Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MASTER THESIS

An LLM-Based Tool for Knowledge Retrieval from (Heterogeneous) Issue Tracking Systems

submitted by

Lukas Ciepielowski

Faculty: MIN

Department: Informatics

Group: Applied Software Technology (MAST)

Course: Master of Science Informatics

Student ID: 7028745

Supervisor: Prof. Dr. Walid Maalej

Co-Supervisor: Dr. Steffen Hauf

Mentor: Tim Puhlfürß

Abstract

[Context] Most software engineering projects rely on Issue Tracking Systems (ITSs) to organize and process work packages. Consequently, this knowledge is distributed across many issues and eventually multiple ITSs. This distribution results in difficulties for developers attempting to retrieve that knowledge. Furthermore, developers tend to favor the presentation of information in smaller chunks. Current machine learning approaches and Large Language Models (LLMs) enable searching distributed datasets and summarizing the discovered knowledge. **[Objective]** The study aims to understand how a novel approach for knowledge retrieval in the context of ITSs is perceived by practitioners and how it performs in contrast to classic ITSs. **[Methodology]** This study reports on a controlled experiment involving 30 participants divided into three test groups. I developed a chatbot tool for this experiment that answers user's questions by generating a summary of issues and a link list tool version that enables users to use natural language searches to get a list of relevant issues. I analyzed the user experience, time efficiency, confidence, and search prompts among the test groups. Additionally, I assessed the trustworthiness of participants in the generated summary and applied automatic metrics to evaluate the chatbot summary. **[Results]** The study revealed that practitioners preferred using the chatbot tool to retrieve knowledge. They also reported the highest confidence in their answers. Participants using the link list tool performed the fastest. The study also revealed the use of different prompt patterns among the test groups. Furthermore, a brief summary containing only essential information is sufficient for making informed decisions. **[Conclusion]** The findings of my study demonstrate that a concise presentation of information is crucial. Otherwise, it leads to crucial information being overlooked. Despite the high level of confidence participants expressed in their responses, they did not fully trust the summaries generated by the LLM.

Zusammenfassung

[Kontext] Die meisten Softwareentwicklungsprojekte stützen sich auf ITSs, um Arbeitspakete zu organisieren und zu verarbeiten. Folglich ist dieses Wissen über viele Issues und schließlich mehrere ITSs verteilt. Diese Verteilung führt zu Schwierigkeiten für Entwickler, die versuchen, dieses Wissen abzurufen. Darüber hinaus bevorzugen Entwickler die Darstellung von Informationen in kleineren Einheiten. Aktuelle Ansätze des maschinellen Lernens und LLMs ermöglichen die Suche in verteilten Datensätzen und die Zusammenfassung des gefundenen Wissens. **[Zielsetzung]** Die Studie zielt darauf ab, zu verstehen, wie ein neuartiger Ansatz zum Wiederauffinden von Wissen im Kontext von ITSs von Fachleuten wahrgenommen wird und wie er sich im Vergleich zu klassischen ITSs verhält. **[Methodik]** Diese Studie berichtet über ein kontrolliertes Experiment mit 30 Teilnehmern, die in drei Testgruppen aufgeteilt wurden. Für dieses Experiment habe ich ein Chatbot-Tool entwickelt, das Fragen von Nutzern beantwortet, indem es eine Zusammenfassung von Issues generiert, sowie eine „Link List“ Tool Version, die es Nutzern ermöglicht, über eine Suche in natürlicher Sprache eine Liste relevanter Issues zu erhalten. Ich analysierte die Benutzererfahrung, die Zeiteffizienz, das Vertrauen und die Suchanfragen der einzelnen Testgruppen. Außerdem bewertete ich die Vertrauenswürdigkeit der Teilnehmer in die generierte Zusammenfassung und wandte Metriken zur Bewertung der Chatbot-Zusammenfassung an. **[Ergebnisse]** Die Studie ergab, dass die Fachleute es vorzogen, das Chatbot-Tool zum Abrufen von Wissen zu nutzen. Sie gaben auch die größte Zuversicht in ihre Antworten an. Die Teilnehmer, die das „Link List“ Tool verwendeten, waren am schnellsten. Die Studie zeigte auch, dass die Testgruppen unterschiedliche Eingabemuster verwendeten. Darüber hinaus reicht eine kurze Zusammenfassung mit den wichtigsten Informationen aus, um fundierte Entscheidungen zu treffen. **[Fazit]** Die Ergebnisse meiner Studie zeigen, dass eine kompakte Darstellung von Informationen entscheidend ist. Andernfalls werden wichtige Informationen übersehen. Trotz des hohen Maßes an Zuversicht, das die Teilnehmer in ihren Antworten zum Ausdruck brachten, vertrauten sie den vom LLM generierten Zusammenfassungen nicht vollständig.

Contents

List of Figures

List of Tables

Listing

1 Introduction

1.1 Problem Description

Imagine, you are a software developer that must implement a new feature. Many users rely on the underlying system, and it is essential that it remains operational at all times. The feature is complex and demands a deep understanding of the system. Therefore, you want to retrieve all necessary information from relevant documentation artifacts before implementing that feature. A colleague gives you a hint that some of this information was discussed in multiple issues a few weeks ago. Consequently, you conduct a time-intensive search process in the multiple ITSs of your company to find the relevant issues, including the ones with a vague issue title. Your feature implementation is delayed significantly because you are investing a substantial amount of time searching for documentation scattered across various artifacts. This leads to issues for both the developers, as mentioned previously, and the users who experience delays in receiving new features. To accelerate the development process you seek a tool-based solution. Your problem could be resolved by a tool that can retrieve extensive knowledge from multiple ITSs based on a single input prompt in natural language. Furthermore, the tool could summarise the information, resulting in a faster, more efficient workflow.

1.2 Research Objectives

Many software engineering projects rely on ITSs to organize and process work packages [8]. Furthermore, developers use issues to document additional information regarding the software solution, e.g. in the issues' descriptions or within additional comments and discussions [4]. However, this information is difficult to retrieve for third parties, such as a developer searching for documentation before executing a software change. Moreover, issues are utilized differently by each organization. As a result, this leads to notable disparities in both the quality and quantity of issues [46, 35]. Consequently, it can leave third parties uncertain about where to access relevant information.

LLMs have become an essential part of recent research. Their potential for improving multiple aspects of software engineering is evident [9], yet they struggle to provide answers to questions outside their training scope [26, 19]. This is a frequently observed phenomenon in machine learning based applications. Recent efforts in research have been made to feed LLMs with external knowledge as additional context information. These tools are able to process information from external sources, like knowledge bases [61] or PDF files [49], feed them to the LLM and deliver external knowledge to users without the need for re-training. Handling multiple, heterogeneous documentation sources can prove to be demanding. Researchers, therefore, advise using a unified, collaborative documentation platform, which facilitates maintenance and contribution [1, 2]. Research about the format of documentation has shown that interactive documents that gradually reveal parts of their information are preferred by developers in contrast to extensive static documents [47].

LLM-assisted chatbots can selectively provide users with the precise information they request during interactive chats, as opposed to delivering the entire documentation. While various studies have focused on extending LLMs with external knowledge, the adaptation of such techniques for ITSs has yet to be thoroughly researched. Recent work has focused on leveraging LLMs to summarize issue threads [30]. However, this research primarily concentrates on summarizing knowledge and does not address the knowledge retrieval process. To the best of my knowledge, there are no knowledge retrieval tools for ITSs that incorporate LLM assistance to facilitate a chat-like experience. Specifically, tools that support knowledge retrieval from multiple ITSs.

This thesis aims to address the following research questions:

- **RQ1:** Do practitioners prefer the retrieval of issue tracker knowledge via a chatbot summary, a link list, or the classic issue tracker interface?
- **RQ2:** How time-efficient are practitioners in retrieving knowledge from the tools?
- **RQ3:** How confident are practitioners when providing answers based on information from a chatbot compared to their own searches in a classic issue tracker?
- **RQ4:** Which search prompts do practitioners use to retrieve knowledge from the chatbot in comparison to the classic issue tracker user interface (UI)?
- **RQ5:** How necessary do practitioners assess the availability of links to the issue tracker system in addition to an issue summarization?
- **RQ6:** How much information from the original issue thread do practitioners require in an issue summary to make informed decisions?

1.3 Contributions

I will share results from a controlled experiment with practitioners. The experiment involved three test groups and a total of 30 participants. I developed a chatbot tool for this experiment. The tool fetches issues from various ITSs and processes the issues' information to be stored in a database. A chatbot interface enables users to retrieve documentation from multiple ITS in natural language. The LLM generates a summary based on the issue information to answer the user's question. The tool utilizes Retrieval-Augmented Generation (RAG) [5] to accomplish this. Furthermore, I created a second tool version that centralizes issue information from multiple ITS and provides a natural language search to look through all issues.

The experiment offers valuable insights into the potential of tools assisted by LLMs to enhance the knowledge retrieval process from ITSs. Moreover, it demonstrates how practitioners perceive LLM tools and identifies the remaining challenges that still need to be addressed.

2 Related Work

2.1 Knowledge Needs

Research has brought attention to the irreplaceable significance of documentation despite persisting challenges and issues. Maalej et al. conducted a large-scale study to investigate code comprehension. The authors identified the importance of face-to-face communication in knowledge sharing. Moreover, knowledge about implementation logic and the intended use of a program helps to understand the code, but this knowledge is rarely documented. The study revealed a significant gap between the actual usage of program comprehension tools and what is available in research. Developers prefer informal methods such as code comments and emails over specialized tools. However, this approach may result in information being scattered across different tools, which can cause problems for developers. They consider source code to be a more reliable source of information than written documentation because, unlike documentation, code is a resource that cannot become outdated. Furthermore, the study indicates that program comprehension and knowledge exchange methods are highly dependent on context, including the task, technology, and individual experience. The findings support the need for personalized, context-aware tools and methodologies to enhance program comprehension and knowledge sharing [38].

Previous research has examined various types of knowledge in Application Programming Interface (API) documentation. This analysis can assist practitioners in assessing the quality of their documentation, improving its organization, and reducing the amount of non-essential content [37]. Developers usually document information about user stories, requirements, source code, and architecture. They utilize several tools for documentation, including source code comments, wiki-based articles, flow charts, and more [24]. Notably, code comments are prone to becoming obsolete over time. A large-scale study by Aghajani et al. highlighted multiple documentation challenges. Significant hurdles were identified, including ensuring correctness, completeness, and maintaining up-to-date documentation. In addition, usability, maintainability, and readability have been identified as problems related to documentation content. Traceability was also frequently cited as a problem. A consistent documentation format and an effort to archive old documentation can be effective [1].

Researchers suggest leveraging collaborative platforms to host the documentation. This encourages contributors to add new content or correct existing inaccuracies in the documentation [2]. However, it is important to consider implementing clear guidelines and establishing mechanisms to encourage the creation of documentation. Although it requires additional effort, code examples can significantly enhance developers' comprehension and are, therefore, often desired. Developers face problems with using the correct data type, headers, and body when dealing with a new REST API. Usage examples help to reduce errors and improve the success rate [57].

Research differentiates between two developer personas: "systematic" and "opportunistic." Systematic developers approach tasks top-down, aiming to understand the system entirely before focusing on specific components. They typically begin with the introductory "Getting Started" section or tutorials when exploring a new library. Opportunistic developers, however, focus on specific tasks and the problems they face, preferring to engage in coding immediately. They look for information, such as code examples, that directly relate to their immediate needs. Forums such as Stack Overflow are commonly utilized, given that the answers provided are widely considered valuable. The timestamps included for questions and responses indicate how up-to-date this information is. Furthermore, the question-answer format seems to be appealing for developers as it conveys a sense of community [40]. Wu et al. proposed an approach to retrieve API knowledge from API tutorials and Stack Overflow posts based on natural language queries. This approach combines the extensive information available in API tutorials with the information on specific programming tasks available in Stack Overflow posts [64]. When encountering issues with an API, developers prefer to use search engines like Google before reading the documentation. It is therefore important that the documentation is visible on the web and available to search engines [40]. The inclusion of a search function is also regarded as highly important. Moreover, documentation content should be organized according to API functionalities. Essential information should be redundantly presented, appearing both in the documentation text and within code examples [41]. While comprehensive documentation is essential, it's important to note that practitioners often prefer interactive documents that reveal only parts of the documentation at a time. This approach allows them to concentrate on relevant information, minimizing distraction from unrelated content. However, this approach also risks hiding information that may be necessary but is not immediately apparent to the reader [47].

