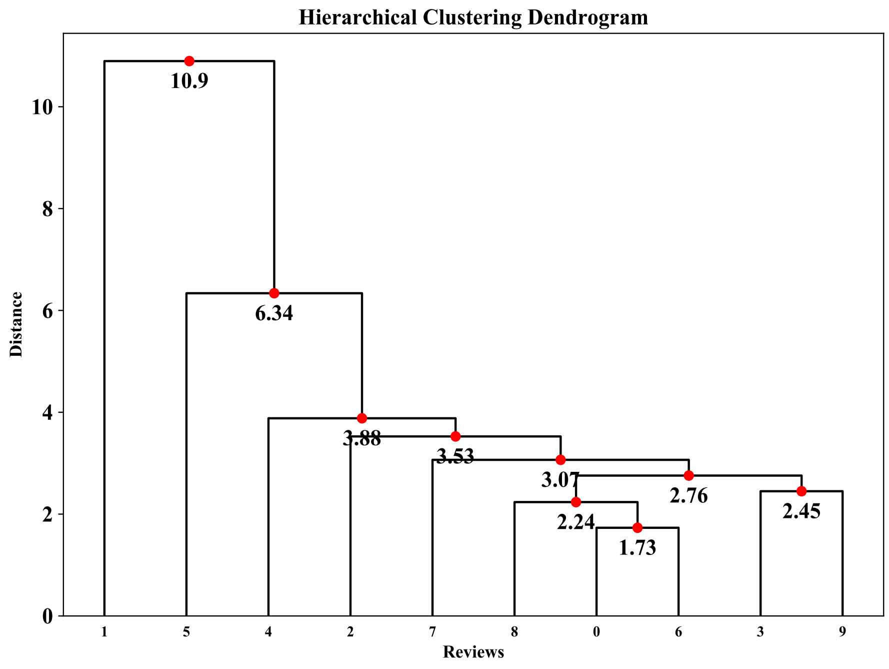


Figure 5.1: Clustering 10 reviews using Ward's method.

The advantage of Ward's method is its ability to automatically cluster reviews, making it an effective tool for generating goal models. However, this method also has a limitation that hinders its application to large-scale review datasets, as explained in the next subsection.

5.1.1 Ward's Method: Limitation

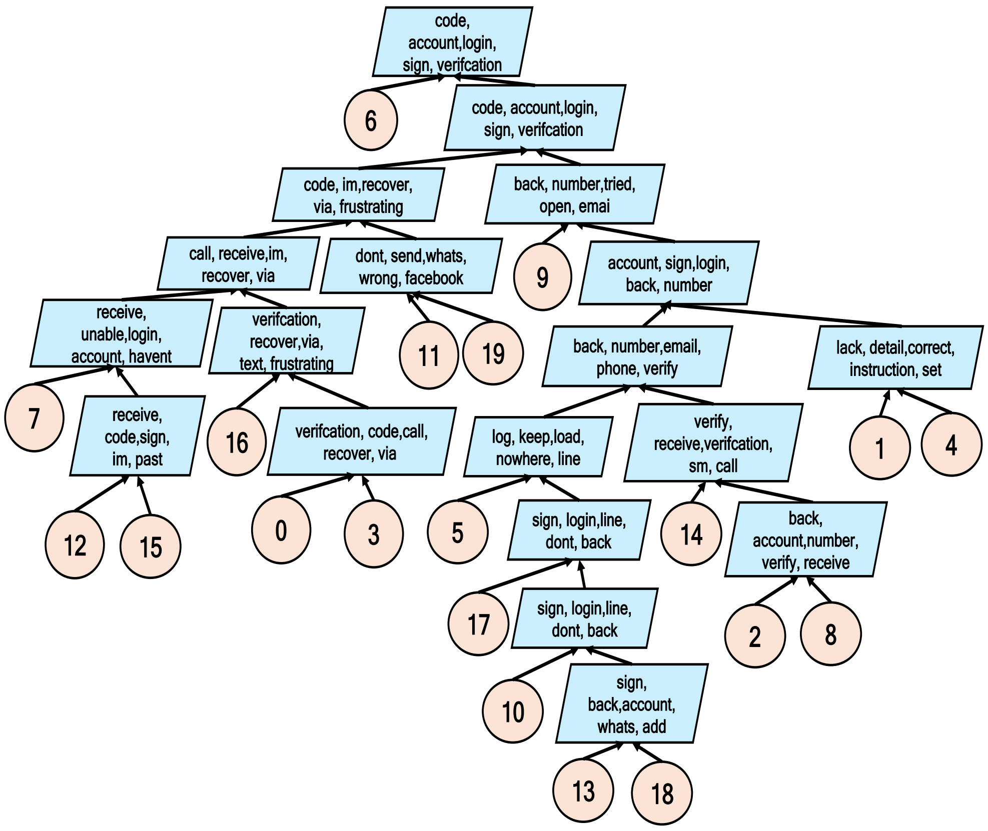

code, account,login, sign, verifcation
6
code, account,login, sign, verifcation
code, im,recover, via, frustrating
back, number,tried, open, emai
call, receive,im, recover, via
dont, send,whats, wrong, facebook
9
account, sign,login, back, number
receive, unable,login, account, havent
verifcation, recover,via, text, frustrating
11
19
back, number,email, phone, verify
lack, detail,correct, instruction, set
7
receive, code,sign, im, past
16
verifcation, code,call, recover, via
log, keep,load, nowhere, line
verify, receive,verifcation, sm, call
1
4
12
15
0
3
5
sign, login,line, dont, back
14
back, account,number, verify, receive
17
sign, login,line, dont, back
2
8
10
sign, back,account, whats, add
13
18

Figure 5.2: Goal model generated using Ward's method.

Figure 5.2 shows a portion of the goal model generated by Ward's method using 20 reviews of LINE. In this figure, the numbered circles represent singleton clusters, where each cluster contains only a single review, identified by its corresponding number. These singleton clusters serve as the initial state in the hierarchical clustering process. The nodes indicate goals derived from reviews, with each goal described by keywords extracted from the reviews. This figure demonstrates a limitation of Ward's method as follows.

Ward's method, prior to clustering, first generates a cluster for each review and treats these clusters as leaf goals. This means that in the initial phase, every single review is assigned its own goal, creating a large number of initial goals. Then, Ward's method allows only two clusters to be merged when generating parent goals, meaning that each goal's sub-goals are limited to two. With a large number of initial goals, this merging process must be repeated many times, significantly increasing the number of goals. As the number of goals increases, the requirements can become obscured within the vast pool of generated goals. The high volume of goals makes it harder for developers to extract requirements from reviews, as the requirements may be hidden among numerous redundant or irrelevant goals.

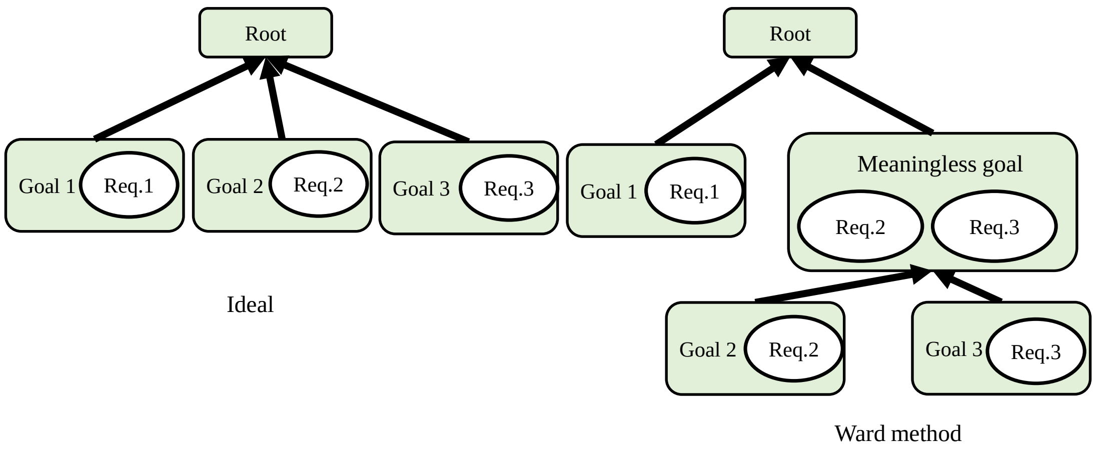


Figure 5.3: Limitation of Ward's method.

Figure 5.3 illustrates the differences between the ideal goal model and the goal model generated using Ward's method. Ideally, three goals (Goal 1, Goal 2, and Goal 3) correspond to three requirements (Requirement 1, Requirement 2, and Requirement 3), ensuring a clear mapping. However, Ward's method, constrained by its binary hierarchical structure, can only generate two goals: one meaningful goal (Goal 1) and another meaningless goal. This redundant goal is then further broken down into two sub-goals (Goal 2 and Goal 3). This limitation leads to the creation of redundant goals.

When generating goal models for a large volume of reviews, the number of redundant goals increases significantly. This both inflates the complexity of the generated model and hinders developers' ability to comprehend and analyze user requirements effectively. This weakness indicates the need for further development of alternative methods to handle large datasets better and produce more accurate goal models.

5.2 Dendrogram-Based Goal Model Refinement

Although the clustering steps are similar to Ward's method, the proposed distance-based clustering method goes further by evaluating the hierarchical relationships of goals based on the differences in distance values between clusters within the dendrogram. The distance value measures the similarity between two clusters in clustering. It reflects the change in the total within-cluster sum of squares when two clusters are merged. Specifically, it is calculated as

$$d = \sqrt{2\Delta E},$$

where ΔE is derived from Equation 5.1. A smaller difference in distance values between a cluster and its parent cluster indicates that the requirements they describe are similar. Therefore, these clusters are not treated as independent goals; instead, the cluster with the largest distance value is selected as the goal. This method significantly reduces the number of goals in the goal model while accurately distinguishing different requirements.

In the distance-based clustering method, clusters with similar distance values are grouped under the same parent goal. This method creates boundaries from the top down. Clusters located above these boundaries are designated as parent goals, while those below are categorized as sub-goals. The determination of boundary values takes into account the distances between clusters and an artificially set reduction rate, R ($0<R<1$). The parameter R is crucial in defining both the number of goal model layers and the total number of goals. Generally, a higher R value produces fewer goals.

Specifically, the distance-based clustering method begins by establishing a root for the cluster that includes all the reviews, which is recognized as the largest cluster. A borderline is established by multiplying the distance of the largest cluster by a factor denoted as R. This borderline acts as a threshold, separating clusters based on distance, thereby determining the parent-child relationships between the goals during goal model generation. The distance-based clustering method examines clusters that fall below this threshold during the subsequent traversal. Specifically, this method identifies clusters that have parent clusters positioned above the borderline. For these identified clusters, goals are identified. Given that these clusters are subsets of the largest cluster, the goals identified in this process serve as sub-goals of the root goal. The clustering method then proceeds by generating new borderlines based on the distances of these identified clusters, again multiplying the distance by the factor R. This iterative process of traversing clusters, identifying goals, and generating subsequent borderlines continues, with each iteration further refining the goals. This cycle repeats until the number of reviews in a cluster falls below the limit set for the proposed LLM method, since the LLM method excels with a smaller number of reviews. The distance-based clustering method breaks down the largest cluster into smaller clusters, each with its own set of refined goals, by iteratively applying the borderline calculation and goal identification process.

Algorithm 4 outlines the distance-based clustering process for goal generation. This algorithm takes a goal G, a review count limit L, and a reduction rate R as inputs, and outputs a sub-tree representing the goal structure. First, a sub-tree is initialized with the input goal G as its root (Line 3). Then, the reviews included in the goal G are vectorized, and a dendrogram is generated using Ward's method (Lines 4-9). This step is the same as in Algorithm 3. The function GENERATE_SUB_GOALS operates recursively to identify sub-goals and construct the hierarchical structure. For each input goal, the function retrieves the cluster corresponding to the goal and calculates its distance (Lines 11-12). A threshold is set by multiplying the cluster distance with the reduction rate R (Line 13). Clusters that meet the threshold criteria—having distances below the threshold but with parent clusters that exceed the threshold—are identified as valid sub-clusters (Line 14).