Analysis of open-source GitHub repositories indicates a noticeable shortage of effective strategies for integrating documentation seamlessly with source code. Enhancing the integration of documentation with source code could lead developers to take care of documentation more thoroughly [51]. Links play a crucial role in establishing connections between source code and relevant knowledge, both within the source code and in commit messages.

Despite their widespread use, links are vulnerable to decay, which can result in the loss of information. Studies have shown that approximately 18% of links in source code and 70% of links commit messages are non-operational. Researchers recommend using permanent links and emphasizing the maintenance of these links [65, 20].

In conclusion, documentation plays a crucial role in software development, yet it encounters several challenges. Although research introduces various documentation tools, they are not always utilized in the industry. Moreover, documentation is prone to becoming outdated, making it difficult to maintain correctness and completeness. To overcome these challenges, researchers propose a centralized and collaborative platform for documentation. Research also shows that practitioners frequently prefer to utilize search engines before consulting comprehensive documentation. In addition, they find a question-answer format, such as that used on Stack Overflow, appealing. Furthermore, practitioners tend to favor being presented with specific sections of documentation at a time, rather than navigating through extensive documents. It is, therefore, important to ensure that knowledge is easily retrievable for practitioners, ideally through an interactive format that presents only the essential parts of the documentation.

2.2 Issue Tracking Systems

ITSs are an essential part of research in computer science. Many software engineering projects use ITSs to organize and process work packages. Projects that are larger in terms of lines of code, involve a greater number of developers, or have a longer history are more likely to have reported issues [8]. Additionally, ITSs are frequently used to document software features that have been implemented or are scheduled for implementation [43]. Therefore, Merten et al. conducted a study to detect software feature requests using machine learning approaches [42]. Furthermore, ITSs not only serve as a repository for tracking bugs, features, and requests but also as a central point for communication. It is common for various stakeholders, both inside and outside the software team, to be involved in a project's ITS [48]. Consequently, ITSs encapsulate a considerable amount of organizational knowledge. Over the years ITSs have evolved from a relatively simple tool to a central part of communication and coordination in software development [7].

Arya et al. identified that issue comments often contain information that can be beneficial to stakeholders. This information accumulates over time as discussions are held in issue comments. The authors identified various types of information present in issue comments, including discussions of solutions, bug reproduction, workarounds, and more [4]. Furthermore, feature requests and the associated comments raised in an ITS provide valuable knowledge for requirements elicitation [62]. These requests can be treated as just-in-time requirements [16]. However, information overload can overwhelm developers, leading to frustration and the risk of overlooking issues. To address this, Baysal et al. introduced a dashboard-like tool designed to mitigate information overload [6]. Furthermore, research about issue prioritization in GitHub has been conducted. These studies reveal that issue labels are neither commonly used nor effective for prioritizing issues. The authors also explored the application of learning-to-rank methods to prioritize issues in a repository [21].

Several studies have investigated the interconnections and dependencies among issues [35, 52]. Issue links can be classified into the following categories: *General Relation*, *Duplication*, *Composition*, *Temporal/Causal*, and *Workflow*. *Duplicate* links are particularly useful as identifying duplicate issues reduces additional effort for maintainers. In addition, Lüders et al. analyzed the performance of state-of-the-art models in predicting issue links and the determinants of their performance [36]. Addressing and resolving bugs is a crucial aspect of software engineering. For this reason, the automatic detection of duplicate bugs is a highly valued feature among practitioners. It enables developers to concentrate on critical bugs [72]. Zhang et al. assessed the effectiveness of duplicate bug detection techniques. They conducted comparative analyses using three ITSs and discovered that existing methods are less effective on GitHub compared to Bugzilla and Jira [70].

Considerable effort is made to provide datasets of ITS to facilitate further analysis in research. Various datasets of Jira, a widely-used ITS, repositories have been proposed [46, 12]. Likewise, a dataset of GitHub repositories was also introduced [11].

Overall, recent research shows that ITSs play a crucial role. ITSs are used by numerous projects in both open source and industry. They serve as an important communication tool by aggregating the different stakeholders of a project. This results in valuable knowledge being distributed across various issues. However, effectively retrieving this information can be challenging. This motivated my thesis in two ways: ITSs play a crucial role in research, and knowledge gained from ITSs can be highly valuable.

2.3 LLM-based Knowledge Retrieval Tools

LLMs have become an essential part of recent research. It is evident that these tools hold promise in improving various dimensions of software engineering. They perform particularly well in tasks such as text generation, text comprehension, question answering, and logical reasoning [9]. Consequently, the usage of LLMs has significantly increased, being used as chatbots, in programming tasks, and in creative work. Despite recent advancements, challenges still remain in the field. These include the limited context length, the need for fine-tuning to achieve optimal performance in downstream tasks, outdated knowledge, hallucinations, and more. LLMs are also susceptible to prompt changes. Even a small difference in the prompt can result in significantly different outputs. Moreover, LLMs may also produce different results for the same prompt [59]. In addition, human-written ground truth is often required to properly evaluate these models. It is also challenging to identify text generated by LLM [25]. Furthermore, recent work has shown that OpenAI's GPT models can be misled to generate toxic and biased outputs. The models can also compromise private information. According to Wang et al, GPT-4 is generally more trustworthy than GPT-3.5. However, it is also more vulnerable to attacks, as it follows instructions more precisely, even if they are misleading [60]. Sivaraman et al. conducted a study on the acceptance of Artificial Intelligence (AI)-based support tools in healthcare. The authors concluded that AI tools were accepted the most as an additional source of information, rather than as the primary source [55]. Additionally, research on AI-assisted decision-making has shown that displaying a confidence score can help increase trust in the responses. However, only if the confidence score displayed is above 80%. Below this threshold, it decreases the trust in the model [71].

In recent years multiple tools for text summarization have been proposed. Kumar et al. contributed with a model capable of summarizing issue threads using the GPT-3-5 Turbo model. These summaries are of improved quality and shorter in length compared to other approaches. The authors analyzed the performance of different summary lengths and concluded that a summary containing 30-50% of the original text strikes a good balance between conciseness and information retention [30]. Additionally, GitSum was introduced as a tool designed to summarize README.MD files, aiming to populate the "About" field in repositories. This field offers a concise summary of the repository's main functionalities and objectives [13]. Xu et al. proposed an approach to generate commit messages based on changes made to the source code in a commit. Their approach includes a broader context than just the differences introduced by a commit, providing a better foundation for generating commit messages [66].

The evaluation of generated summary texts can be performed using metrics like ROUGE [34] or METEOR [32]. Practitioners commonly utilize these metrics when evaluating generated text [13, 66, 27, 68]. Even though these automatic metrics are not reliable enough to replace human evaluation for text generation tasks [23], they can provide additional insights. Recent research has proposed metrics that use pre-trained LLMs to evaluate generated texts, thus providing metrics that are more in line with human judgments. BERTscore, for instance, utilizes the BERT model to calculate a similarity score between tokens in candidate and reference sentences. The token similarity is computed through contextual embeddings rather than exact matches [69]. Similarly, BARTscore employs the BART model to evaluate LLM generated text [67].

LLMs are limited in their ability to provide answers to questions beyond their training scope, as indicated by studies such as [26] and [19]. This is a frequently observed phenomenon in machine learning based applications. Consequently, recent research efforts have focused on leveraging external knowledge to enhance the capabilities of LLMs. Wang et al. proposed KnowledGPT, a framework that integrates LLMs with various Knowledge Bases (KBs). It features a personalized KB that accumulates knowledge from user inputs. It serves as a symbolic memory, providing the ability to store and access specialized knowledge. The retrieval process consists of three steps: firstly, generate a search code based on a user prompt, retrieve knowledge using that search code, and finally, read the retrieved knowledge to answer the question [61]. ChatDB has also been proposed, a framework that integrates databases as symbolic memory for LLMs. It operates by executing the appropriate SQL query to retrieve or insert data into the database in response to a user prompt [22]. Moreover, a chatbot designed to interact with PDF files was introduced, enabling users to query information from a PDF file by asking questions in natural language [49]. The internet is the largest source of continuously updated information. Therefore, Komeili et al. proposed a model capable of generating an internet search query based on a user's prompt. By utilizing this query, the model then retrieves potentially up-to-date information. Additionally, they provide a new dataset collected from human-to-human conversations to evaluate their model [28]. In conclusion, these tools extract information from several external sources and integrate this information as custom context into LLMs.

Recent years have shown the high impact of LLMs on research and software development. LLMs show remarkable performance in various tasks, especially in text generation and understanding. Furthermore, LLMs are often used for text summarization tasks. As a result, many approaches have been proposed for summarizing issue threads, README files, and generating commit messages. Consequently, new metrics for evaluating LLM-generated text have been introduced. However, LLMs also have some drawbacks, such as limited context length and outdated knowledge. Recent research has focused on integrating external knowledge into LLMs to counteract these drawbacks. However, to the best of my knowledge, no work has focused on integrating external knowledge from multiple heterogeneous ITS into LLMs.

3 Methodology

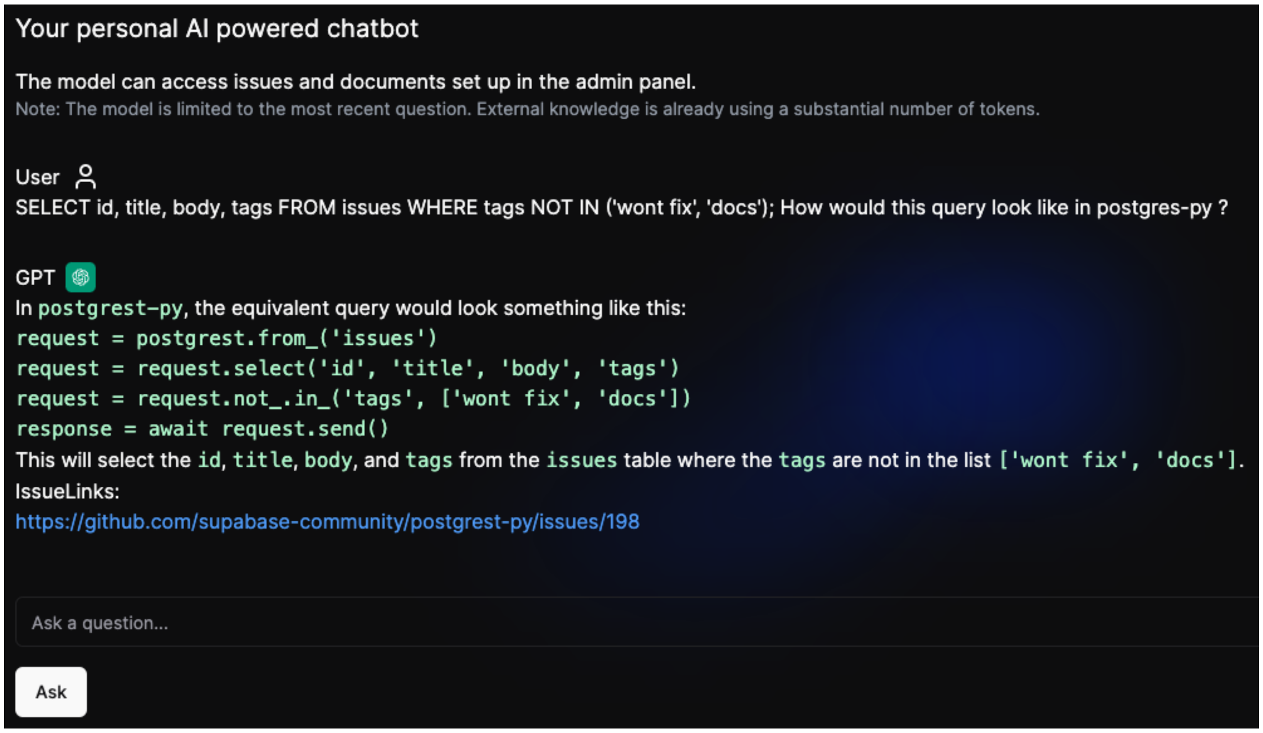
3.1 Experimental Design

I designed a controlled experiment in a laboratory setting. The experiment was structured to conclude within approximately 30 minutes. Upon welcoming participants, I gave a brief introduction covering the privacy policy, study purpose, and tasks. Moreover, participants were instructed not to strive for a perfect answer and that every answer is valuable, right or wrong. In order to maintain a calm environment, the participant was isolated in an empty room only with the experimenters present. To reduce observational bias, I positioned myself facing away from the screen while participants were evaluating the tool [39]. Furthermore, participants were instructed to use the thinking aloud technique to express their thoughts while executing the tasks [3].

The experiment involved three test groups. The first group was the chatbot group, which utilized a tool that included interaction with a chatbot. This chatbot provided immediate information without requiring manual searches (Figure 3.1). Additionally, links to relevant issues were included alongside the summary. The second group, the link list group, utilized a simplified version of the tool with a search function, providing participants with links to relevant issues (Figure 3.2). I explained the tool's functionality further for all test groups to ensure that participants were familiar with its usage.

To locate relevant information, participants followed the provided links and searched within the designated issues. In both groups, the same mechanism was used to retrieve relevant issues. The issue list (control) group engaged with the search function of two ITSs, requiring participants to skim through issues on a website (Figure 3.3).

Given that each group exclusively used one of the tool versions, this experiment was conducted as a between-subjects study. Within-subjects studies typically offer greater statistical power with fewer participants [56], making it a preferable option when participant recruitment is challenging as they involve each participant testing all tool versions. To address potential carry-over effects, it would be necessary to allocate distinct sets of issues for each question. This, however, introduces the challenge of ensuring issue similarity in difficulty, making the comparison between tool versions more complex. The between-subjects study



Your personal AI powered chatbot

The model can access issues and documents set up in the admin panel.
Note: The model is limited to the most recent question. External knowledge is already using a substantial number of tokens.

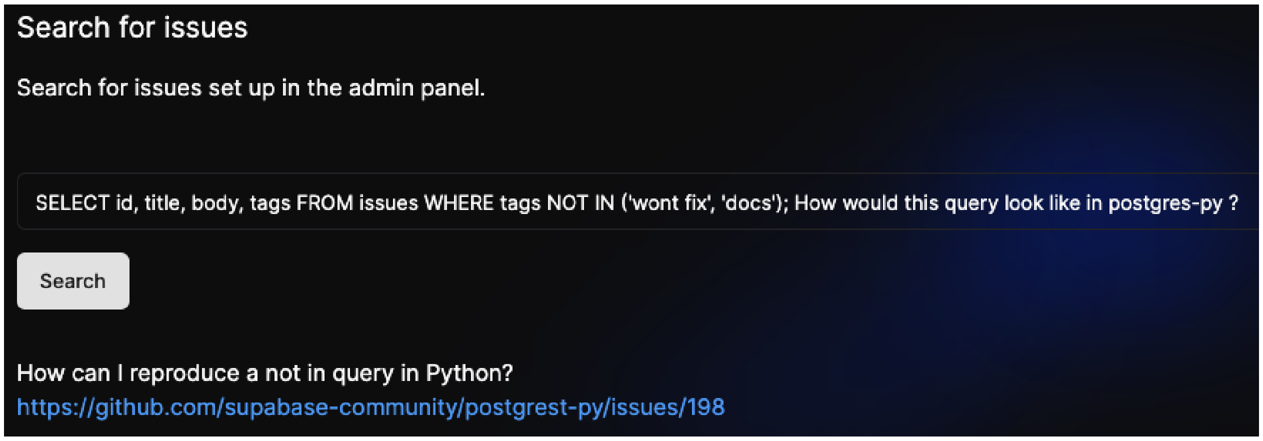
User
SELECT id, title, body, tags FROM issues WHERE tags NOT IN ('wont fix', 'docs'); How would this query look like in postgres-py ?

GPT
In postgrest-py, the equivalent query would look something like this:

```
request = postgrest.from_('issues')
request = request.select('id', 'title', 'body', 'tags')
request = request.not_.in_('tags', ['wont fix', 'docs'])
response = await request.send()
```

This will select the id, title, body, and tags from the issues table where the tags are not in the list ['wont fix', 'docs'].
IssueLinks:
https://github.com/supabase-community/postgrest-py/issues/198

Figure 3.1: The chatbot tool version. Illustration of the interaction between a user and the LLM, showcasing a question posed by the user and the corresponding response.



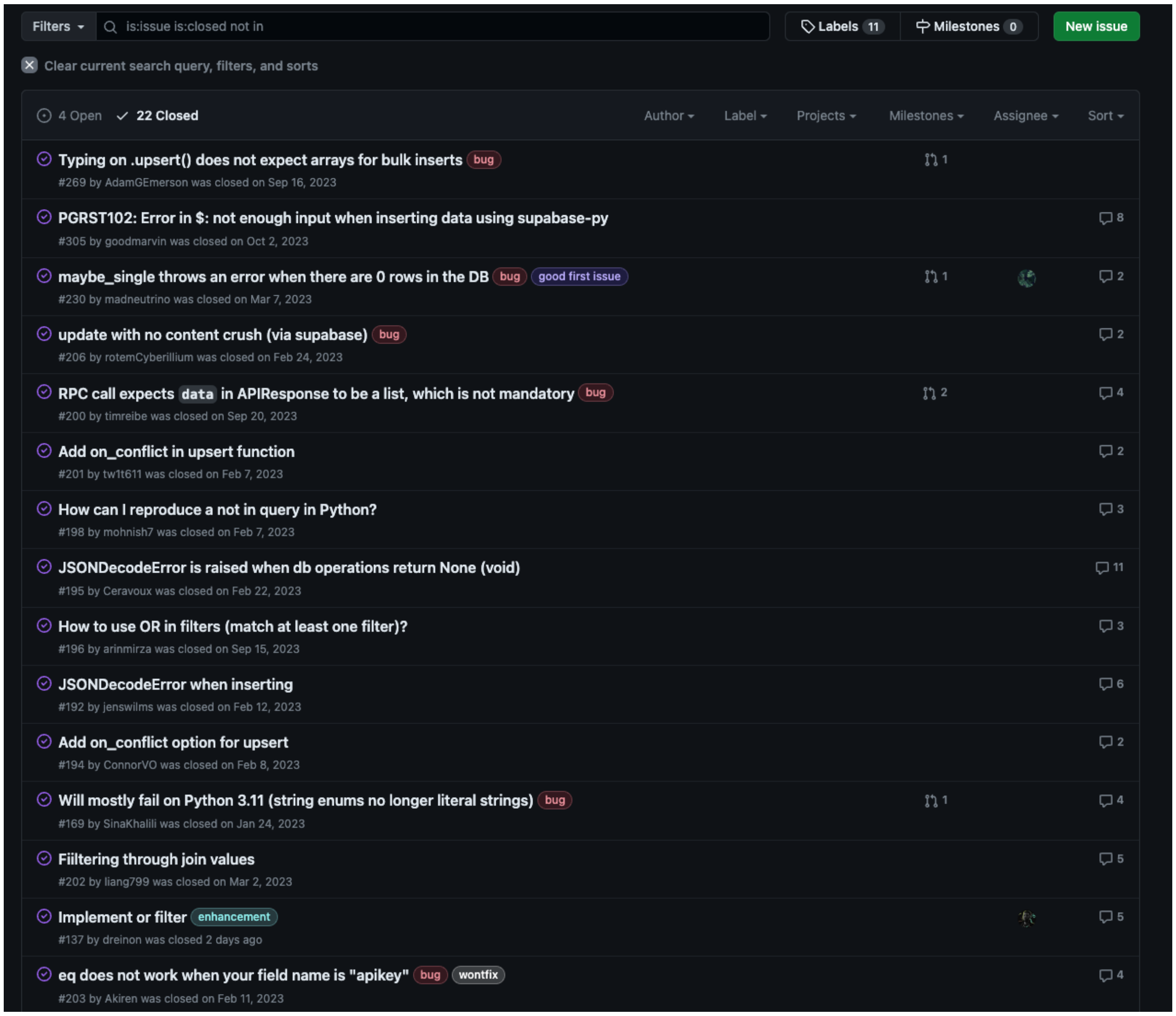
Search for issues

Search for issues set up in the admin panel.

SELECT id, title, body, tags FROM issues WHERE tags NOT IN ('wont fix', 'docs'); How would this query look like in postgres-py ?

How can I reproduce a not in query in Python?
https://github.com/supabase-community/postgrest-py/issues/198

Figure 3.2: The link list tool version. Illustration of a user using natural language to search for issues.



Typing on .upsert() does not expect arrays for bulk inserts bug
#269 by AdamGEmerson was closed on Sep 16, 2023

PGRST102: Error in $: not enough input when inserting data using supabase-py
#305 by goodmarvin was closed on Oct 2, 2023

maybe_single throws an error when there are 0 rows in the DB bug good first issue
#230 by madneutrino was closed on Mar 7, 2023

update with no content crush (via supabase) bug
#206 by rotemCyberillium was closed on Feb 24, 2023

RPC call expects `data` in APIResponse to be a list, which is not mandatory bug
#200 by timreibe was closed on Sep 20, 2023

Add on_conflict in upsert function
#201 by tw1t611 was closed on Feb 7, 2023

How can I reproduce a not in query in Python?
#198 by mohnish7 was closed on Feb 7, 2023

JSONDecodeError is raised when db operations return None (void)
#195 by Ceravoux was closed on Feb 22, 2023

How to use OR in filters (match at least one filter)?
#196 by arinmirza was closed on Sep 15, 2023

JSONDecodeError when inserting
#192 by jenswilms was closed on Feb 12, 2023

Add on_conflict option for upsert
#194 by ConnorVO was closed on Feb 8, 2023

Will mostly fail on Python 3.11 (string enums no longer literal strings) bug
#169 by SinaKhalili was closed on Jan 24, 2023

Fiiltering through join values
#202 by liang799 was closed on Mar 2, 2023

Implement or filter enhancement
#137 by dreinon was closed 2 days ago

eq does not work when your field name is "apikey" bug wontfix
#203 by Akiren was closed on Feb 11, 2023

Figure 3.3: Issue list of an open-source repository from GitHub. Filtered by closed issues.

mitigates the risk of the carry-over effect [63]. To control the comparability of the groups, I randomly assigned participants to each group. Moreover, by assigning the same tasks to every participant, I effectively reduce concerns about the choice of issues influencing study outcomes.

The experiment incorporates demographic questions designed to assess the characteristics of the participant groups. I included questions about the experience with SQL, React, and ITS since the tasks involved these technologies. Participants were required to enter their years of experience; I also accepted values such as 0.5 and 1.5. While experience with these technologies may enhance task performance, I tried to select tasks that are accessible to participants without prior experience. This is why I chose two distinct technologies for the tasks: a UI library and a database library. I aimed to minimize the risk of participants facing difficulties, especially those who might struggle with a specific field or technology. Following that, participants were presented with three tasks: one related to the UI library and two related to the database library. I explicitly told the participants to ask me if they had any questions about the current task, ensuring their understanding of the task. To further investigate this potential concern, each question was accompanied by additional inquiries. Before participants used the tool to retrieve an answer for the respective question, they were instructed to rate how confident they felt about their understanding of the question, using a Likert scale ranging from 1 to 5. A time limit of 5 minutes per task was imposed to ensure that the runs remained comparable (maximum 30 minutes total), emphasizing that no perfect solution was necessary. Additionally, I recorded the duration required for participants to complete each task to address RQ2. Furthermore, to answer RQ3, participants were asked to assess their confidence in the correctness of their answers, utilizing a Likert scale once again. Participants were also instructed to copy and paste both the prompt they used and the answer given by the tool into the questionnaire. This enables further analysis of the summary quality produced by the LLM and the prompt patterns employed by participants. To evaluate the summaries and address RQ6, I utilized metrics like ROUGE and BERTscore. Additionally, I employed open coding to identify the prompt patterns that were used, thus answering RQ4. Participants in the summary group were asked whether it was necessary to click on one of the provided links or if the summary provided by the LLM was sufficient, in order to address RQ5. If they responded affirmatively, they were given the opportunity to rate the level of necessity on a Likert scale ranging from 1 to 5. This question aims to determine if participants utilized the provided issue links in addition to the summary and if they discovered any information within the issues that was not included in the summary. Furthermore, participants were asked to rate the trustworthiness of the summaries provided by the tool if the issue links were not available.