Each valid sub-cluster is processed as follows: if the number of reviews in the sub-cluster exceeds the review count limit L, a sub-goal is created with the sub-cluster's reviews and added to the sub-tree as a child of the input goal (Lines 16-18). The function is then called recursively to process the new sub-goal (Line 19). The process continues until no further valid sub-clusters can be identified or the review count limit is not met. Finally, the constructed sub-tree, representing the goal structure, is returned as the output of the algorithm).

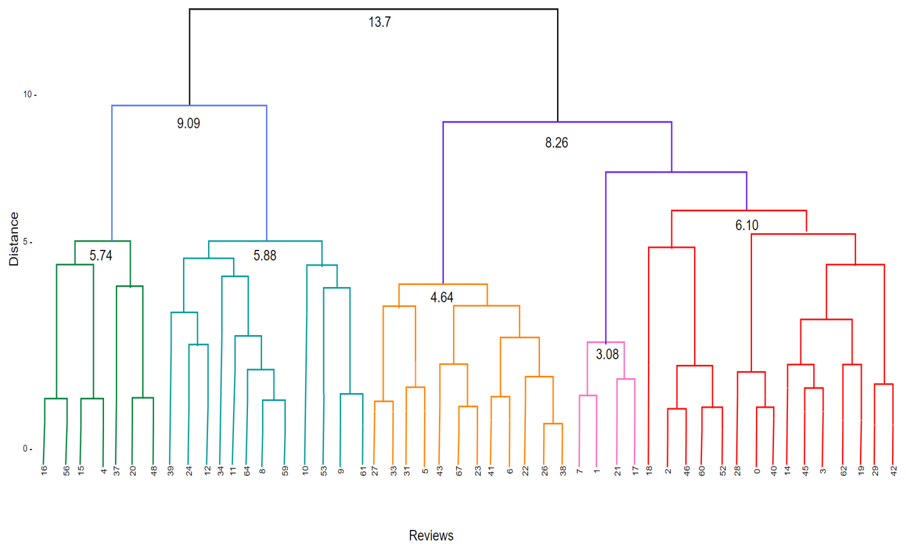


Figure 5.4: Clustering reviews by distance.

Figure 5.4 illustrates an example of a dendrogram resulting from clustering 50 reviews within a goal using Ward's method. The horizontal axis represents the review IDs. The numbers indicate the square root of twice the increase in the total within-cluster sum of squares after merging two clusters, i.e., $\sqrt{2\Delta E}$, where ΔE is derived from Equation 5.1.

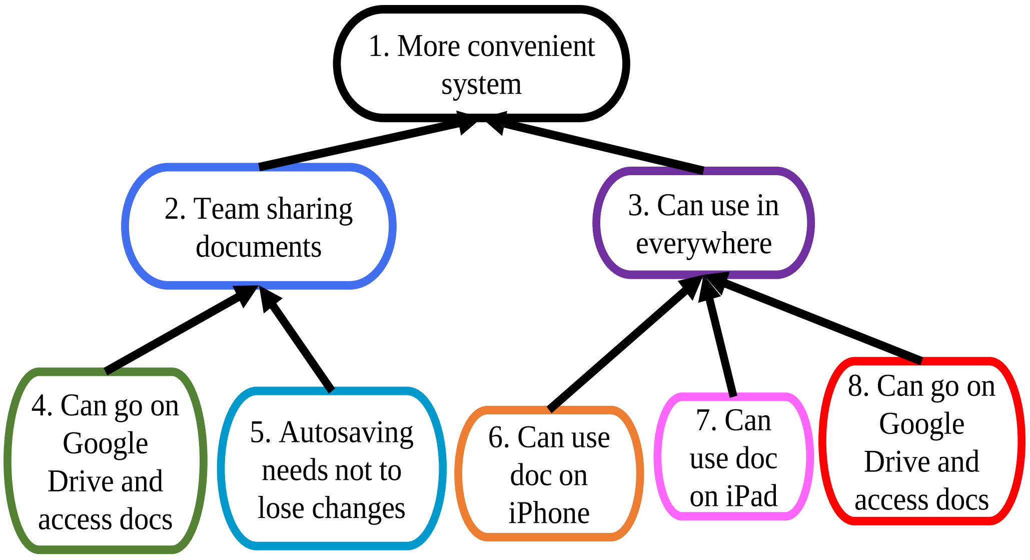

1. More convenient system
2. Team sharing documents
3. Can use in everywhere
4. Can go on Google Drive and access docs
5. Autosaving needs not to lose changes
6. Can use doc on iPhone
7. Can use doc on iPad
8. Can go on Google Drive and access docs

Figure 5.5: Goal model generated by the distance-based clustering algorithm.

The reduction rate was set to 0.7 based on the results of preliminary experiments, where several reduction rates were tested, including 0.5, 0.6, 0.7, and 0.8. The goal model generated with a reduction rate of 0.7 is the most similar to the manually generated model. The root node represents all reviews, with a distance value of 13.7. Thus, the boundary value becomes 9.59, corresponding to 70% of the distance value of the root cluster. The proposed distance-based method then traverses the clusters and identifies those that have a greater distance value than the boundary. In this case, no clusters have a lager distance value. Consequently, only the root cluster constitutes the root goal, and the two child clusters constitute two sub-goals, goals 2 and 3. These child clusters are highlighted in blue and purple. For Goals 2 and 3, boundaries are generated separately. For each goal, the descendant clusters above the boundary line are merged to form the same goal. For Goal 2, the identified clusters are green and cyan, while for Goal 3, the identified clusters are orange, pink, and red. Finally, the process stops when the number of reviews included in a goal becomes too small. The resulting goal model is depicted in Figure 5.5. In this example, for illustrative purposes, the LLM was not used. Instead, the descriptions of the goals were manually added after checking all the reviews within each goal.

Algorithm 4 Distance-based Clustering for Goal Generation

```
1: Input: goal $G$, review count limit $L$, reduction rate $R$
2: Output: $sub\_tree$    ▷ Return the constructed sub-tree
3: $sub\_tree \leftarrow$ initialize_tree($G$)    ▷ Initialize a sub-tree with root as $G$
4: for $review$ in $G.reviews$ do
5:     $sentencelist \leftarrow$ divided sentences in review    ▷ segment reviews with commas, periods, exclamation marks, and question marks
6:     $bows \leftarrow$ bag-of-words (BoW) of $sentencelist$    ▷ generate BoW
7:     for {word_id, frequency} in $bows$ do
8:         $array[review][word\_id] \leftarrow array[review][word\_id] + frequency$
9: $dendrogram \leftarrow$ ward_method($G.reviews$)    ▷ Generate dendrogram using Ward's method based on reviews in $G$
10: function GENERATE_SUB_GOALS($dendrogram$, $goal$, $L$, $R$)
11:     $cluster \leftarrow$ get_cluster($dendrogram$, $goal$)    ▷ Get the cluster corresponding to $goal$
12:     $cluster\_distance \leftarrow$ get_distance($cluster$)    ▷ Get distance of the cluster
13:     $threshold \leftarrow cluster\_distance * R$    ▷ Set threshold based on reduction rate
14:     $sub\_clusters \leftarrow$ find_valid_clusters($dendrogram$, $threshold$)    ▷ Identify clusters whose distances are below the threshold but whose parent clusters have distances above the threshold.
15:     for $sub\_cluster$ in $sub\_clusters$ do
16:         if review_count($sub\_cluster$) > $L$ then
17:             $sub\_goal \leftarrow$ create_sub_goal($sub\_cluster$)    ▷ Create a sub-goal containing the reviews from the cluster
18:             add_to_tree($sub\_tree$, $goal$, $sub\_goal$)    ▷ Add sub-goal as a child of $goal$ in the sub-tree
19:             GENERATE_SUB_GOALS($dendrogram$, $sub\_goal$, $L$, $R$)    ▷ Recursively process sub-goal
20: GENERATE_SUB_GOALS($dendrogram$, $G$, $L$, $R$)
```

CHAPTER 6

TWO METHODS FOR LEVERAGING LLM CONVERSATIONS

The proposed LDA topic modeling and distance-based clustering method can handle a large volume of reviews. However, the resulting goal models sometimes remain difficult to interpret. To enhance interpretability, the potential of using LLMs to generate goal models has been explored. By leveraging the natural language understanding capabilities of LLMs, more comprehensible goal models can be generated.

To generate goal models, this study used the cutting-edge Generative Pre-trained Transformer 4 (GPT-4), an LLM developed by OpenAI. Its main purpose is to engage in interactive conversations with users, providing responses that are contextually appropriate across a vast array of prompts and queries. This advanced language model exhibits exceptional capabilities in a variety of tasks, including chatbots, language translation, text generation, and summarization.

GPT-4 can be directly used for generating goal models. When provided with user reviews and prompts such as “If I give you a certain number of user reviews, can you generate a multi-level structured goal model, even if it is incomplete?”, GPT-4 outputs

(1) Multi-Device Support
    (a) Seamless Account Retrieval
    (b) Synchronization of Emails and Surprises across Devices
(2) Account Transfer and Device Upgrades
    (a) Easy Transfer of Data and Contacts
        i. Transfer Contacts

Figure 6.1: List generated directly by GPT-4.

a list of items. Figure 6.1 illustrates the results of goal model generation using 70 user reviews of LINE. Based on this list, a goal model could be manually created. Each item in the list is considered a goal. The description text for each item is treated as the label for the corresponding goal. The refinement relationships between items are regarded as parent-child relationships among the goals.

Despite the potential for generating goal models, the direct use of GPT-4 shown above does not sufficiently harness its capabilities. The following sections discuss better ways to use GPT-4 for goal modeling.

6.1 Obtaining Goals with Relevant Reviews

One limitation of solely using GPT-4 to generate goal models based on user reviews is the difficulty in clearly representing the relationship between the reviews and the corresponding goals. More specifically, unlike the goals produced by the other two methods, the goals generated by GPT-4 do not come with relevant reviews. One may be able to understand the requirement from the goal description (i.e., label). However, the generated goal descriptions are often simplified, possibly omitting details of the requirements and their context within the reviews. Consequently, the information developers derive from these goals is insufficient for a comprehensive understanding of user requirements. Even if developers request a complete display of all relevant reviews, GPT-4 typically provides only a limited selection, making it difficult to grasp the full scope of user requirements.

The proposed goal model generation method solves the challenge of establishing relationships between goals and reviews. In this method, the task of generating goal models is redefined as the task of classifying reviews into clusters, where each cluster represents a distinct goal. GPT-4 is utilized to determine the appropriate cluster for each review, thereby establishing the relationship between goals and their corresponding reviews. This allows developers to identify and easily refine specific goals based on various factors, such as the number of associated reviews or the content within those reviews. For instance, developers could decide to refine goals with more related reviews, as a larger volume of reviews typically indicates greater user interest or more complex requirements. Conversely, goals associated with only a few reviews may not require significant refinement, which could reduce GPT-4's processing time and enhance the goal analysis process for developers.

The process is divided into two key steps. The first involves using GPT-4 to cluster reviews, with each resulting cluster being interpreted as a goal. The prompt for this step is:

Prompt 1: Can you cluster the following reviews?

The second step occurs after developers analyze the generated goals and decide which goals to refine. GPT-4 is then employed to refine the selected goals. For example, if the first cluster is selected for refinement, the prompt for this step is:

> Prompt 2: Can the first category be refined, and if so, what would the relevant reviews in the subdivided categories look like? By "relevant reviews," I mean the reviews I provided earlier. Cannot generate reviews; all reviews should belong to the first category classified earlier. Each comment should belong to only one subcategory, and each subcategory should be akin to a goal in the goal model in the requirement model.

It is essential to include the phrase "Cannot generate reviews" in the prompt to prevent the model from producing inaccurate outputs. Additionally, it is important to clearly state that each review should be associated with only one goal.