Following these questions, the experiment also incorporated the standardized User Expe-

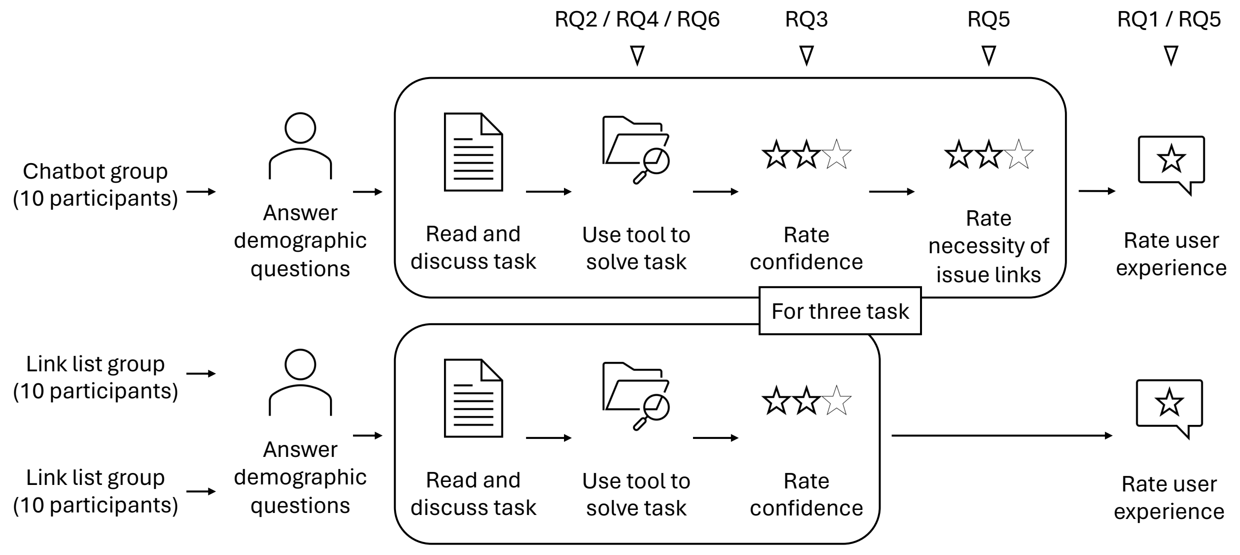


Figure 3.4: Overview of the experimental design: A schematic representation illustrating the structured procedure and activities undertaken by the Chatbot, Link list, and Issue list groups across three testing phases.

rience Questionnaire (UEQ) to evaluate the usability of all three versions [31] and to answer RQ1. Finally, participants were given the opportunity to comment on the positive and negative aspects of the tool, along with suggestions for further improvements. For an overview of the experimental design refer to Figure 3.4. The experimental design was verified against the Pre-Submission Checklist [53].

3.2 Task Selection

For the experiment, I selected issues from open-source projects that were intended to be straightforward tasks, aiming to reduce the cognitive load on participants. Moreover, I focused exclusively on closed issues to ensure that a solution was available. Given that the questions are sourced from various open-source projects, it is expected that participants may not be familiar with them. Additionally, by selecting issues from potentially unfamiliar projects, I aimed to reduce expert bias. This approach also enables the interviewing of developers from various fields. These questions were intentionally selected from two distinct platforms, namely GitHub and GitLab. This deliberate choice was made to prevent the experiment from being influenced by platform-specific knowledge. The issues follow a question-and-answer format, as required for the experiment. This also reflects use cases where ITSs are employed for user support. Users can raise issues that require a response from the support team. Solutions, together with additional knowledge, may already exist distributed across various issues. When a recurring issue is encountered by multiple users, it is crucial to facilitate easy access to information from previous solutions. This scenario reflects a common use case at European XFEL (EuXFEL). The five-minute time constraint does not entirely align with this use case, as usually, more time is available to find a solution. However, it was crucial to maintain consistency and comparability across the experiment rounds. Furthermore, the total duration of the experiment should not exceed 30 minutes. All three tasks involved recurrent inquiries, thus a solution is already available. Appendix A provides the exact wording of the tree task used in the controlled experiment.

Task 1: The `react-phone-number-input` library provides a React component that streamlines the process of inputting phone numbers. Based on the phone number in the input field, it automatically identifies the country, applies the appropriate format, and displays the corresponding country flag. For added flexibility, users can manually set the country by selecting the available options from a dropdown list. The initial task involved customizing the dropdown list to include Unicode flags also referred to as emoji flags. The correct solution was to implement a custom `countrySelectComponent`. No specific implementation was required. It was sufficient to indicate that a custom component should be used.

Task 2: `Postgrest-py` is a `PortgresREST` client for Python. PortgresREST is a standalone web server that turns your PostgreSQL database directly into a RESTful API. Postgrest-py enables developers to formulate SQL queries in Python syntax using a predefined set of methods. For the second task, participants were required to convert a traditional SQL query into postgrest-py syntax, specifically the `NOT IN` part of that query. The specific SQL query involved is presented in Listing 3.1. The expected solution should include syntax that matches `.not_.in_`.

```sql
SELECT id, title, body, tags FROM issues
WHERE tags NOT IN ('wont fix', 'docs');
```

Listing 3.1: The SQL query from the second task that participants were required to translate.

Task 3: The final task, also related to postgrest-py, highlighted an issue where a query returned data unexpectedly. The query was supposed to filter results based on whether an `apikey` matched a specific value. However, despite no `apikey` in the database matching the search criteria, the query still returned data. The issue stemmed from the column `apikey`, which was already in use internally. Resolving the issue involved identifying the naming conflict and renaming the column to prevent this unintended behavior.

3.3 Tool Creation

3.3.1 Implementation

The tool was built using Next.js[1] and Supabase.[2] This decision was primarily influenced by a starter kit[3] that processes all markdown files within a directory, utilizing them as custom context for the LLM. This starter kit focused on documentation files. It served as a starting point for the implementation of my tool. Additionally, I am well-versed in Next.js, a full-stack framework for building web applications based on React. Developers can use TypeScript as the sole programming language to write both frontend and backend code in a single repository. With the latest features such as React Server Components (RSC) and Server Actions, Next.js empowers developers to rapidly create full-stack applications. The utilization of Server Actions eliminates the necessity to write APIs for the interaction between the frontend and the backend of the tool. To mutate data you simply invoke a function that executes on the server, instead of going through a conventional API endpoint. This approach not only streamlines the development process but also facilitates the rapid evolution of the tool. The tool aims to enhance the knowledge retrieval process from multiple ITS with the assistance of LLMs. To accomplish this, the tool retrieves information from ITS to extract relevant data. For this, you paste the URL of a repository into the admin panel of the tool, see Figure 3.5.

The tool queries the title, description, and comments for each issue. Subsequently, it generates embeddings from the provided data, using the OpenAI embedding models (version *text-embedding-ada-002*) [15]. Embeddings are vector representations of text, where similar words or phrases have similar vectors, capturing the semantic relationships between them. These embeddings, along with the plain text of the issue, are stored in a database. Supabase provides a hosted PostgreSQL database together with a client library for Javascript. Whenever users ask the tool questions, it retrieves the most relevant embeddings and injects them as context for the LLM. However, the model is limited to the context of the most recent question, as the issue embeddings consume a significant number of tokens. The model receives this information as part of the prompt without the need for fine-tuning, exemplifying RAG [5]. This makes the tool highly dynamic, as new knowledge only needs to be retrieved from the ITS and becomes immediately available to the model. Refer to Figure 3.6 for a detailed overview of the tool. Relevant embeddings are obtained by employing a vector similarity function that compares the search embedding generated from the question with each embedding stored in the database. This comparison involves calculating the dot product between the two embeddings. A threshold value between zero and one determines which

[1]https://nextjs.org/
[2]https://supabase.com/
[3]https://github.com/supabase-community/nextjs-openai-doc-search

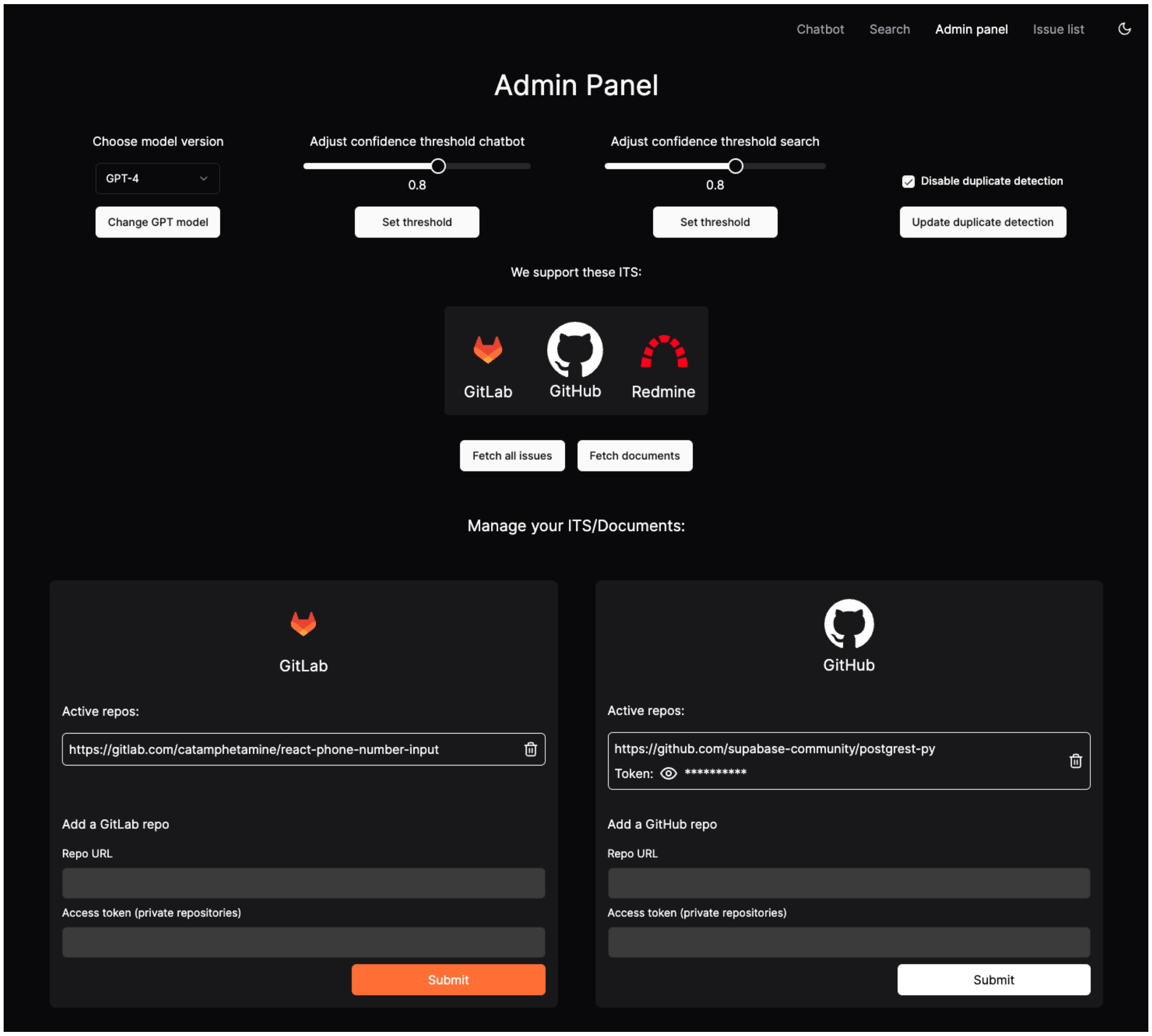

Chatbot
Search
Admin panel
Issue list
Admin Panel
Choose model version
GPT-4
Change GPT model
Adjust confidence threshold chatbot
0.8
Set threshold
Adjust confidence threshold search
0.8
Set threshold
Disable duplicate detection
Update duplicate detection
We support these ITS:
GitLab
GitHub
Redmine
Fetch all issues
Fetch documents
Manage your ITS/Documents:
GitLab
Active repos:
https://gitlab.com/catamphetamine/react-phone-number-input
Add a GitLab repo
Repo URL
Access token (private repositories)
Submit
GitHub
Active repos:
https://github.com/supabase-community/postgrest-py
Token: **********
Add a GitHub repo
Repo URL
Access token (private repositories)
Submit

Figure 3.5: The **Admin panel** of the tool allows admins to change the GPT model version and adjust the threshold of the similarity function for the chatbot and link list version of the tool. Furthermore, admins are able to add repository URLs to fetch new issues and provide an access token if required.

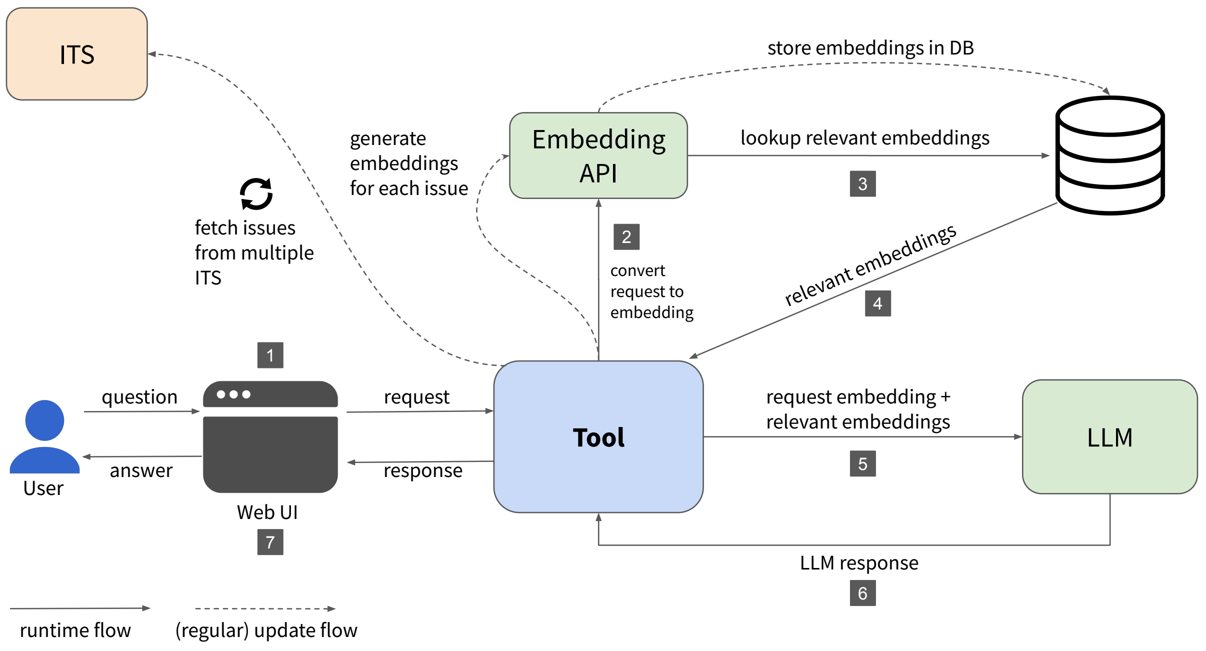

ITS
store embeddings in DB
generate embeddings for each issue
Embedding API
lookup relevant embeddings
3
fetch issues from multiple ITS
2
convert request to embedding
relevant embeddings
4
1
question
request
Tool
request embedding + relevant embeddings
LLM
answer
response
5
User
Web UI
7
LLM response
6
runtime flow
(regular) update flow

Figure 3.6: Flow diagram illustrating the runtime and update flow of the chatbot tool.

embeddings are considered relevant. The tool is accessible through a web-based UI.

I designed a custom prompt (Listing 3.2) which directs the model to utilize only the provided context information. If the required information is not available, the model is explicitly instructed not to generate an answer. This choice is based on the decision to constrain the model to questions where the answers are expected to be found in the KB. The purpose of this decision is to mitigate the risk of the model generating incorrect responses due to hallucination. However, this approach limits the model's ability to leverage its broad general knowledge, which may contain the correct answer. The prompt was designed based on recommendations from OpenAI [50]. In addition to utilizing a custom prompt, I configured the model's temperature to zero. This makes the model more deterministic, which was a desired attribute. The tool incorporates links to the specific issues that were utilized to generate the corresponding answer, as illustrated in Figure 3.1. Users can interact with the tool in two ways: through a chatbot that generates a summary or by receiving a list of the relevant links. Both options are integrated within the same tool, allowing users to select their preferred method of interaction.

```
const prompt = codeBlock`
    ${oneLine`
    Use the provided context sections delimited by triple quotes to answer
  the question.
    Respond in markdown format.
    If you are unsure and the answer cannot be found (context sections empty
 ), say
    "Sorry, I don't know how to help with that."
    `}

    Context sections: """
    ${contextText}
    """

    Question:
    ${sanitizedQuery}

    Answer as markdown (including related code snippets if available):
  `
```

Listing 3.2: The custom prompt instructs the model to only use the given context sections and refrain from providing an answer otherwise. The **contextText** consists of the relevant issues retrieved from the database.

3.3.2 Choice of LLM

The field of LLMs has seen significant attention, leading to the introduction of numerous capable models. I decided to utilize OpenAI's GPT models [44] for my study. OpenAI provides a JavaScript Software Development Kit (SDK), streamlining the interaction with the model. Additionally, they host the models, eliminating the need for further setup. This was particularly crucial, as it allowed me to dedicate my focus to the functionality of the tool. Furthermore, OpenAI offers a variety of different models. At the time of conducting this study, the most recent options were GPT-3.5-turbo (*gpt-3.5-turbo-0613* and *gpt-3.5-turbo-1106*) and GPT-4 (*gpt-4-0613*). Future OpenAI models are expected to continue improving performance, resulting in better summaries and more precise answers. Indeed, during my study, OpenAI released new GPT-4 models with a context size approximately 15 times larger than the current GPT-4 model. This enables the inclusion of additional context or the introduction of follow-up questions. Throughout the tool's development, I employed and tested GPT-3.5-turbo and GPT-4. The same questions were directed to both models and their responses were examined to identify any distinctions. GPT-4 provides superior, more appropriate answers to the questions asked. While comparing both models, I observed that GPT-3.5-turbo occasionally includes code snippets that are not included in the provided context. Consequently, these code snippets often proved to be inaccurate. No significant time differences between the models were observed. However, it's worth noting that longer responses tend to correlate with longer response times. Nevertheless, the tool provides the flexibility to switch between the two different models.

3.3.3 Knowledge Sources

The tool is able to retrieve knowledge from various sources such as ITS and documents. To retrieve data from different ITSs, a decision was necessary regarding which ITS platforms to support. The integration of these ITS required an initial implementation to retrieve the correct information. Subsequently, the tool enables the seamless connection of multiple repositories from the specified ITS directly within its UI. GitHub stands out as one of the most widely used public platforms with over 100 million users [14], making it an obvious choice. GitLab also ranks itself as one of the most used platforms, particularly favored by companies for its self-hosting capabilities [54]. Both platforms provide a convenient GraphQL API for retrieving information about repositories and related issues. After implementing an initial GraphQL query, all repositories of that platform can be fetched. Redmine is also supported considering EuXFEL's usage of this platform. Issues can be retrieved by calling a REST-API endpoint that returns all issues from the specified repository.

The primary focus of the tool is on ITS. Nevertheless, the tool also accommodates documents, such as Word or PDF files, which are stored directly in a designated directory. The tool segments Word documents into sections according to the headings within the document and subsequently stores these sections in the database. These sections can be retrieved in the same way as issues. Storing the entire document could result in challenges related to the token limit of LLMs.

3.3.4 Tool Configuration

As highlighted before in Section 3.1, the summary and the link list group used two different tool versions. The chatbot group received a summary provided by the LLM based on the relevant issues. Meanwhile, the link list group was provided with a list of relevant issue links. To ensure consistent results, the same threshold value of 0.8 was applied for both groups to retrieve relevant issues. This specific value was chosen through testing, aiming to identify relevant issues without overwhelming participants with excessive information.

For the chatbot group, the GPT-4 model (version *gpt-4-0613*) was used exclusively during the experiment. Even though the tool enables users to choose between the different versions, it was necessary to keep results consistent across all participants. Before the experiment, I connected the two repositories utilized for the experiment with the tool. Subsequently, the tool fetched all issues from these repositories, making them available for the GPT-4 model.

Table 3.1: Roles of participants in the three test groups; CB = chatbot, LL = link list, IL = issue list

	Software Engineers	**Researchers**	**Managers**	**Others**
CB / LL / IL	6 / 7 / 4	2 / 1 / 2	1 / 0 / 2	1 / 2 / 2

3.4 Experiment Execution

An initial pilot study involving three PhD students was conducted to estimate the completion time and to verify the clarity and comprehensibility of the tasks. Following that, I refined the wording of a question to improve clarity. Additionally, I added the functionality to add line breaks to the input field, as this was reported by the PhD students in the pilot study. The experiment primarily took place at the EuXFEL over a span of two days in February. Subsequently, I conducted the experiment with three professionals from Engel & Völkers Technology. A total of 30 participants were involved, with each test group consisting of 10 participants. The average completion time in the chatbot group was 28.27 minutes, 26.57 minutes for the link list group, and 28.26 minutes for the issue list group. All experiments were conducted in person.

Table 3.1 offers a comprehensive overview of the various roles of the participants. The predominant role was software engineers (17). This was followed by researchers (5), managers (3), and other roles (5). I recruited participants through connections in academia and industry, ensuring that each participant was engaged in the field of software engineering. The mean and standard deviation of years of experience in SQL was reported as 2 (2.97) for the chatbot group, 2.85 (4.73) for the linked list group, and 2.71 (5) for the issue list group. These significant differences in experience are attributed to outliers. While most participants indicated a low level of experience in SQL, ranging from 0 to 2 years, a few reported up to 14 years of experience. Participants in the chatbot group reported their experience in React with 0.75 (1.48), in the linked list group with 1.38 (2.72), and in the issue list group with 0.4 (1.1). This reflects a generally lower familiarity with this technology among participants. Experience with ITS was 5.55 (4.07) for the chatbot, 6.65 (4.40) for the linked list, and 5.05 (5.84) for the issue list group. Participants had the most experience with ITSs, likely due to the use of ITSs across various domains in software engineering. Similar to SQL, outliers with up to 15 years of experience influenced the standard deviation. A visual comparison of the three groups' experience with each technology is presented in Figure 3.7.