Figure 6.2 shows the list of goals generated using this method, with the input consisting of user reviews within a parent goal. Note that this parent goal has been generated using the distance-based clustering method. The output includes categories 1 and 2, regarded as two sub-goals of the parent goal. The relevant reviews listed under each category are considered reviews belonging to the sub-goal. For the two categories, the LLM provides a summary sentence for each, which becomes the label for the corresponding sub-goals: "Seamless Account Transfer" and "Effective Account Recovery." Meanwhile, these labels are recorded to determine whether any goals are redundant and to decide when to terminate the goal-generation process.

For the reviews within each sub-goal, the LLM method repeats the above steps to generate sub-goals for these goals. This iterative process gradually constructs the hierarchical structure of the goal model. After each sub-goal is generated, the LLM method compares its label with the previously recorded labels. If a similar label is found, it indicates that the generated goal does not describe a new requirement. In such cases, the goal is deleted, and no further sub-goals are generated for it.

By using this method, developers can leverage the strengths of GPT-4 in generating goal models while simultaneously visualizing the connections between user reviews and

Category 1: Seamless Account Transfer

Relevant Reviews: "Multi device support is a joke."

"This has got to be one of the worst account transfer experience ever. Why do you need to make logging in to a new device so hard?"

...

Category 2: Effective Account Recovery

Relevant Reviews: "So LINE blocks you if you don't change your account over when you get a new phone. This is about the dumbest thing ever."

"I lost all my chat history because I changed my phone and forgot my password."

...

Figure 6.2: List of goals generated by GPT-4 using user reviews of LINE

corresponding goals.

6.2 Requirement-Extraction: the LLM Method

This section proposes how GPT-4 can be used to produce supplementary information that explains the requirement represented by a specific goal. This extra information can be annotated with the goal node, providing helpful insights to system developers. This information extraction process is performed by supplying GPT-4 with the reviews associated with the goal. However, when simply asked to extract requirements from user reviews, GPT-4 often identifies features instead. In the context of requirement engineering, while requirements and features are closely related, they represent distinct concepts. The fundamental difference between requirements and features lies in the former referring to the problem domain, while the latter refers to the implementation domain.

A requirement refers to the essential characteristics, performance criteria, or services that a system must fulfill. It encapsulates the expectations of users, conditions that the system must satisfy, or functionalities to be implemented. Requirements are typically categorized into functional and non-functional requirements. Functional requirements outline what the system should accomplish, such as specific tasks or operations it must perform. Conversely, non-functional requirements detail how the system should operate, covering aspects like performance, reliability, security, and user experience.

In contrast, a feature is a specific operation or task that the system can perform, representing a concrete capability or behavior identified during the requirement analysis and definition process. Features focus on the specific ways in which a system can meet its requirements, often representing how those requirements are realized in practice. While requirements are generally proposed by users, features are defined and implemented by developers.

Although user reviews include both requirements and features, it is important to recognize that users are not developers. While their expressed requirements should be considered seriously, the features they describe are not always essential for implementation. For instance, some features could be too difficult to implement, or their requirements could be met through alternative solutions. Consequently, the features extracted by GPT-4 from these reviews may not be sufficient to guide software development. This highlights the challenge of accurately extracting requirements from user review, a task that remains crucial in continued development.

Therefore, a prompt was added to help GPT-4 differentiate between requirements and features. This prompt includes an example with a review and the corresponding requirement that should be extracted. The example defines what qualifies as a requirement, thereby improving the accuracy of requirement extraction. The prompt content is as follows:

Prompt: I will give you some app reviews, and can you extract the requirements from them? By requirements, I mean the reasons why a certain feature is needed. For example: I want a dark mode because I often work at night. The requirement in this sentence is that the user often works at night. The extracted requirements are in a tree structure.

Typically, training an LLM to extract requirements requires only a single example. Once the training is complete, the LLM can extract requirements from reviews of various apps. For instance, the previously mentioned example “I want a dark mode, because I often work at night” comes from Google Docs. An LLM pre-trained with this example can also extract requirements from reviews of other apps, such as LINE or YouTube. Thus, the LLM can automatically extract requirements from reviews across different apps after pre-training with a single example.

Algorithm 5 outlines the process of generating a goal model using an LLM-based approach. This algorithm takes a single goal G as input and outputs a goal sub-tree. First, an empty sub-tree is initialized with the input goal G as its root (Line 3). Additionally, a label library is created to store the labels of sub-goals and avoid redundancy (Line 4). The function LLM_GENERATE_GOAL_MODEL begins by clustering the reviews included with the current goal using an LLM (Line 6). These clusters represent sub-goals, each with a set of related reviews and a descriptive label.

For each generated cluster, this algorithm checks whether its label addresses a new requirement by comparing it against the label library (Line 8). If the label is unique, a new sub-goal is created using the cluster’s reviews and label (Line 9). This sub-goal is then added to the sub-tree as a child of the input goal, and the label library is updated with the new label (Lines 10-11). The process is applied recursively to all sub-goals of the input goal (Line 13).

After processing all goals in the sub-tree, an LLM is used to generate requirement

1. Improve User Experience and App Functionality

- requirement: General improvement of the app's interface and features.

- reason: To make the app more intuitive and efficient for users, leading to increased satisfaction and usage.

1.1 Enhance Multi-Device Support

- requirement: Seamless synchronization and operation across multiple devices.

- reason: Users often switch between devices and expect a consistent experience without loss of functionality or data.

...

Figure 6.3: Requirements and their explanations generated by GPT-4 using the example-based prompt.

descriptions for each goal based on its included reviews (Lines 16-17). This step ensures that each goal is clearly described in terms of its underlying requirements.

Figure 6.3 illustrates the requirements and their explanations generated using the example-based prompt. The input consists of the goals generated using the method introduced in the previous section and the reviews included within each goal. The output provides detailed explanations for the requirements, including an introduction to each requirement and the rationale behind it. This ensures a better understanding of the requirement and its context.

In the proposed approach, an LLM is also used to produce the labels of goals. The LDA and distance-based clustering methods utilize GPT once to generate labels for the identified goals after the goal model structure has already been established. In contrast, the LLM method requires two calls: the first is to treat goal generation as a clustering problem and produce a hierarchical relationship among goals, and the second is to create detailed requirement descriptions for each goal based on its reviews, ensuring developers gain a deeper understanding of user requirements.

Algorithm 5 LLM-based Goal Model Generation

```
1: Input: goal $G$
2: Output: $sub\_tree$                                          ▷ Return the refined goal sub-tree
3: $sub\_tree$ ← initialize_tree($G$)                          ▷ Initialize a sub-tree with $G$ as the root
4: $label\_library \leftarrow \emptyset$                        ▷ Initialize an empty label library
5: function LLM_GENERATE_GOAL_MODEL($goal$)
6:     $clusters \leftarrow LLM\_generate\_clusters(goal.reviews)$ ▷ Generate clusters based on the goal's reviews
7:     for each $cluster$ in $clusters$ do
8:         if not is_already_described($cluster.label$, $label\_library$) then ▷ Check if the cluster's label describes an unaddressed requirement
9:             $sub\_goal$ ← create_sub_goal($cluster.reviews$, $cluster.label$) ▷ Create a sub-goal from cluster
10:            add_to_tree($sub\_tree$, $goal$, $sub\_goal$)   ▷ Add sub-goal to the sub-tree
11:            $label\_library \leftarrow label\_library \cup \{cluster.label\}$ ▷ Update label library with the new requirement
12:    for each $sub\_goal$ in children($goal$, $sub\_tree$) do
13:        LLM_GENERATE_GOAL_MODEL($sub\_goal$)                ▷ Recursively generate goal models for sub-goals
14: LLM_GENERATE_GOAL_MODEL($G$)                                ▷ Generate goal model starting from $G$
15: for each $goal$ in $sub\_tree$ do
16:     $goal.description \leftarrow LLM\_generate\_req\_desc(goal.reviews)$ ▷ Generate requirement description for each goal
```

CHAPTER 7

EXPERIMENTS

7.1 Experiments: Objectives and Research Questions

This chapter shows the results of experiments using the proposed approach. The proposed approach was implemented as follows. The LDA model, together with the calculation of coherence scores, was implemented in the Python language. The Python Gensim library [58] was used for coherence score calculation. As for the distance-based method, a Python program was written to derive a dendrogram from a set of user reviews associated with a topic. Obtaining a goal model from a dendrogram was performed manually. The LLM-based method was executed using the browser-based interface of GPT-4.

To conduct the experiments, 1,000 reviews were collected from the App Store for LINE, YouTube, and Google Docs. Preprocessing for the collected reviews was performed before generating the goal models, including stop word removal and lemmatization. The stop words included those provided by NLTK [5] and additional words that do not describe requirements, such as ’app,’ ’love,’ and ’useful.’ The experiments were performed using the preprocessed reviews.

A total of five experiments were conducted. The first experiment evaluated the execution time of the three methods used in the proposed approach, namely, LDA topic modeling, the distance-based clustering method, and the LLM-based method.

The remaining experiments concern the quality of the goal models generated by the proposed approach. The experiments used the method formulated by Shimada et al. [56] as a baseline, with the goal models created manually serving as the ground truth.

The following research questions were defined to guide the experiments and analysis.
RQ1: How does the number of reviews affect the performance of LDA topic modeling and distance-based clustering?
RQ2: Does the proposed method generate goal models with higher precision and recall compared to the existing method?
RQ3: Have the requirements been successfully extracted?
RQ4: Can the goal model generated by the proposed method practically assist developers in analyzing user requirements?

Each research question corresponds to each of the remaining four types of experiments.

To answer **RQ1**, the precision and recall of the LDA topic modeling method and the distance-based clustering method were evaluated as the number of reviews varied. The result was used to identify the appropriate circumstances for applying each method.

To answer **RQ2**, the quality of the goal model generated by the proposed method was compared with that generated by the existing method in the next experiment.

RQ3 concerned the practical applicability of the proposed approach. This was answered by comparing the requirements represented by the goal models generated by the proposed method and the ground truth.

Finally, **RQ4** was addressed by checking whether the requirements extracted by the proposed approach were actually implemented later using the update log of the LINE app.

7.2 Execution Time Analysis of Methods

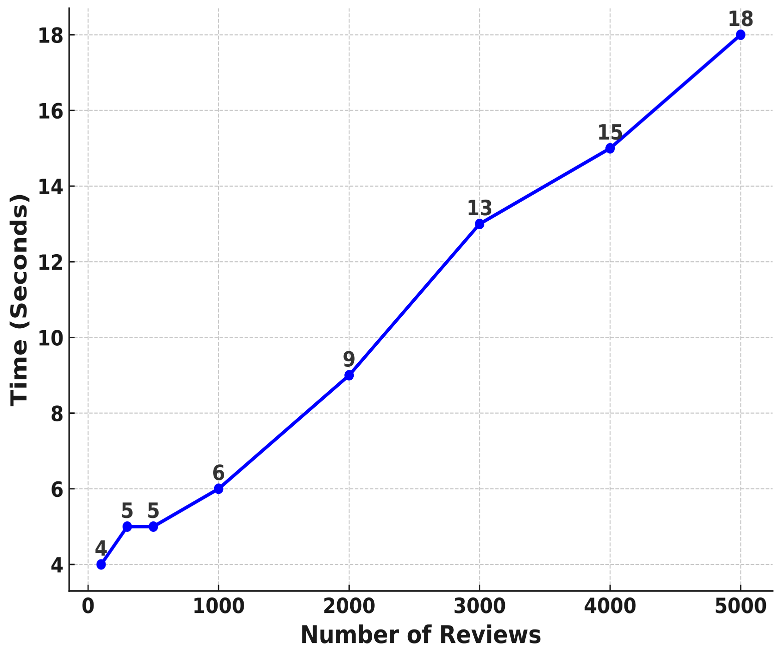


Figure 7.1: Execution time for topic generation using the LDA method.

The execution times of LDA topic modeling, distance-based clustering, and the LLM-based method were evaluated. The measurement was performed on an Apple MacBook Pro with a 3.5 GHz M2 Max processor and 32 GB of RAM. For the LDA modeling method, the execution time was tested across datasets with varying numbers of user reviews to measure the time required to generate topics. The number of topics for each LDA test was set to 10, and the time required from inputting the reviews to assigning topics to them all was recorded. As shown in Figure 7.1, the time required for LDA topic generation increased linearly with the number of reviews.