Initially, I included a question about the experience with LLMs. However, after the experiment, it became clear that this inclusion had not been carefully considered. The phrasing of the question was imprecise, potentially causing ambiguity among participants whether it referred to the experience in developing LLMs or simply interacting with them, for instance,

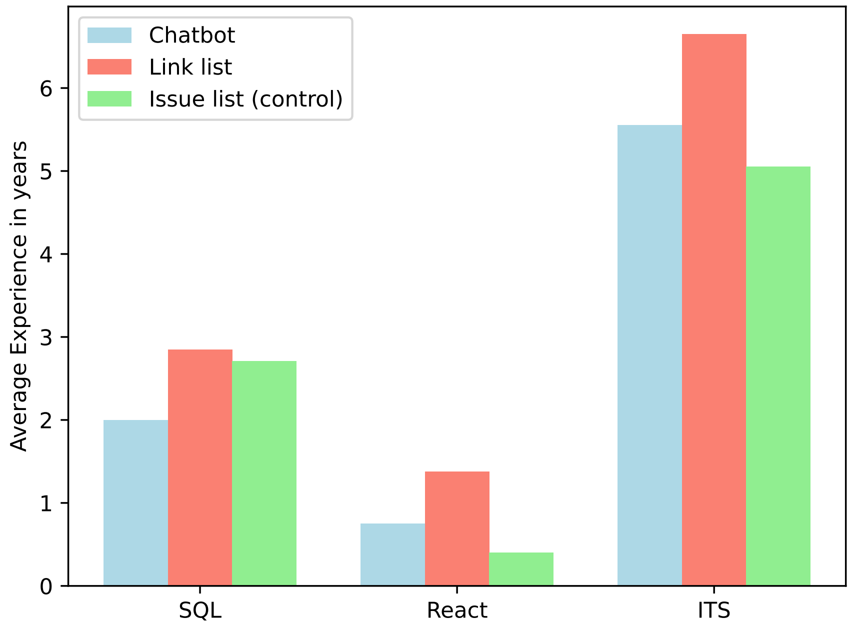


Figure 3.7: Mean experience in years of participants in SQL, React, and ITS

using ChatGPT. The original intention was to assess participants' ability to formulate efficient prompts. Due to this ambiguity, the question was excluded from the analysis.

Before starting each task, participants were asked to rate their confidence in understanding the task on a Likert scale from one to five. Participants first read the task and then had the opportunity to ask questions. I clarified any uncertainties to ensure that participants had a clear understanding. In Task 1, participants in the chatbot group rated their understanding of the task with an average score of 4.00, while those in the linked list and issue list group rated theirs at 4.40. In Task 2, participants in the chatbot and issue list group reported an average score of 4.40, whereas those from the linked list group reported theirs at 4.20. Finally, for Task 3, the confidence was reported with 3.90 for the chatbot, 4.20 for the link list, and 3.80 for the issue list group. This task involved participants identifying a solution to a bug rather than completing a specific task, resulting in lower scores.

3.5 Data Analysis

My experiment covered both quantitative and qualitative data analysis. In the quantitative analysis, I employed statistical tests to assess differences among the three test groups. The analysis specifically focused on the confidence ratings for each task, reflecting the participants' confidence in their provided answer, alongside the time taken to complete each task. For my analysis, I selected either the parametric one-way ANOVA test or the non-parametric Kruskal-Wallis test [29] based on the fulfillment of statistical assumptions. The first assumption for both tests was that observations were independent, which was fulfilled as the experiment was conducted with each participant individually and they were instructed not to share any insights with other participants. Additionally, participants were randomly assigned to each test group. The one-way ANOVA test assumed that the data followed a normal distribution, which I tested using the Shapiro-Wilk test [45]. Both tests further assumed equal variances in the three groups. This was tested with the Levene's test [33] if the data followed a normal distribution or with the Fligner Killeen test [17] otherwise. Both the one-way ANOVA and Kruskal-Wallis tests indicate a significant difference between the test groups but do not specify which group exactly. Therefore, a post-hoc analysis is required to determine the specific groups that differ. If a significant difference is identified, the Tukey test will be used following a one-way ANOVA, while the Dunn's test will be used subsequent to the Kruskal-Wallis test. Furthermore, I examined the number of participants who successfully completed the task within each group. All necessary calculations were done using functions from the SciPy Statistical Functions module [10].

Results from the chatbot group that used the tool version with LLM support were further analyzed using automatic metrics for text generation tasks. I examined the model's summary in comparison to the original content of the issue on which the task was based on, utilizing ROUGE, a commonly used metric for text summarization tasks [13, 66, 30, 68], and BERTscore [69], known for its higher correlation with human evaluation. Initially, I intended to use BARTscore [67] as well. However, the absence of a pip-installable version for Python makes it cumbersome to use. In addition, BARTscore requires fine-tuning and setup to outperform BERTscore. Using BERTscore alongside ROUGE was sufficient for my study. I analyzed the number of participants who used the provided issue links in addition to the tool's summary and how trustworthy they rated the tool's summary. Subsequently, I analyzed how the use of these issue links, affected participants' reported confidence in their answers. Moreover, I conducted open coding to examine prompt patterns and identify recurring themes. My tutor also conducted open coding to ensure completeness and correctness. Furthermore, I analyzed the length of the prompts utilized by participants and compared them across tasks and groups.

To quantify the user experience, I used the UEQ [31] and its official data analysis tool [58]. This tool provided descriptive statistics, including mean values and standard deviations and inferential statistics, specifically the t-test. Since the t-test is only applicable for comparing two groups, I performed pairwise comparisons among all pairs within the three test groups. For the qualitative evaluation, I manually analyzed answers to the following free-text questions included at the end of the survey: *"What did you like about the interaction with the provided tool ?"*, *"What did you not like about the interaction with the provided tool ?"* and *"Do you have additional suggestions to improve the tool's features ?"*. I, along with my tutor, employed open coding for this analysis.

4 Results

4.1 User Experience Evaluation

4.1.1 User Experience Questionnaire (UEQ)

As my experiment involved three test groups, I conducted a pairwise comparison of the user experience evaluation. This included comparisons between the chatbot and linked list group, the linked list and issue list group, and finally, the chatbot and issue list group. All three tools were perceived as mostly positive in all six categories, see Table 4.1. Only the *Novelty* of the linked list and the issue list tool were rated negatively. Ratings range from -3 to +3.

Chatbot vs. Issue List

Attractiveness (1.35/0.70:) The chatbot tool received higher ratings in this category, indicating that participants perceived more satisfaction while using the chatbot tool. However, the results fail to indicate a significant difference based on the commonly used alpha level of 0.05. Further analysis revealed that one participant perceived the tool differently from the majority of participants. While the average *Attractiveness* rating of the other nine participants was 1.61, one participant reported a rating of -1.00. Given that the sample size was limited to ten data points, this had a significant influence on the ratings. Analysis conducted without this record shows a significant difference between the chatbot and issue list groups, with a p-value of 0.0368.

Perspicuity (1.08/0.80): This minor difference suggests that both tools are user-friendly and intuitive. Developers are likely familiar with ITSs, as they are commonly employed in many software development projects. However, the participants were able to use the chatbot tool effectively with just a brief introduction. This indicates that the chatbot tool is user-friendly for newcomers.

Efficiency (1.88/0.95): A significant difference was observed in this category. The chatbot tool generates a summary of relevant issues, eliminating the need to manually skim through multiple issues to find the required information. This streamlines the process and makes

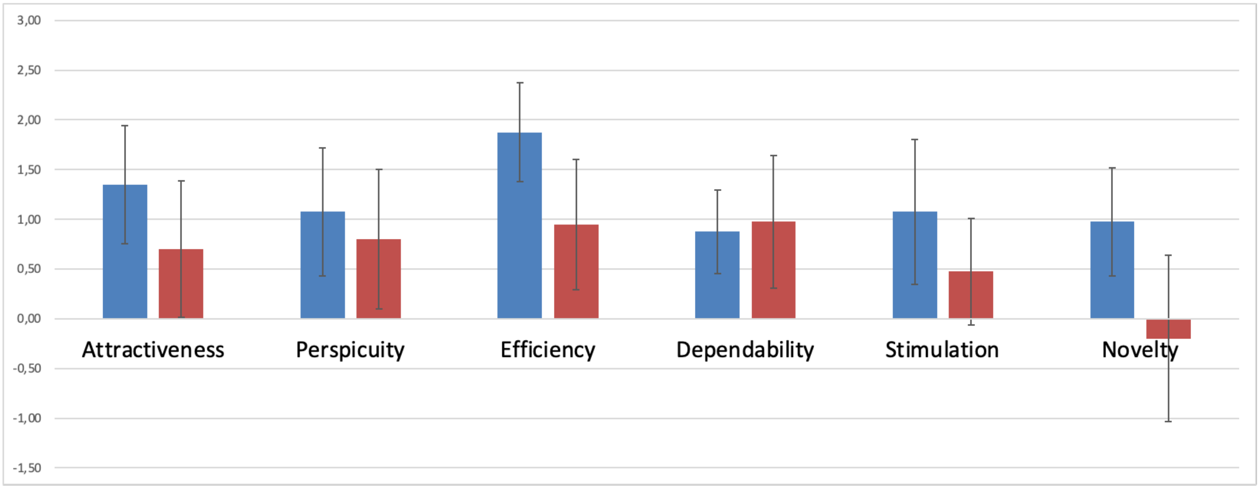


Figure 4.1: UEQ scale means of the six categories. Chatbot (blue) vs issue list (red)

the interaction more efficient. The chatbot tool received its highest score on the practicality scale, with a value of 2.1.

Dependability (0.88/0.98): The chatbot tool received slightly lower ratings compared to the ITSs from the issue list group. Participants from the chatbot group rated the *Security* and *Predictability* with lower values. One possible reason for this is the non-deterministic nature of LLM-generated text, which can lead to unpredictable outcomes. Additionally, security concerns might have contributed to the lower ratings, as users may be concerned about how their data is handled.

Stimulation (1.08/0.48): The chatbot tool was rated higher, although the difference was not statistically significant. Similar to the findings in the *Attractiveness* category, an outlier skewed the overall score. Nine participants rated the *Stimulation* with an average score of 1.36, while one participant rated it with a score of -1.50. A higher rating for the chatbot tool compared to the issue list tool is expected, considering users often find novel tools more engaging and exciting.

Novelty (0.98/-0.20): The chatbot tool was rated significantly higher than the issue list tool. This is in line with expectations as ITSs are a well-established tool in software engineering. The chatbot tool introduces a novel method of interacting with knowledge from ITSs, resulting in a higher rating in this category.

In general, the chatbot tool received higher ratings across five categories, although not always significantly higher. For a visual representation, please refer to Figure 4.1. It is important to note that the sample size of 10 data points is not representative and outliers heavily influenced these results. The standard deviation in all six categories ranged from 0.68-1.18. Therefore, a broader analysis with a larger sample size is necessary to draw robust conclusions.

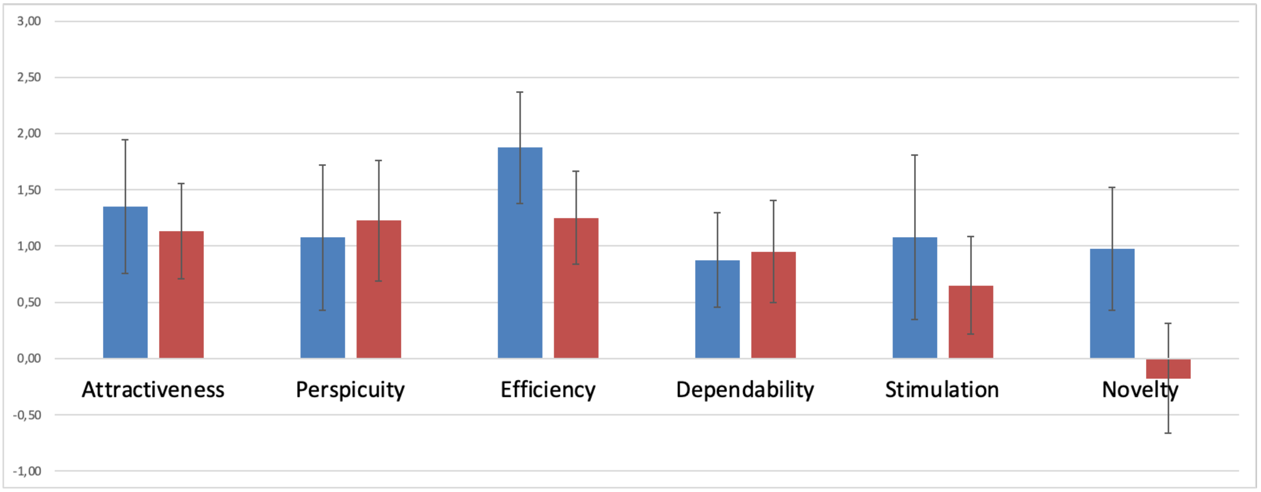


Figure 4.2: UEQ scale means of the six categories. Chatbot (blue) vs link list (red)

Chatbot vs. Link List

Attractiveness (1.35/1.13:) Both tools received a similar score, but the chatbot tool has a slight advantage. They share the same UI, so a similar score is expected. The streaming of the response generated by the LLM likely made a difference.

Perspicuity (1.08/1.23): The link list tool was rated slightly higher. While the tool supported natural language input, some participants utilized shorter keywords for their prompts. This potentially reduces the complexity of formulating a good prompt.

Efficiency (1.88/1.25): The chatbot tool summarizes issue content for the user, thus eliminating the need to manually click on issues to find relevant information. This explains the higher efficiency rating the chatbot tool received. However, with a p-value of 0.0747, it fails to meet the threshold for statistical significance.

Dependability (0.88/0.95): In this category, both tools received a similar rating. As previously discussed, the chatbot tool's slightly lower rating may be attributed to its inclusion of a language model.

Stimulation (1.08/0.65): The chatbot tool was rated higher than the link list tool. This higher rating is likely attributed to the additional interaction with an LLM, which improved the score.

Novelty (0.98/-0.18): Once again, this higher score can be attributed to the new interaction with knowledge from ITSs that the chatbot tool provides. While the link list tool centralized information from ITSs, participants were still required to skim through issues manually.

The difference between the two tools is marginal, but the chatbot tool performed slightly better, as illustrated in Figure 4.2. Only the *Novelty* category demonstrated a significant difference. This is not unexpected, given that both tools appear and function similarly. However, the inclusion of the summary generated by the LLM impacted some categories.

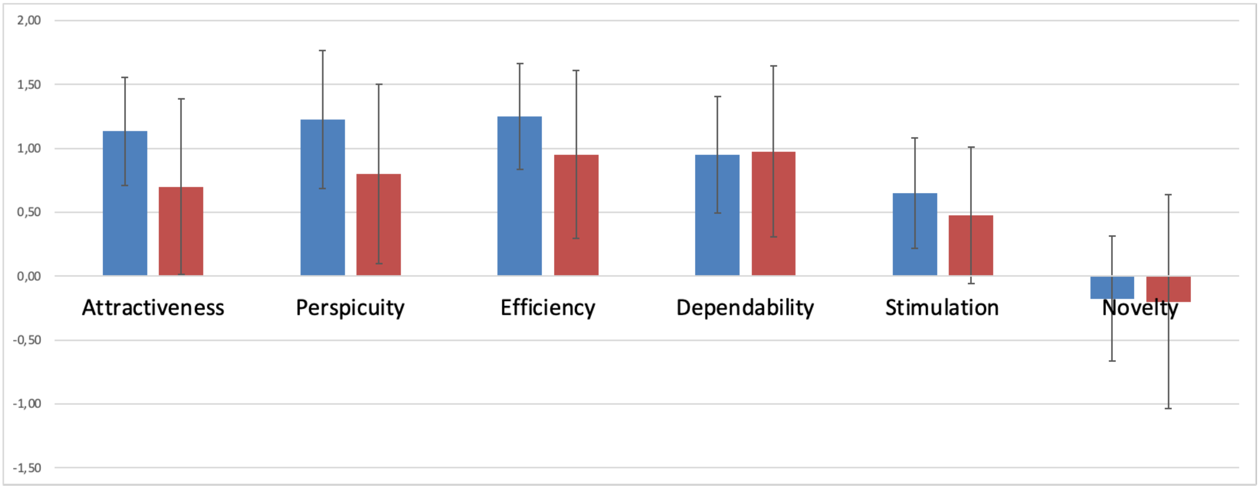


Figure 4.3: UEQ scale means of the six categories. Link list (blue) vs issue list (red)

Link List vs. Issue List

Attractiveness (1.13/0.70:) The link list tool received higher scores than the issue list tool, indicating that participants found it more enjoyable to use. This result is consistent with the findings of the chatbot tool, which also received higher scores.

Perspicuity (1.23/0.80): As with the chatbot tool, these results are promising, given that participants were using the tool for the first time. The link list tool offered a natural language search, which was a new addition. However, it simplified searching for relevant issues, resulting in the highest score among the test tools.

Efficiency (1.25/0.95): The link list tool received higher ratings for the *Efficiency*. It provides a list of relevant issues sorted by relevance in a clear and simple format. The simple UI allows users to focus on the primary information while reducing the potential for distraction.

Dependability (0.95/0.98): Both tools received nearly identical ratings from the participants. This indicates that the tools provide similar functionality that is predictable and meets expectations.

Stimulation (0.65/0.48): The link list tool received a slightly higher rating than the issue list tool. Support for natural language queries may be the reason for this difference.

Novelty (-0.18 /-0.20): Both tools received a negative rating in this category. The link list tool is capable of searching multiple repositories and presents the information in a unified UI. However, the end result looks similar to the ITSs used by the issue list group.

The link list tool improved the user experience in most categories, although the improvements were not significant, as shown in Figure 4.3. However, this is expected since the link list tool does not introduce many new features. The ability to search across multiple ITSs may not have been fully appreciated by participants, as each task required information from a single repository.

Table 4.1: Pairwise comparison of user experience between the three test groups. Ratings of the six UEQ categories (mean and standard deviation), including the t-test's p-value. CB = chatbot group, LL = linked list group, IL = issue list group

Group	Attractiveness	Perspicuity	Efficiency	Dependability	Stimulation	Novelty
CB	1.35 (0.96)	1.08 (1.04)	1.88 (0.80)	0.88 (0.68)	1.08 (1.18)	0.98 (0.88)
IL	0.70 (1.11)	0.80 (1.14)	0.95 (1.06)	0.98 (1.08)	0.48 (0.86)	-0.20 (1.35)
p-value	0.1789	0.5794	**0.0420**	0.8072	0.2117	**0.0350**
CB	1.35 (0.96)	1.08 (1.04)	1.88 (0.80)	0.88 (0.68)	1.08 (1.18)	0.98 (0.88)
LL	1.13 (0.68)	1.23 (0.87)	1.25 (0.67)	0.95 (0.73)	0.65 (0.70)	-0.18 (0.79)
p-value	0.5683	0.7308	0.0747	0.8153	0.3427	**0.0065**
LL	1.13 (0.68)	1.23 (0.87)	1.25 (0.67)	0.95 (0.73)	0.65 (0.70)	-0.18 (0.79)
IL	0.70 (1.11)	0.80 (1.14)	0.95 (1.06)	0.98 (1.08)	0.48 (0.86)	-0.20 (1.35)
p-value	0.3093	0.3606	0.4601	0.9524	0.6234	0.9603

To conclude the pairwise comparison: The chatbot tool performed the best in *Attractiveness*, *Efficiency*, *Stimulation*, and *Novelty*. However, in some cases, the tool failed to achieve statistically significant differences. Further analysis revealed that one participant in the chatbot group seemed to perceive the tool differently. Since each group consisted of 10 participants, the outlier had a significant impact on the results. Nevertheless, the results indicate that participants appreciated the novel approach to knowledge retrieval from ITSs. In addition, the high score in *Efficiency* is promising. The link list tool performed best in *Perspicuity*, while the issue list tool performed the best in *Dependability*.

4.1.2 User Experience Comments

Chatbot Group

Participants (6 mentions) particularly highlighted the ease of use of the chatbot tool. Additionally, they praised the tool for its fast responses (3) and for delivering comprehensible results (1). Furthermore, its similarity to ChatGPT (2) and the provision of references (2) were appreciated by the participants. The UI (2) and the tool's ability to incorporate knowledge from various projects (1) were also highly valued by the participants.

On the other hand, participants noted the absence of prompting assistance (3) and the ability to ask follow-up questions (2). Some references were deemed irrelevant to the given question (2). Moreover, some participants requested a relevance score for the provided issue links (1) and a mode where the LLM would give a response based on its general knowledge when no issues were found (1). It was also suggested that the issue content could be summarized into problem-solution statements to improve the chatbot responses (1). Participants requested brief summaries of the provided issue links (2). Finally, it was noted that the code highlighting (1) could be improved and that the input field should have a fixed position at the bottom (1), similar to ChatGPT's UI.

Link List Group

Similar to the chatbot tool, participants praised the link list tool for its ease of use (6) and its fast responses (3). Furthermore, participants valued the compactness of the results (1), the natural language interaction (1), and the clarity of the answers provided (1). It was also mentioned that the results are based on a predefined set of issues (1).

As observed in the chatbot group, participants asked for short summaries of the provided issue links (2). A relevance score for each issue link (4) and prompting assistance (3) were particularly missed by participants. It was noted that the tool should always provide feedback, particularly in cases where no relevant issues are found (1). Additionally, participants reported that the UI could be improved (1) and that documentation about the tool should be integrated (1).

Issue List Group

Participants praised the ease of use (5), the familiarity (2), and the speed of the tool (2). They also acknowledged that the tool performed as expected (3).

However, participants noted the absence of prompting assistance (3) and natural language support (1). It was stated that the information was solely based on issues (1) and that incorporating general documentation could be beneficial (1). Additionally, participants mentioned that the tool only supports short queries (1) and that documentation of the tool is lacking (1).

To answer **RQ1**, participants in the experiment preferred using the chatbot tool to retrieve knowledge from ITSs. The chatbot introduced an innovative and easy-to-use tool that enhanced the process by eliminating the need for manual issue searching, resulting in a more enjoyable and efficient experience. It outperformed the other tools in *Attractiveness*, *Efficiency*, *Stimulation*, and *Novelty*. The link list tool was considered a minor improvement over the classic ITSs. It introduced new features like a natural language search and a new UI. The classic ITSs were rated highest in terms of *Dependability* and *Efficiency* by the participants.

4.2 Time Efficiency

I measured the time each participant took to complete a task and will report the average times along with the standard deviations in a minutes:seconds format. If a participant was unable to provide an answer, their recorded time was excluded from the analysis. The average completion time for Task 1 was 4:18 (1:17) for the chatbot, 3:27 (1:37) for the link list, and 4:00 (1:14) for the issue list group. After data processing, five recorded times remained for both the chatbot and link list groups and seven for the issue list group. For Task 2, the chatbot group's average time was 2:32 (0:57), the link list group's was 3:14 (1:13), and the issue list group's was 3:47 (1:12). Following data processing, eight recorded times were left for the chatbot group, seven for the linked list group, and five for the issue list group. For Task 3, the chatbot group took around 3:55 (0:55), the link list group around 2:51 (1:34), and the issue list group around 2:48 (1:10). After data processing, seven recorded times were left for the chatbot group, six for the linked list group, and eight for the issue list group. All times are listed in Table 4.2.

One important consideration is that participants in the chatbot group must wait for the LLM to finish generating its summary, which may take a few seconds. This delay contributed to a marginal increase in the average response times. Participants using the chatbot formulated the longest search prompts, as detailed in Section 4.4. These longer prompts further further increased the time required. However, in Task 2, the time was still lower compared to the other groups. Participants in this task were able to copy the original query, instruct the chatbot to translate it, and receive the translated query as part of the summary. This represented a benefit of using the chatbot: its ability to directly apply the provided context to address the given question. However, participants had to wait for the chatbot to generate its response, unlike traditional methods where they navigate through multiple issues to find the relevant information. Whether this is beneficial or not varies depending on the task at hand. Additionally, some participants utilized the provided issue links to validate the generated summary, which had an impact on the recorded times. Overall, participants in the link list group achieved the fastest times. The link list tool simplified the search by enabling them to input more detailed prompts, which likely made it easier to find the relevant issues. Furthermore, during the experiment, I observed that several participants missed important information while skimming through issues or the issue list provided by the ITSs. The keyword-based search in the ITSs proved to be inefficient when multiple issues matched the search criteria. This resulted in a lengthy list of issues for participants to check, leading to longer response times.