The distance-based clustering method was similarly tested using the same datasets as the LDA modeling method to measure the time required to generate the dendrogram. The results show that the dendrogram generation time increases non-linearly as the number of

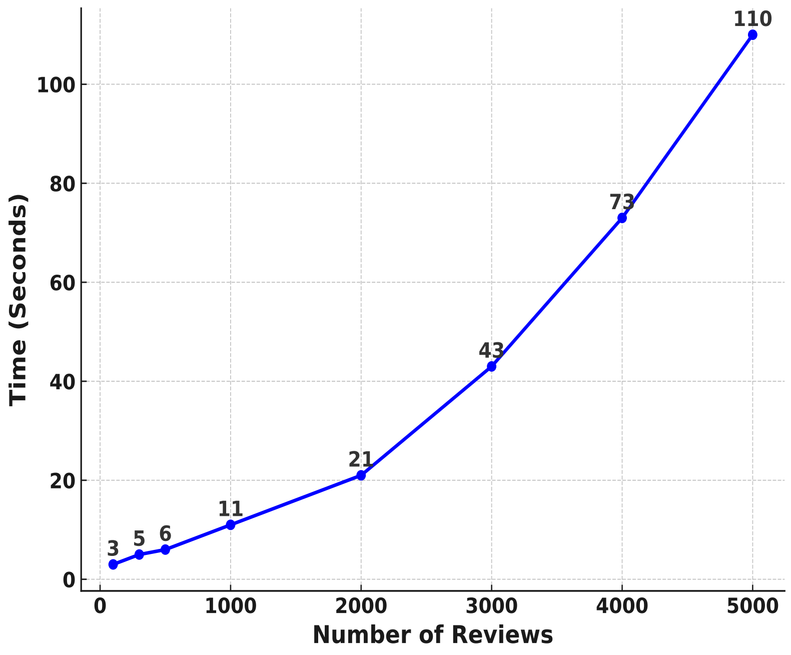


Figure 7.2: Execution time for the dendrogram generation using distance-based clustering.

reviews grows, as illustrated in Figure 7.2. The relationship suggests a significant rise in processing time with larger datasets.

The LLM-based method was assessed by measuring the time required to conduct one calculation for goal model generation. Figure 7.3 illustrates the time taken by GPT to generate a goal model with different review counts. Due to the randomness of GPT-generated content, each dataset was tested ten times, with results presented as the box plot to show the range of time required. The results indicate that the time GPT requires to generate goal models does not increase proportionally with the number of reviews. Instead, substantial variability in time was observed when generating goal models from the same dataset, which is attributed to the randomness in GPT's output. The execution time depended more on the generated content's length than the volume of reviews.

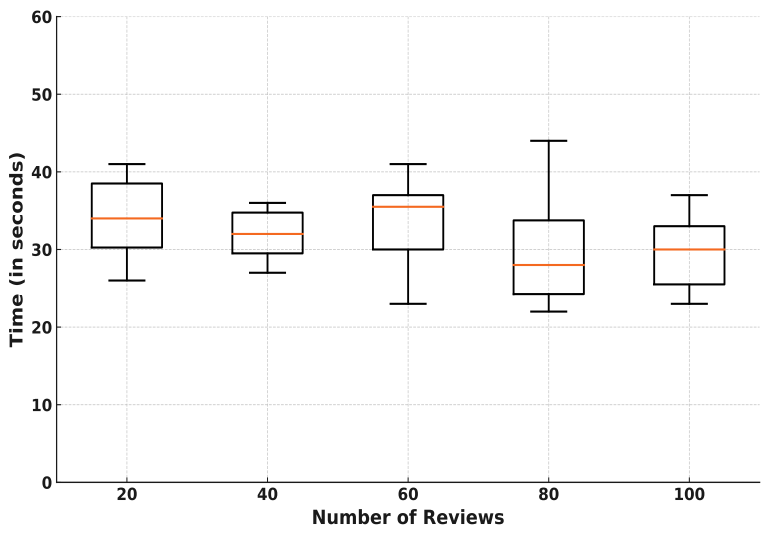


Figure 7.3: GPT method: Execution time for goal model generation.

7.3 Comparison: LDA and Clustering

7.3.1 Experimental Settings

The experiment described in this section is designed to evaluate the performance of the LDA goal model and distance-based clustering method as the number of reviews varies. The experiment tested the effectiveness of these two methods across different sample sizes, ranging from 100 to 400 reviews. The LINE app reviews were used for the experiment. The test data were constructed by selecting 400 of these reviews as follows. First, four representative requirements were selected. Then, for each of the requirements, a collection of reviews describing the requirement were manually selected. A dataset of a particular size was compiled by combining reviews from some of the four collections. Table 7.1 provides a detailed breakdown of how the reviews were allocated.

The two methods were applied to each data set. As the purpose of this experiment was to compare the basic characteristics of the LDA and distance-based clustering methods, the proposed techniques were not utilized. Specifically, 1) for the LDA method, the

Table 7.1: Number of reviews assigned to each requirement across different test rounds. Req. represents the number of reviews associated with each specific requirement.

Test Round	Req. 1	Req. 2	Req. 3	Req. 4	Total
Round 1	50	50	-	-	100
Round 2	50	50	50	-	150
Round 3	100	100	-	-	200
Round 4	100	100	50	-	250
Round 5	150	100	50	-	300
Round 6	150	100	100	-	350
Round 7	160	100	100	40	400

number of topics was set to the number of requirements (i.e., 2 to 4) a priori, resulting in as many goals as the requirements, and 2) for the distance-based clustering method, the proposed technique for aggregating intermediate goals was not applied.

The outcomes of the two methods were evaluated with respect to precision, recall, and F1-score. The measures were calculated for each of the requirements as follows: First, a goal that best describes the requirement was selected. This was done by selecting the goal with the greatest number of reviews describing that requirement. Then, all the reviews associated with that goal were subjected to the evaluation.

The evaluation was performed based on a confusion matrix, as shown in Table 7.2. TP (True Positive) indicates the reviews present in the goal that describe the correct requirement. FP (False Positive) represents reviews that are present in the goal but do not describe the requirement. FN (False Negative) signifies the reviews describing the requirement but not present in the goal. For example, if reviews 1.1, 1.2, and 1.3 describe a particular requirement but the reviews present in the selected goal are reviews 1.1 and 2.1,

Table 7.2: Confusion Matrix

		True Condition	
		Positive	Negative
Predicted	Positive	True Positive (TP)	False Positive (FP)
Condition	Negative	False Negative (FN)	True Negative (TN)

then the precision is 0.5, and the recall is 0.33. The F1-score is defined as the harmonic mean of precision and recall, providing a single metric as follows.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

7.3.2 Experimental Results

Figure 7.4 presents the F1-score, precision, and recall for the LDA and clustering method across different numbers of user reviews. The x-axis represents the number of reviews, ranging from 100 to 400, while the y-axis indicates the score values for precision, recall, and the F1-score. For the LDA method, a noticeable decrease in precision and recall is observed as the number of reviews increases from 100 to 400. This decline is reflected in the corresponding F1-scores, with a marked drop in performance particularly between 100 and 200 reviews. Despite this, the LDA method maintains a relatively stable trend after 250 reviews, with F1-scores leveling off around 65. In contrast, the clustering method shows higher precision and recall scores at lower review counts (100 to 200), but these metrics decline more sharply as the review count increases beyond 300. The clustering method's F1-scores also decrease significantly as the number of reviews approaches 400, indicating a potential decrease in clustering performance under higher review volumes. Overall, the figure illustrates that both methods experience a decline in performance with increasing review counts, but the LDA method shows a more consis-

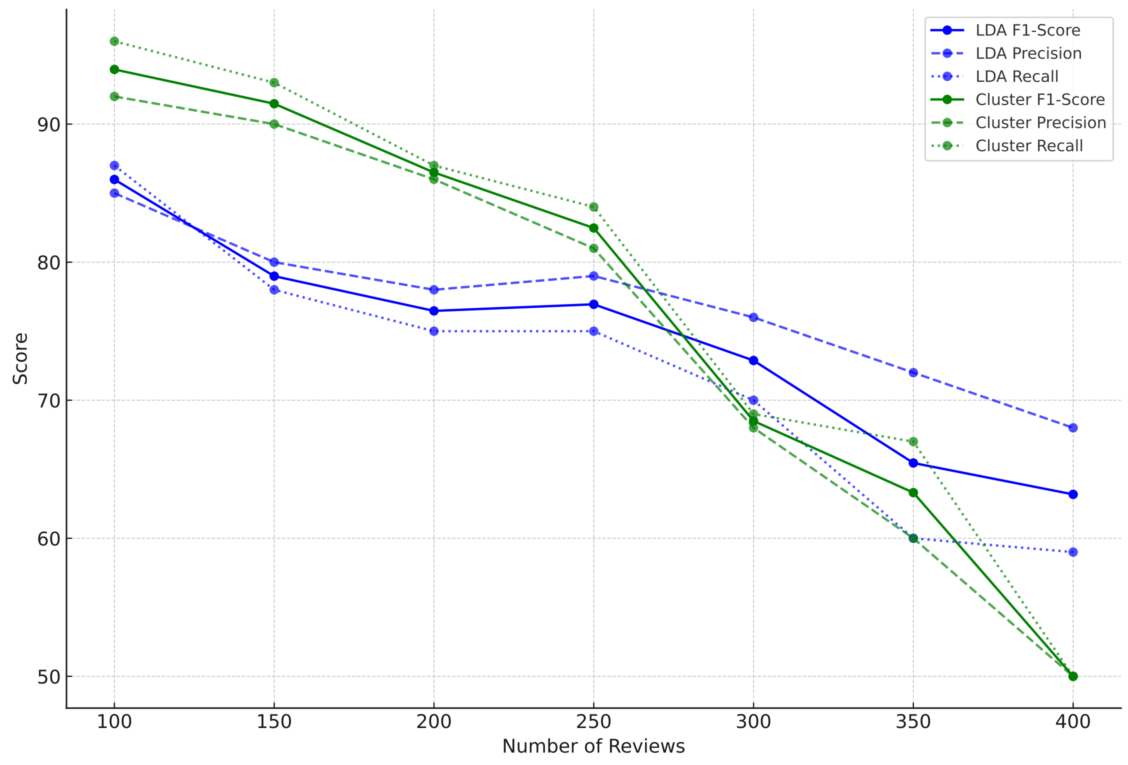


Figure 7.4: F1-Score, precision, and recall for LDA and distance-based clustering methods.

tent decline, while the clustering method exhibits a sharper drop in effectiveness at higher review counts.

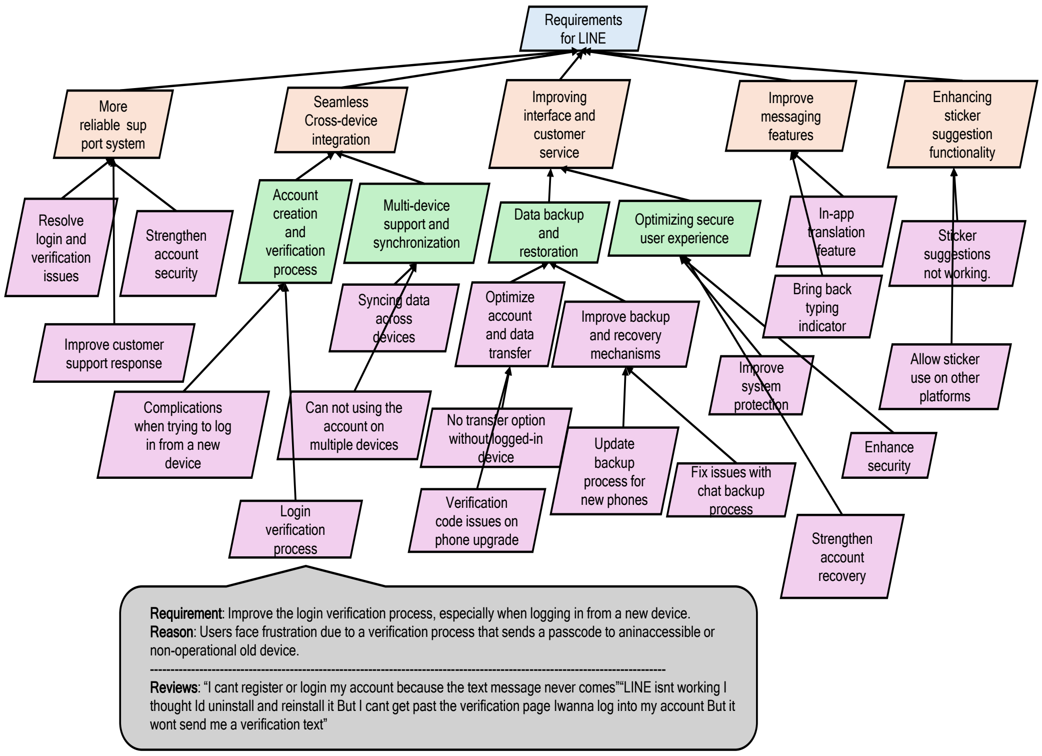
7.3.3 Answer to RQ 1

When comparing the performance of LDA and clustering methods across varying numbers of reviews, both methods showed a decrease in effectiveness as the number of reviews increased. However, the clustering method exhibits a more pronounced decline in performance, whereas the LDA method demonstrates a more gradual reduction in accuracy. Notably, both methods perform similarly to datasets containing 250 and 300 reviews. Therefore, the proposed method switches the three methods as follows. The LDA topic modeling method is employed for goals with more than 300 reviews; the distance-based clustering method is used for goals with more than 100 but fewer than 300 reviews; and the LLM method is chosen for goals with fewer than 100 reviews.

7.4 Comparison: Proposed and Existing Methods

7.4.1 Experimental Settings

This experiment evaluated the accuracy of goal model generation using the proposed method in comparison to the existing method. For the study, 3000 user reviews were collected from three different apps. The proposed method was used to generate three goal models, one for each app. The existing method was similarly applied to generate three goal models from the same set of reviews. Figures 7.5, 7.6, and 7.7 illustrate the goal models generated using the proposed method. In these models, the orange goals were generated by the LDA topic model, the green goals were generated by a distance-based clustering method, and the pink goals were generated by GPT-4. Each goal in the goal


Requirements for LINE
More reliable sup port system
Seamless Cross-device integration
Improving interface and customer service
Improve messaging features
Enhancing sticker suggestion functionality
Resolve login and verification issues
Strengthen account security
Improve customer support response
Account creation and verification process
Multi-device support and synchronization
Data backup and restoration
Optimizing secure user experience
In-app translation feature
Bring back typing indicator
Sticker suggestions not working.
Allow sticker use on other platforms
Syncing data across devices
Optimize account and data transfer
Improve backup and recovery mechanisms
Improve system protection
Enhance security
Complications when trying to log in from a new device
Can not using the account on multiple devices
No transfer option without logged-in device
Update backup process for new phones
Fix issues with chat backup process
Strengthen account recovery
Login verification process
Verification code issues on phone upgrade

Requirement: Improve the login verification process, especially when logging in from a new device.
Reason: Users face frustration due to a verification process that sends a passcode to aninaccessible or non-operational old device.

Reviews: “I cant register or login my account because the text message never comes”“LINE isnt working I thought Id uninstall and reinstall it But I cant get past the verification page Iwanna log into my account But it wont send me a verification text”

Figure 7.5: Goal model generated by the proposed method using LINE user reviews.

model generated by the proposed method was evaluated as follows: First, each goal node was manually mapped to a node in the ground truth goal model with the most similar characteristics, including their positions in the hierarchical structures and the reviews associated with them. Then, the goal was evaluated by comparing its associated reviews with those of the mapped ground truth goal, using precision, recall, and F1 measures. The goals generated by the existing method were similarly evaluated.

7.4.2 Experimental Results

Figures 7.8, 7.9, and 7.10 display the precision, recall, and F1 scores for goals generated from reviews of three different apps: LINE, GoogleDocs, and YouTube, respectively. In these figures, the horizontal axis represents precision, while the vertical axis represents recall. Overlaying the scatter plot are F1 score contours, represented by black lines indi-

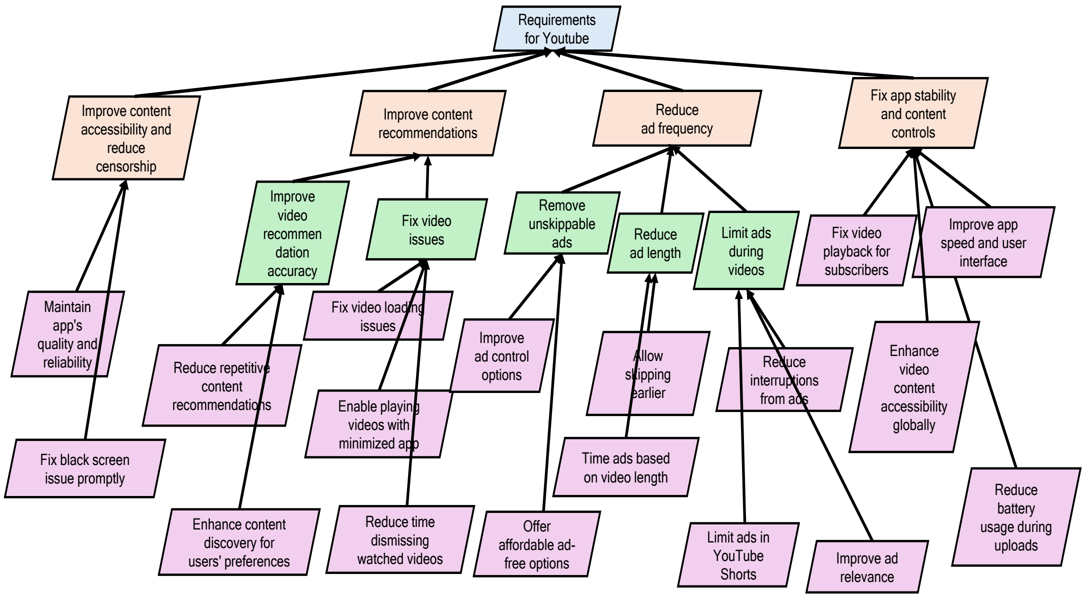

Requirements for Youtube
Improve content accessibility and reduce censorship
Improve content recommendations
Reduce ad frequency
Fix app stability and content controls
Improve video recommen dation accuracy
Fix video issues
Remove unskippable ads
Reduce ad length
Limit ads during videos
Fix video playback for subscribers
Improve app speed and user interface
Maintain app's quality and reliability
Fix video loading issues
Improve ad control options
Reduce repetitive content recommendations
Enable playing videos with minimized app
Allow skipping earlier
Reduce interruptions from ads
Enhance video content accessibility globally
Fix black screen issue promptly
Time ads based on video length
Reduce battery usage during uploads
Enhance content discovery for users' preferences
Reduce time dismissing watched videos
Offer affordable ad-free options
Limit ads in YouTube Shorts
Improve ad relevance

Figure 7.6: Goal model generated by the proposed method using user reviews of YouTube.

cating regions of constant F1 scores. These contours help visualize how the combination of Precision and Recall influences the F1 score, with higher contours closer to the top-right corner of the plot. Each dot represents a goal, with the blue dots corresponding to goals generated by the proposed method and red dots representing goals generated by the existing method. The coordinates of each dot indicate the precision and recall of the reviews included in that goal.

7.4.3 Answer to RQ 2

The results indicate that the goal model generated by the proposed method demonstrates higher accuracy, while the goal model produced by the existing method struggles to achieve high scores in both precision and recall simultaneously. The existing method employs Ward's method, which couples clusters based on distances. This method can lead to issues where a cluster with very few reviews is combined with one that has a significantly larger number of reviews. Consequently, one cluster may exhibit high precision and low accuracy, while the other shows high recall and low precision.

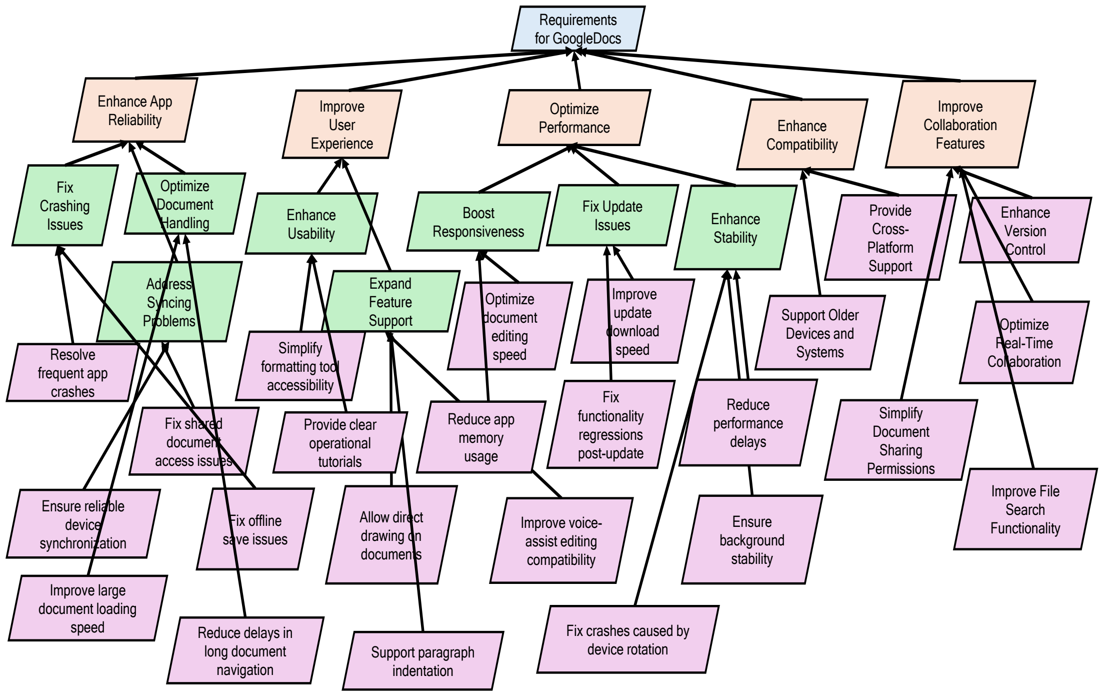

Requirements for GoogleDocs
Enhance App Reliability
Improve User Experience
Optimize Performance
Enhance Compatibility
Improve Collaboration Features
Fix Crashing Issues
Optimize Document Handling
Enhance Usability
Boost Responsiveness
Fix Update Issues
Enhance Stability
Provide Cross-Platform Support
Enhance Version Control
Address Syncing Problems
Expand Feature Support
Optimize document editing speed
Improve update download speed
Support Older Devices and Systems
Optimize Real-Time Collaboration
Resolve frequent app crashes
Simplify formatting tool accessibility
Fix shared document access issues
Provide clear operational tutorials
Reduce app memory usage
Fix functionality regressions post-update
Reduce performance delays
Simplify Document Sharing Permissions
Improve File Search Functionality
Ensure reliable device synchronization
Fix offline save issues
Allow direct drawing on documents
Improve voice-assist editing compatibility
Ensure background stability
Improve large document loading speed
Reduce delays in long document navigation
Support paragraph indentation
Fix crashes caused by device rotation

Figure 7.7: Goal model generated by the proposed method using user reviews of Google-Docs.

The accuracy and performance of the model are not solely dependent on the clustering algorithm but are also influenced by the content of the reviews themselves. For example, on platforms like YouTube, user reviews often revolve around a limited set of topics, with a repetitive use of language. In such cases, the existing method tends to perform better, as with the increased frequency of keywords, it becomes easier to cluster reviews that describe similar requirements, thereby improving accuracy. However, even for the YouTube app, the proposed method showed better performance.

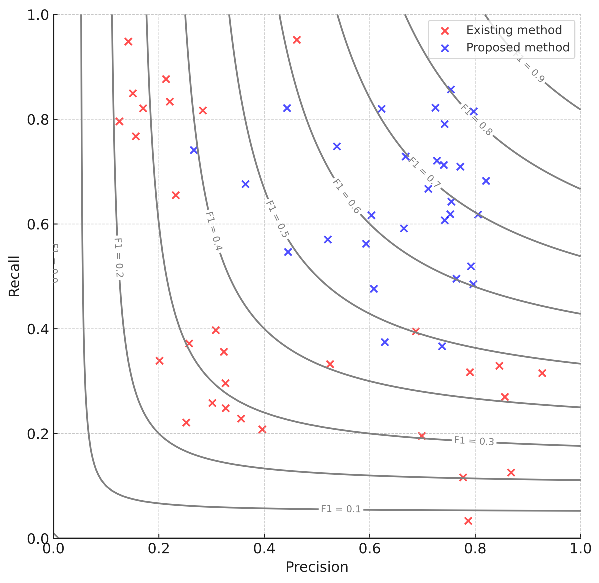


Figure 7.8: Precision, recall, and F1 scores for goals generated from LINE reviews.

7.5 Effectiveness of Requirements Extraction

7.5.1 Experimental Settings

This experiment evaluated the proposed method's ability to extract requirements from reviews. The following steps were applied to a set of 1000 reviews for each of the three apps. The first step involved gathering all the requirements from the ground truth and compiling them into a list. Although the requirements derived had a wide granularity spectrum, only the finest ones were selected. Each of these requirements corresponded to each leaf goal. Figure 7.11 shows a portion of the requirements extracted from the ground truth. Next, the requirements included in the goals generated by the proposed method were collected and also compiled into a list. The labels or descriptions generated by GPT-4 for the goals were used to extract the requirements. Finally, a comparison was made between the two lists to calculate the coverage rate of the requirements. Here, the coverage rate is the ratio of requirements derived from the generated goal model to the

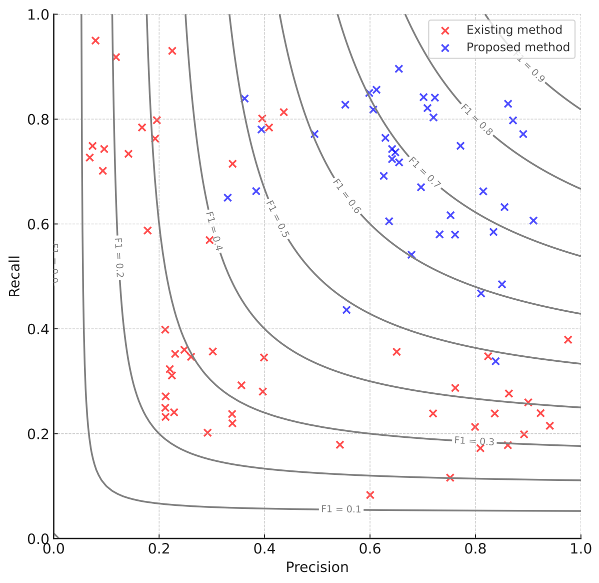


Figure 7.9: Precision, recall, and F1 scores for goals generated from GoogleDocs reviews.

ground-truth requirements.

7.5.2 Experimental Results

Table 7.3 summarizes the results of this experiment. The table presents the number of requirements extracted by the proposed method, the ground-truth requirements, and the corresponding coverage rates for three apps: LINE, Google Docs, and YouTube. For LINE, the proposed method extracted 15 requirements out of 16 ground-truth requirements, resulting in a coverage rate of 0.94. Google Docs achieved a slightly lower coverage rate of 0.86, with 18 out of 21 ground-truth requirements identified. In contrast, YouTube had only 10 requirements in both the extracted and ground-truth sets, yielding a coverage rate of 1. This lower number of requirements for YouTube is due to the context of its reviews, which predominantly focus on a single issue—advertisements—making the requirements fewer and easier to capture.

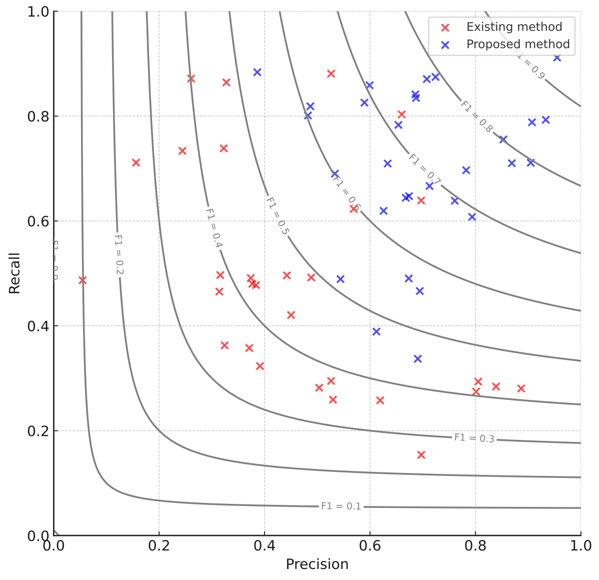


Figure 7.10: Precision, recall, and F1 scores for goals generated from YouTube reviews.

Table 7.3: The numbers of extracted and ground-truth requirements.

	Proposed method	**Ground truth**	**Coverage rate**
LINE	15	16	0.94
Google Docs	18	21	0.86
YouTube	10	10	1

7.5.3 Answer to RQ 3

The generated goal model successfully covered about 90% of the requirements. Based on the results, it can be concluded that the goal models generated by the proposed method can capture most of the requirements.

1. Login Requirement: Enhanced Verification Process Ensures that the login system includes a robust verification process to secure user access.
2. Login Requirement: Removal of Regional Restrictions Implement a feature that allows users to log in from any region without geographical limitations.
3. Login Requirement: Support for Multiple Devices Enable users to log in and access their accounts simultaneously on multiple devices.
4. Login Requirement: Email Integration for Login Notifications Provide users with the ability to receive login notifications and support via email to enhance security and user experience.
5. Notification Requirement: Customizable Reminder Settings Allow users to customize the method and frequency of reminders according to their preferences.
6. Notification Requirement: Continuous Reminders Without Interruption Ensure that reminder notifications continue without interruption until acknowledged by the user.

...

Figure 7.11: List of extracted requirements.

7.6 The Proposed Method: Practical Utility

7.6.1 Experimental Settings

To evaluate the practical utility of the proposed method, a comparison was conducted between the goal model generated from user reviews for LINE and the app's actual update logs. By examining the correspondence between the requirements described in the goal model and the features or improvements implemented in the app updates, we can assess whether the requirements in the generated model reflect those that developers aim to fulfill. If such alignment is observed, it indicates that the goal model is consistent with the developers' thought process, demonstrating its practical relevance to application development. Furthermore, this suggests that in future app updates, the automatically generated goal model could provide guidance for developers, serving as a tool to help the development process.

The reviews used to generate the goal model were collected in 2020, providing a snapshot of user requirements from that period. Meanwhile, a comprehensive set of 173 update records from LINE, covering a period from January 2020 to October 2024, was collected. These records were then filtered to exclude updates that lacked requirement-specific information, such as "We've fixed some bugs and improved features to make LINE even better.", leaving 50 updates that contained clear references to modifications relevant to user requirements.

7.6.2 Experimental Results

Figure 7.12 illustrates the update heatmap for LINE, where goals marked in red indicate frequent updates addressing the requirements described by these goals, while those marked in blue represent requirements that received less attention from developers. Among the 29 goals in the goal model, 21 were mentioned at least once in the update logs, high-

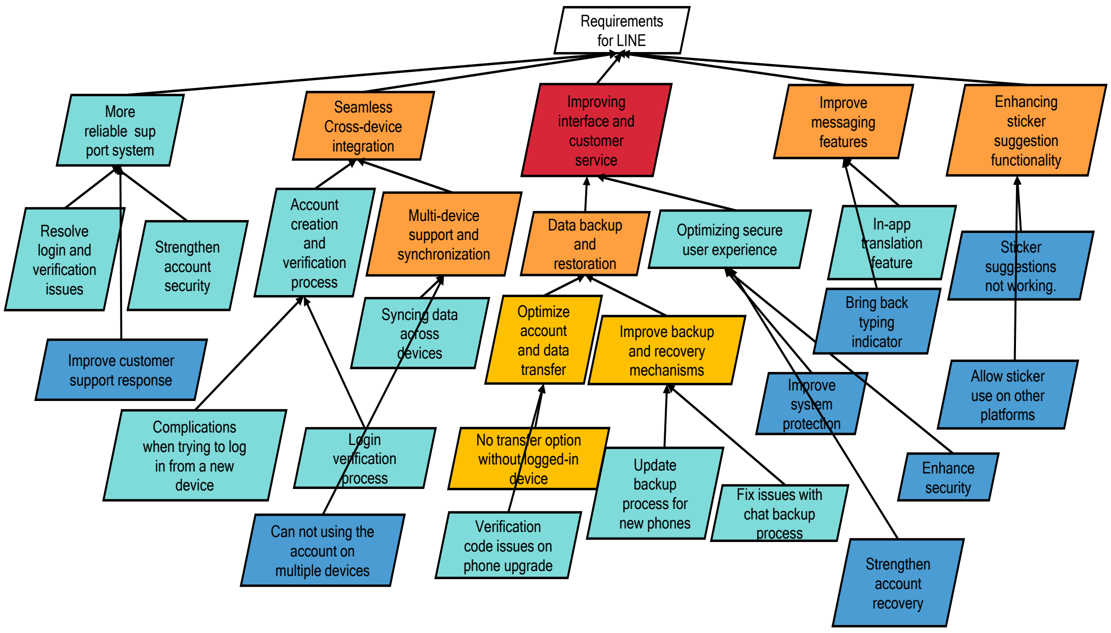

Requirements for LINE
More reliable sup port system
Seamless Cross-device integration
Improving interface and customer service
Improve messaging features
Enhancing sticker suggestion functionality
Resolve login and verification issues
Strengthen account security
Improve customer support response
Account creation and verification process
Multi-device support and synchronization
Data backup and restoration
Optimizing secure user experience
In-app translation feature
Sticker suggestions not working.
Bring back typing indicator
Syncing data across devices
Optimize account and data transfer
Improve backup and recovery mechanisms
Improve system protection
Allow sticker use on other platforms
Complications when trying to log in from a new device
Login verification process
No transfer option without logged-in device
Update backup process for new phones
Fix issues with chat backup process
Enhance security
Can not using the account on multiple devices
Verification code issues on phone upgrade
Strengthen account recovery

Figure 7.12: Requirement update frequency in LINE.

lighting that most goals garnered focus from developers. Of 50 update logs, 46 referenced requirements were described in the goal model, demonstrating that the requirements extracted through the goal model had high coverage.

As shown in Figure 7.12, the requirement "Improving interface and customer service" emerged as the requirement that developers focused on the most, with its sub-goals also being frequently mentioned in updates, particularly those related to data backup and transfer, such as the update log item "You can now set up regular automatic backups of your chat history," which corresponds to the goal "Improve backup and recovery mechanisms." This indicates significant attention to this aspect. Meanwhile, some goals were mentioned in updates, but their sub-goals, such as specific suggestions regarding stickers, were not implemented. In this case, the issues regarding stickers were addressed differently from the one mentioned in the user reviews.

7.6.3 Answer to RQ 4

The goal model generated from user reviews shows a high overlap between the identified requirements and the subsequent app update logs. This indicates that the requirements captured by the goal model align closely with the areas developers aim to improve. Consequently, the generated goal model proved valuable both in identifying user requirements and guiding developers toward improvements.

Additionally, by analyzing the "popularity" of different goals within the goal model, it becomes evident which aspects of the app developers prioritize. This insight enables development teams to focus on the most critical areas, optimizing resource allocation and enhancing user satisfaction.

CHAPTER 8

DISCUSSION

Structurally, the goal models generated by the proposed method are similar to the ground truth. The goal models generated by the existing method have greater depth but limited width. This is due to Ward's method, which restricts the number of sub-goals, affecting the width of the generated goal models. In contrast, the proposed method generates goal models with greater width and less depth, similar to the structure of the ground truth. The nature of user reviews influences this structural characteristic. Most reviewers use brief statements to express their requirements rather than providing systematic explanations, producing goal models with little depth. Additionally, independent reviews usually have no clear connection, and their app usage contexts differ, leading to diverse requirements that cover various aspects of the app, contributing to the generated goal models' greater width. The LDA topic model used in the proposed method effectively captures this diversity of requirements by categorizing reviews describing different requirements into separate topics, which are treated as sub-goals of the same goal. Given the diversity of requirements, the number of sub-goals is higher, resulting in a goal model with greater width.

Regarding the generated models' accuracy, the goal model generated using the proposed method demonstrates superior performance in precision and recall when compared to the existing method. In the goal model generated by the existing method, the low clustering accuracy causes reviews that describe the same requirement to be scattered across multiple goals. When a large number of reviews pertain to a particular requirement, these dispersed reviews may still dominate the content of a goal. This situation makes it challenging for less frequently mentioned requirements to be adequately captured by a goal. As such, multiple goals ultimately describe the same requirement in models generated by the existing method, which significantly diminishes the goal model's overall accuracy.

The existing method's unsatisfactory performance is primarily attributed to the inherent complexities of user reviews, which are diverse, covering a wide array of topics such as product functionality, user experience, and customer service. This necessitates language models that can handle diverse topics and vocabularies. Reviews are typically unstructured, lacking a consistent format or template, which means language models must decipher information and intentions in these unformatted texts. Furthermore, reviews often feature colloquial language, including slang, abbreviations, and emoticons, all of which require language models to adeptly interpret casual expressions and informal language. Emotions like satisfaction, frustration, or excitement are commonly expressed in reviews, making it essential for language models to understand these emotional cues to evaluate user attitudes and prioritize their requirements. Additionally, user reviews may contain spelling errors, grammar mistakes, or unconventional language, challenging language models to either correct these issues or comprehend the intended meaning through context. Finally, the brevity of some reviews, which can succinctly convey complex needs or issues with just a few words, demands that language models make precise inferences from minimal data. These factors contribute to the challenges faced by language models in effectively extracting requirements, causing the existing method's perceived inadequacy.

Additionally, the existing method relies on the distance-based clustering method, namely, Ward's method, which converts reviews into vectors and measures the similarity of reviews through vector distance. However, as emphasized by Peter et al. [14], vector similarity does not always indicate similarity to the requirements expressed in reviews. Even if reviews share the same keywords, it does not mean they share the same requirements. For instance, consider the following:

- Thanks to the app's automatic translation feature, I can communicate with people from other countries.
- Why am I unable to log in when I travel to other countries?

Despite both sentences referencing "other countries" and being clustered together, they represent distinct requirements. This situation can result in decreased clustering accuracy and pose challenges in generating precise goal models. In comparison, GPT-4 can accurately identify the requirements within user reviews. Thanks to its extensive pre-training on various text datasets, including web pages, books, and articles, it has been exposed to a vast array of language patterns, styles, and topics. This exposure enables it to understand complex content, including slang, domain-specific terminology, and the unconventional expressions often found in user reviews. GPT-4 interprets the underlying intentions and information from complex user reviews, allowing it to generate precise goal models even when faced with incomplete or non-standard user reviews.

The proposed method is designed to extract requirements from reviews. Other types of data, such as requirement documents, have different characteristics from reviews, making the goal models generated using this proposed method potentially less effective. Requirement documents are typically fewer in number but describe more complex and systematic requirements. This makes methods designed for handling large volumes of reviews, such as LDA or distance-based clustering, unsuitable. Additionally, the LLM-based method is constrained by token limitations, making it challenging to fully describe all requirements. Therefore, using non-review data to generate goal models often fails to accurately represent the requirements contained within.

Although this study has offered some new insights, the limitations of the proposed methods must be acknowledged. One of the limitations is the difficulty in extracting all requirements. Some reviews describe multiple requirements. In such cases, the proposed method can extract only one requirement, leaving the others unextracted. These additional requirements can only be extracted from other reviews. To address this issue, future considerations include dividing reviews into sentences during preprocessing and using sentences as input. By splitting reviews into sentences, the requirements can be extracted from each sentence more accurately, thereby enhancing the accuracy of requirement extraction. This approach aims to handle reviews that contain multiple requirements appropriately.

The goal model generated by the proposed method does not represent a full range of logical relationships between goals. For example, the sub-goals under the same parent goal are only connected by AND relationships, meaning the parent goal is considered fulfilled only when all sub-goals are completed. However, in goal models used in actual development, the relationships between sub-goals include not just AND but also OR. Development specifications systematically describe the relationships between goals, clearly defining which are mandatory and which are optional. In contrast, the goals in the model generated by the proposed method are derived from user reviews. Individual users each believe their particular requirements should be included, making it difficult to determine the relationships between goals.

Issues common to LLMs emerge when using GPT-4 to generate goal models. For instance, due to LLM's varying randomness, caused by the temperature settings and random sampling strategies, the results of classifying user reviews may differ significantly. Each processing of the same reviews yields different goals in terms of quantity and content. Sometimes, the LLM categorizes reviews based on user sentiment rather than actual requirements, which affects the accuracy of the generated models. To obtain precise goal models, it often requires multiple generations and manual verification thus, complicating automation.

Token limitation is a common constraint in LLMs due to their architecture and computational demands, restricting the number of tokens (words, punctuation marks, or subword units) the model can handle simultaneously. This limitation affects the length of input and output; if the combined number of tokens exceeds the model's maximum capacity, the input will be truncated, or the output may be incomplete. For GPT-4, processing more than 100 reviews at a time is generally unfeasible. To address this issue, LDA and clustering methods are initially used to categorize reviews into topics and clusters, reducing the burden on GPT-4 by limiting the number of reviews processed together.

Given these limitations, further improvements and adaptations are necessary to realize the potential fully of using LLMs in review analysis. Beyond improving the LLMs or how they are applied, refining the goal model interactively is a realistic solution. For this, an interactive goal modeling tool [45] and an editor [60] have been developed. Combining LLM-based elicitation and interactive tools can reduce the burden of analysis and improve the goal model's precision.

I now address the threats to the validity of the evaluation results. The ground truth for this study was generated by a single researcher manually checking all the reviews and extracting the requirements. Since the reviewers are not professionals and their descriptions of requirements may be imprecise, different individuals may have extracted slightly different requirements. Consequently, the created goal model, as the ground truth, may differ if constructed by another researcher. Furthermore, due to the LLM method's inherent randomness, there are limitations in the proposed method's reproducibility.

CHAPTER 9

RELATED WORK

Requirements engineering assumes a pivotal role in the domain of software development, as underscored by Bourque et al. [7]. Many studies, e.g., [2, 4, 11, 14, 19, 20, 35] have leveraged natural language processing and sentiment analysis techniques to extract user requirements from reviews. However, they have not focused on the relationships between requirements. For instance, Licorish et al. focus on identifying key attributes contributing to the success of an application, rather than analyzing how these attributes are interrelated [35]. Gao et al.'s two studies emphasize identifying patterns and trends in user reviews and assessing the quality of reviews to extract requirements but do not delve into the hierarchical or interdependent relationships among requirements [19, 20]. Dkabrowski et al. highlight the extraction of information from user feedback to support software maintenance and evolution but do not explore how requirements interact to drive system improvements [11]. Astegher et al. propose a framework that integrates user feedback into requirements engineering, primarily focusing on requirement elicitation and prioritization, without addressing the logical relationships or the construction of sub-goals [2]. Bettenburg et al. define the characteristics of good bug reports, yet these characteristics are not directly linked to the upstream or downstream relationships of requirements [4]. Devine et al.'s social media mining framework provides a new perspective for requirement elicitation but does not consider how requirements collectively form a coherent goal structure [14]. Therefore, while these studies concentrate on requirement extraction and classification, they lack systematic modeling and analysis of the relationships between requirements.

Studies [23, 24, 29, 46, 52] employed similar clustering techniques, but their research subjects were not user reviews. Instead, they focused on software artifacts, such as code snippets, design documents, and development-related communication. For instance, Iqbal et al. focused on clustering software artifacts to aid in software maintenance and reuse [29]. Guzman et al. explored emotions and useful information in software repositories, such as developer communications in issues and pull requests [23, 24]. Robeer et al. applied clustering techniques to software requirements documents [52], while Ni et al. introduced a tool for clustering software artifacts to improve code reuse [46]. Unlike software artifacts dealt with in these studies, which aimed to enhance software development processes, user reviews often contain informal, varied, and unstructured content, posing unique challenges for clustering and requirement extraction.

Many researchers have evaluated existing review analysis methods. Maalej and Nabil [39] present a spectrum of techniques for categorizing application reviews into four distinct types. These techniques have been methodically evaluated through experiments to ascertain their accuracy. Notably, the Naive Bayes classifier emerged as the most effective technique among these machine learning methods. In a related fashion, Dkabrowski et al. [12] compare three opinion mining methods: GuMa [25], SAFE [31], and ReUS [17]. They conducted two empirical studies to evaluate the performance of these methods. The first study focuses on opinion-mining methods aimed at extracting features and identifying associated sentiments from app reviews, while the second concentrates on searching for user feedback related to specific features. The results show that the actual effectiveness of these methods was lower than initially reported, raising concerns about their practical use. It must be emphasized that the aforementioned studies primarily concern review analysis, and not goal model identification.

Notably, several researchers believe that LLMs could potentially revolutionize existing software development practices. This has led to numerous methods being proposed that leverage LLMs for software modeling. Nakagawa and Honiden [44] introduce a semi-automated process for goal model generation that employs generative AI founded on the MAPEK loop mechanism. Their two case studies demonstrate that this process, based on the MAPEK loop mechanism, is efficacious in goal model construction without omitting any goal descriptions. Additionally, Cámara et al. [8] comprehensively investigate ChatGPT's performance and utility in modeling tasks, while simultaneously identifying its principal limitations. Their research findings underscore that the current iteration of ChatGPT exhibits limited efficacy in software modeling, especially when compared to its code-generation capabilities. It exhibits variegated syntax and semantic defects, lacks response consistency, and faces scalability challenges. Ding and Ito [16] introduce a Self-Agreement framework, where the model generates multiple opinions. They employ another model to evaluate the consistency among these opinions, ultimately selecting the candidate with the highest agreement score. This method fine-tunes a pre-trained LLM using a dataset of question-opinion-agreement pairs, demonstrating that the model can effectively identify and achieve agreement among diverse opinions without relying on human-annotated data. Their experimental results show that the fine-tuned model, despite having significantly fewer parameters than existing large models, achieves comparable performance, highlighting the effectiveness and innovation of this approach in automating the discovery of opinion agreement. Their research focuses on finding consensus among diverse opinions, whereas this dissertation centers on analyzing the consistency of reviews and generating goal models.

CHAPTER 10

CONCLUSION

10.1 Conclusion

This dissertation presents a method for generating goal models using user reviews. The proposed method comprises three components: the LDA topic model, the distance-based clustering method, and the large language model (LLM) method. Depending on the volume of user reviews, the method dynamically selects one of these components to generate the goal model.

Since LDA is suited for processing large volumes of reviews, the proposed approach uses it to generate the root goal and its close subgoals. Since LDA's precision may decline as the number of reviews increases, the distance-based clustering method is applied next. When dealing with even fewer reviews, the LLM-based method is utilized, enabling a more nuanced analysis of reviews.

The experimental results demonstrated that the proposed method improves accuracy compared to the existing method. Additionally, it effectively extracts requirements from a large number of reviews and visualizes these requirements by generating goal models. This is achieved by dynamically selecting the three methods, as shown in the experimental results.

The proposed method entails using an LLM-based method for obtaining goal descriptions from user reviews. The experimental results indicate that the method can produce goal descriptions that are easy to understand by effectively extracting requirements from user reviews.

10.2 Future Work

There are four potential future research directions for improving the proposed method; One and two focus on extending the method's capabilities, thus enhancing its functionality while three and four emphasize exploring broader contexts for applying the method, aiming to expand its utility in diverse scenarios.

Extending the proposed approach to capture more complex goal relationships: The proposed method identifies basic goal relationships but does not comprehensively address complex interactions such as dependencies, conflicts, and logical connections (e.g., and/or relationships). To enhance the expressiveness and practicality of the goal models, future research could explore the use of LLM techniques or customized prompts to identify these relationships. By detecting and representing these elements, the resulting goal models could better reflect requirements and provide developers with a more robust framework for decision-making and implementation.

Incorporating sentiment analysis to prioritize goals based on review ratings and the number of reviewers: Future research could enhance the proposed method by integrating sentiment analysis to prioritize goals in the generated goal model. By analyzing the review scores and considering the number of reviewers, priority levels could be assigned to goals to reflect user sentiment and requirements. This prioritization would help developers focus on the most critical or widely requested features, improving the goal model's practicality.

Extending the proposed approach to analyze requirement documents for automatic goal model generation: The current method focuses on generating goal models from user reviews, representing user requirements. Future research could extend this approach to include analyzing requirement documents and enabling automatic goal model generation from more formalized inputs. This extension would increase the method's flexibility and applicability across various stages of the software development lifecycle, from initial requirements gathering to iterative refinement.

Comparing goal models of similar apps to identify factors contributing to their popularity: Future research could involve generating and comparing goal models for similar apps to analyze the factors contributing to their popularity. This approach could reveal insights into user preferences and competitive advantages by examining the differences in identified goals, features, and priorities. Such comparisons could help developers understand which features or goals resonate more with users, providing valuable guidance for strategic improvements and feature prioritization.

BIBLIOGRAPHY

[1] A. Abdelrazek, Y. Eid, E. Gawish, W. Medhat, and A. Hassan. Topic modeling algorithms and applications: A survey. *Information Systems*, 112:102131, 2023.

[2] M. Astegher, P. Busetta, A. Perini, and A. Susi. Requirements for online user feedback management in re tasks. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 336–336. IEEE, 2021.

[3] A. Begel and T. Zimmermann. Analyze this! 145 questions for data scientists in software engineering. In *Proc. of the 36th International Conference on Software Engineering*, pages 12–23, 2014.

[4] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 308–318. ACM, 2008.

[5] S. Bird, E. Loper, and E. Klein. *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[7] P. Bourque, R. Dupuis, A. Abran, J. W. Moore, and L. Tripp. The guide to the software engineering body of knowledge. *IEEE software*, 16(6):35–44, 1999.

[8] J. Cámara, J. Troya, L. Burgueño, and A. Vallecillo. On the assessment of generative ai in modeling tasks: an experience report with chatgpt and uml. *Software and Systems Modeling*, pages 1–13, 2023.

[9] L. Ceci. Number of reviews posted on the apple app store from users in the united states from 1st quarter 2021 to 2nd quarter 2022, by category. `https://www.statista.com/statistics/1334065/us-apple-app-store-app-reviews-by-category/`, 2023.

[10] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang. Ar-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, pages 767–778. ACM, 2014.

[11] J. Dąbrowski, E. Letier, A. Perini, and A. Susi. Mining user feedback for software engineering: Use cases and reference architecture. In *2022 IEEE 30th International Requirements Engineering Conference (RE)*, pages 114–126. IEEE, 2022.

[12] J. Dąbrowski, E. Letier, A. Perini, and A. Susi. Mining and searching app reviews for requirements engineering: Evaluation and replication studies. *Information Systems*, page 102181, 2023.

[13] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, Apr. 1993.

[14] P. Devine, J. Tizard, H. Wang, Y. S. Koh, and K. Blincoe. What's inside a cluster of software user feedback: A study of characterisation methods. In *2022 IEEE 30th International Requirements Engineering Conference (RE)*, pages 189–200. IEEE, 2022.

[15] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall. What would users change in my app? summarizing app re-

views for recommending software changes. In *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering*, pages 499–510, 2016.

[16] S. Ding and T. Ito. Self-agreement: A framework for fine-tuning language models to find agreement among diverse opinions. *arXiv preprint arXiv:2305.11460*, 2023.

[17] M. Dragoni, M. Federici, and A. Rexha. An unsupervised aspect extraction strategy for monitoring real-time reviews stream. *Information processing & management*, 56(3):1103–1118, 2019.

[18] M. Fowler. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.

[19] C. Gao, J. Zeng, M. R. Lyu, and I. King. Online app review analysis for identifying emerging issues. In *Proceedings of the 40th International Conference on Software Engineering*, pages 48–58, 2018.

[20] C. Gao, J. Zeng, Z. Wen, D. Lo, X. Xia, I. King, and M. R. Lyu. Emerging app issue identification via online joint sentiment-topic tracing. *IEEE Transactions on Software Engineering*, 48(8):3025–3043, 2021.

[21] N. Genc-Nayebi and A. Abran. A systematic literature review: Opinion mining studies from mobile app store user reviews. *Journal of Systems and Software*, 125:207–219, 2017.

[22] X. Gu and S. Kim. "What parts of your apps are loved by users?"(T). In *Proc. of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 760–770. IEEE, 2015.

[23] E. Guzman, R. Alkadhi, and N. Seyff. A needle in a haystack: What do twitter users say about software? In *2016 IEEE 24th international requirements engineering conference (RE)*, pages 96–105. IEEE, 2016.

[24] E. Guzman, M. Ibrahim, and M. Glinz. A little bird told me: Mining tweets for requirements and software evolution. In *2017 IEEE 25th International requirements engineering conference (RE)*, pages 11–20. IEEE, 2017.

[25] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*, pages 153–162, 2014.

[26] H. F. Hofmann and F. Lehner. Requirements engineering as a success factor in software projects. *IEEE software*, 18(4):58–66, 2001.

[27] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

[28] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004.

[29] T. Iqbal, M. Khan, K. Taveter, and N. Seyff. Mining reddit as a new source for software requirements. In *2021 IEEE 29th international requirements engineering conference (RE)*, pages 128–138. IEEE, 2021.

[30] H. Jiang, J. Zhang, X. Li, Z. Ren, D. Lo, X. Wu, and Z. Luo. Recommending new features from mobile app descriptions. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 28(4):1–29, 2019.

[31] T. Johann, C. Stanik, W. Maalej, et al. SAFE: A simple approach for feature extraction from app descriptions and app reviews. In *Proc. of the 2017 IEEE 25th international requirements engineering conference (RE)*, pages 21–30. IEEE, 2017.

[32] H. Kaiya, H. Horai, and M. Saeki. Agora: Attributed goal-oriented requirements analysis method. In *Proceedings IEEE joint international conference on requirements engineering*, pages 13–22. Ieee, 2002.

[33] S. Kutabish, A. M. Soares, and B. Casais. The influence of online ratings and reviews in consumer buying behavior: a systematic literature review. In *International Conference on Digital Economy*, pages 113–136. Springer, 2023.

[34] W. Li and A. McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584, 2006.

[35] S. A. Licorish, B. T. R. Savarimuthu, and S. Keertipati. Attributes that predict which features to fix: Lessons for app store mining. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pages 108–117, 2017.

[36] S. A. Licorish, A. Tahir, M. F. Bosu, and S. G. MacDonell. On satisfying the android os community: User feedback still central to developers' portfolios. In *2015 24th Australasian Software Engineering Conference*, pages 78–87. IEEE, 2015.

[37] S. Ma, S. Wang, D. Lo, R. H. Deng, and C. Sun. Active semi-supervised approach for checking app behavior against its description. In *Proc. of the 2015 IEEE 39Th annual computer software and applications conference*, volume 2, pages 179–184. IEEE, 2015.

[38] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik. On the automatic classification of app reviews. *Requirements Engineering*, 21(3):311–331, Sep 2016.

[39] W. Maalej and H. Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *2015 IEEE 23rd international requirements engineering conference (RE)*, pages 116–125. IEEE, 2015.

[40] W. Maalej and D. Pagano. On the socialness of software. In *Proc. of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 864–871. IEEE, 2011.

[41] H. Malik, E. M. Shakshuki, and W.-S. Yoo. Comparing mobile apps by identifying 'hot'features. *Future Generation Computer Systems*, 107:659–669, 2020.

[42] S. McIlroy, N. Ali, H. Khalid, and A. E. Hassan. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering*, 21:1067–1106, 2016.

[43] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on software engineering*, 18(6):483–497, 1992.

[44] H. Nakagawa and S. Honiden. MAPE-K loop-based goal model generation using generative ai. In *13th International Workshop on Model-Driven Requirements Engineering (MoDRE)*. IEEE, 2023.

[45] H. Nakagawa, H. Shimada, and T. Tsuchiya. Interactive goal model construction based on a flow of questions. *IEICE Transactions on Information and Systems*, E103.D(6):1309–1318, 2020.

[46] A. Ni, D. Ramos, A. Z. Yang, I. Lynce, V. Manquinho, R. Martins, and C. Le Goues. Soar: a synthesis approach for data science api refactoring. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 112–124. IEEE, 2021.

[47] M. Oriol, M. Stade, F. Fotrousi, S. Nadal, J. Varga, N. Seyff, A. Abello, X. Franch, J. Marco, and O. Schmidt. Fame: supporting continuous requirements elicitation by combining user feedback and monitoring. In *Proc. of the 2018 ieee 26th international requirements engineering conference (re)*, pages 217–227. IEEE, 2018.

[48] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *2013 21st IEEE international requirements engineering conference (RE)*, pages 125–134. IEEE, 2013.

[49] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, 61(2):217–235, 2000.

[50] R. Rehurek and P. Sojka. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2), 2011.

[51] S. Ren, H. Nakagawa, and T. Tsuchiya. Goal model structuring based on semantic correlation of user reviews. *Intelligent Decision Technologies*, 16(Preprint):1–12, 2022.

[52] M. Robeer, G. Lucassen, J. M. E. Van Der Werf, F. Dalpiaz, and S. Brinkkemper. Automated extraction of conceptual models from user stories via nlp. In *2016 IEEE 24th international requirements engineering conference (RE)*, pages 196–205. IEEE, 2016.

[53] M. Röder, A. Both, and A. Hinneburg. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408, 2015.

[54] S. Scalabrino, G. Bavota, B. Russo, M. Di Penta, and R. Oliveto. Listening to the crowd for the release planning of mobile apps. *IEEE Transactions on Software Engineering*, 45(1):68–86, 2017.

[55] N. Seyff, F. Graf, and N. Maiden. Using mobile re tools to give end-users their own voice. In *Proc. of the 2010 18th IEEE International Requirements Engineering Conference*, pages 37–46. IEEE, 2010.

[56] H. Shimada, H. Nakagawa, and T. Tsuchiya. Goal model construction based on user review classification. In *Joint Proceedings of REFSQ-2019 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track co-located with the 25th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2019)*, 2019.

[57] B. Szmrecsanyi. *Grammatical variation in British English dialects: A study in corpus-based dialectometry*. Cambridge University Press, 2012.

[58] R. Řehůřek and P. Sojka. Software framework for topic modelling with large corpora. 2010.

[59] J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[60] K. Watanabe, H. Nakagawa, and T. Tsuchiya. KAOS modeling editor: A tool for semi-automated goal modeling. in *Companion Proc. of the 42nd International Conference on Conceptual Modeling (ER)*, pages 1–5, 2023.

[61] H. Wu, W. Deng, X. Niu, and C. Nie. Identifying key features from app user reviews. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 922–932. IEEE, 2021.

[62] E. S. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of ISRE'97: 3rd IEEE International Symposium on Requirements Engineering*, pages 226–235. IEEE, 1997.

[63] D. Zowghi. "Affects" of User Involvement in Software Development. In *Proc. of the 2018 1st International Workshop on Affective Computing for Requirements Engineering (AffectRE)*, pages 13–13. IEEE, 2018.