Furthermore, I employed statistical tests to assess the significance of differences in the time taken by participants across the three test groups. I utilized the Shapiro-Wilk test to

Table 4.2: Measured mean + (standard deviation) in minutes:seconds for each task and each group, including the statistical significance (p-value)

Group	Task 1	Task 2	Task 3
Chatbot	4:18 (1:17)	2:23 (1:00)	3:54 (0:51)
Link List	3:27 (1:37)	3:02 (1:15)	3:09 (1:38)
Issue List	4:00 (1:14)	3:48 (1:05)	2:50 (1:06)
p-value	0.618	0.079	0.210

evaluate the normality of the data across all groups. For Task 1, the data exhibited normal distribution for all three groups. Additionally, the Levene's test confirmed equal variances, thus enabling the use of the one-way ANOVA test. However, the one-way ANOVA test reported no significant difference across the groups. Similarly, for Tasks 2 and 3, all necessary assumptions were met to conduct the one-way ANOVA test. The results, detailed in Table 4.2, did not reveal any statistically significant differences among the groups. Notably, for Task 2 the reported p-value was 0.078. Although this value suggests a potential difference in the time taken, it falls short of the widely accepted significance threshold of 0.05.

Answering **RQ2**, the average response time for the chatbot group was slightly slower compared to the other groups. However, it is important to note that participants had to wait for the LLM to generate the summary. When the LLM was able to directly incorporate the retrieved knowledge into the response, it resulted in significantly faster response times for participants. Participants using the issue list tool achieved faster response times, as the natural language input allowed them to identify relevant issues more efficiently. Conversely, the keyword-based search in the ITSs proved inefficient when numerous issues matched the search criteria, leading to an overload of displayed issues and making it difficult to identify the correct one.

4.3 Confidence On Provided Answers

Participants were asked to rate their confidence in the answers they provided using a Likert scale ranging from one to five. I performed statistical analysis to compare the three test groups. This involved the Shapiro-Wilk test to evaluate the normality of the data distribution in each group. For Task 1, the data within the chatbot group deviated from normal distribution, which required the use of the Kruskal-Wallis test. To verify the assumption of homogeneity of variances across the groups, I applied the Fligner test, which confirmed equal variances with a p-value of 0.72. The Kruskal-Wallis test revealed no significant differences between the groups. The same process was repeated for Tasks 2 and 3. In both tasks, the

Table 4.3: Analysis of the confidence ratings (measured mean + standard deviation) in their answers rated by participants on a 5-point scale. Including the p-value of the Kruskal-Wallis test.

Group	Task 1	Task 2	Task 3
Chatbot	2.83 (0.98)	4.10 (0.74)	4.22 (0.67)
Link List	4.20 (0.84)	3.25 (1.28)	3.57 (1.27)
Issue List	3.00 (1.15)	2.83 (1.15)	3.60 (0.84)
p-value	0.091	0.076	0.282

Kruskal-Wallis test again showed no significant differences. However, with a p-value of 0.076, it narrowly missed the threshold of 0.05 for significance in Task 2. All results, along with the p-values, are reported in Table 4.3.

For Task 1, the average confidence was 2.83 (0.98) for the chatbot, 4.20 (0.84) for the link list, and 3.00 (1.15) for the issue list group. If a participant was unable to provide an answer within the time limit or ended the task early, their response was excluded from the analysis. After data processing, six answers were left for the chatbot, five for the linked list, and seven for the issue list group. Out of these, two participants from the chatbot and linked list group were able to find the correct answer. Participants from the issue list group gave four correct answers. To identify the correct issue for Task 1, participants had to use specific keywords. This proved problematic and resulted in participants not finding the relevant issue. For Task 2, confidence was rated 4.10 (0.74) in the chatbot, 3.25 (1.28) in the linked list, and 2.83 (1.15) in the issue list group. Following data processing, ten records remained in the chatbot, eight in the linked list, and six in the issue list group. Participants performed better in this task, providing nine correct answers in the chatbot, six in the linked list, and four in the issue list group. Participants in the chatbot group benefited from the ability to copy and instruct the chatbot to translate the original query, enhancing their ability to provide the correct answer. This is evident in the number of participants who were able to provide the correct answer. However, some participants reported that they could not rate their confidence with the highest value of five. They would normally execute a query to confirm that it is working as expected and to check for syntax errors. For Task 3, participants rated their confidence with 4.22 (0.67) in the chatbot, 3.57 (1.27) in the linked list, and 3.60 (0.84) in the issue list group. After data processing, nine answers were left for the chatbot, seven for the link list, and ten for the issue list group. Task 3 appeared to be the easiest for participants as nine correct answers were given in the chatbot and issue list group, while participants from the linked list group gave six correct answers. This finding is unexpected because the participants rated their confidence in understanding this task the lowest prior to undertaking it. Overall, the chatbot group reported the highest confidence levels, except for Task 1.

To answer **RQ3**, participants who used the chatbot tool were surprisingly the most confident in their answers. This outcome was unexpected, as one might anticipate that conducting your own search would be the most reliable source of information. However, participants also verified the summary using the provided issue links. This suggests that the combination of a concise summary and a verification step proved to be a reliable source of information. Participants in the issue list group reported the lowest confidence values.

4.4 Analysis Of Prompt Patterns

Participants in the chatbot group primarily utilized natural language, framing their inputs as questions directed towards the chatbot. This aligns with the expected interaction model with chatbots. The majority of participants in the chatbot group used wordings such as *"How can I ..."*, *"Can you help ..."* and *"translate ..."*. In Task 2, which involved translating a SQL query, six out of ten participants directly copied the query from the task for their prompt. Similarly in Task 3, eight out of ten participants copied the query from the task instructions. Generally, this approach resulted in positive outcomes since the task instructions served as an effective foundation for identifying relevant issues.

I have categorized the prompts into distinct patterns, as detailed in Table 4.4. The variation in prompt patterns between the chatbot users and those in the issue list group can be attributed to the required input methods: natural language for chatbots and keyword-centric prompts for the GitLab and GitHub ITS. The participants in the linked list group represent an intermediary case. Some participants utilized natural language, in line with the tool's capabilities and instructions, while others chose to use keywords or brief prompts. The majority of prompts from participants interacting with the chatbot consisted of a single sentence, although a few extended up to three sentences in length. The results, shown in Table 4.5, highlight how the prompt length differentiates substantially across the three test groups. The average character counts for the chatbot group were 128.70 (87.19) in Task 1, 145.20 (98.54) in Task 2, and 196.50 (67.17) in Task 3. The link list group's average prompt

Table 4.4: Prompt patterns used by participants

Prompt pattern	Chatbot	Linked List	Issue List
Natural language query	29	14	0
Keyword Query	1	15	28
Included library name	21	17	2
Copied query from task	14	5	0

Table 4.5: Analysis of the prompt length (mean + standard deviation) in number of characters.

Group	Task 1	Task 2	Task 3
Chatbot	128.70 (87.19)	145.20 (98.54)	196.50 (67.17)
Link List	47.60 (22.18)	42.80 (24.30)	55.50 (45.76)
Issue List	17.44 (24.08)	19.67 (12.09)	10.30 (8.21)

lengths were shorter, with 47.60 (22.18) characters in Task 1, 42.80 (24.30) in Task 2, and 55.50 (45.76) in Task 3. Participants in the issue list group utilized the shortest prompts, averaging 17.44 (24.08) characters in Task 1, 19.67 (12.09) in Task 2, and 10.30 (8.21) in Task 3. While it is understandable that the chatbot group used significantly longer prompts than the issue list group, it is surprising how noticeably shorter the prompts from the link list group are compared to the chatbot group. The links list tool featured a natural language search input but lacked a conversation with a chatbot. The conversation mode of the chatbot tool may have encouraged participants to use more words and prompts that resemble a dialog more closely. Although the prompts differed between each group, certain question-specific keywords were present in each prompt.

Some participants had to refine their prompts to obtain the necessary information to solve the given task. These prior attempts were not included in the analysis. However, this also shows that prompting is not trivial. While most participants successfully solved the task with a single prompt, others had to refine their prompt multiple times. Additionally, formulating the right prompt proved time-consuming for some participants.

In response to **RQ4**, practitioners primarily utilize natural language prompts when interacting with a chatbot. As a result, they tend to use longer sentences and formulate their prompts as questions directed towards the chatbot. In contrast, practitioners using a classic ITS typically employ short, keyword-based prompts. This approach is expected, as ITSs include a keyword-based search. However, the chatbot group's prompts were significantly longer than those of the link list group, even though the latter also supported a natural language search. The chatbot's conversational mode likely encouraged participants to utilize more verbose and dialog-like prompts.

4.5 Necessity Of Links To Issues

Participants in the chatbot group were asked whether they utilized the provided issue links or exclusively relied on the generated summary to solve the given task. If they utilized the issue links, they were required to rate the necessity of the issue links in addition to the summary

Table 4.6: Analysis of mean values + standard deviation on the confidence ratings per task, grouped by whether participants used the provided issue links in addition to the summary or not

Task	Issue Link Clicked	Confidence in Correctness	Links Necessity	Issue Link Not Clicked	Confidence in Correctness
Task 1	6x	2.33 (1.21)	2.83 (0.75)	4x	2.5 (1.00)
Task 2	5x	4.40 (0.55)	2.60 (0.89)	5x	3.80 (0.84)
Task 3	7x	4.00 (0.58)	3.68 (1.21)	3x	3.67 (2.31)

on a Likert scale from one to five. In Task 1, six participants clicked on one of the provided issue links. They rated the necessity of the links in addition to the tool's summary with 2.83 (0.75). In Task 2, five participants utilized the issue links, rating the necessity with 2.60 (0.89). Seven participants utilized the issue links in Task 3, rating the necessity with 3.68 (1.21). This higher score is likely justified by the design of the task. Participants were required to identify that the presented problem was caused by a bug in the library. This seemed to confuse some participants, so they used the issue links for a clearer understanding and to validate the summary by the chatbot. Furthermore, I observed that several participants accessed these links either after answering the question (completing the task) or when they were asked in the survey if they had used the links in addition to the summary. They referred to the issue links to confirm the accuracy of the summary by briefly checking it against the content of the issues. This is reflected in the necessity ratings, which were not particularly high, suggesting that the tasks could have been completed without consulting these links.

Table 4.6 shows the confidence levels reported by participants, grouped by whether they used the issue links in addition to the summary. The mean values for Task 1 are similar, suggesting that the issue links had no significant impact. In Tasks 2 and 3, average confidence levels marginally increased. Although this analysis is based on a small sample size, it indicates a trend that the issue increased the confidence of participants, although only slightly. The overall level of trust participants had in the generated summary was rated with a mean value and standard deviation of 2.44 (1.13). This rating suggests that participants do not entirely trust the text generated by LLMs, even when it is based on accessible knowledge.

Answering **RQ5**, practitioners rated the necessity of the issue links for each task with an average score of (2.83/2.60/3.68). This indicates that the links were not essential for answering the questions. Instead, they served as a verification mechanism in case of doubt. This is also reflected in practitioners' trust in text generated by the LLM. Furthermore, the results indicate that practitioners who utilized the issue links reported a slightly higher confidence rating.

4.6 Information Needs

I evaluated the generated chatbot summaries using automatic metrics, including ROUGE-1, ROUGE-2, ROUGE-L, and BERTscore. Participants were required to copy and paste the generated summary they used to solve the specified task into the questionnaire. The evaluation process involved comparing each response to the original content of the issue on which the task was based. Summaries were excluded from the analysis if they were not based on the issue or if a participant did not provide any answer. For each task, I report the mean value of the summary scores. For Task 1, three summaries were analyzed, while for Tasks 2 and 3, nine summaries were analyzed. Table 4.7 summarizes the Precision, Recall, and F1-score for the four metrics across each task. The ROUGE metrics, particularly ROUGE-1 and ROUGE-2, measure the overlap of n-grams between the candidate text and the reference sentences. Meanwhile, ROUGE-L assesses the longest common subsequence [34].

Table 4.8 shows the average length of the generated summaries for each task in comparison to the length of the issue the task was based on. Notably, the summary for Task 1 was longer than the issue itself. The issues comprised only a brief title, description, and a few short comments. Conversely, the summaries for Tasks 2 and 3 were shorter than the reference issue. If an issue is relatively brief, it is possible that the summary may be longer because the chatbot supplements it with additional text. However, this does not affect the quality of the response.

In almost all cases, the values of the ROUGE metrics improve from Task 1 to Task 3, indicating that the summaries for Task 3 show better overlap with the reference issues. The ROUGE-2 scores are consistently lower than the ROUGE-1 and ROUGE-L scores across all tasks. This indicates that the bi-gram overlap is lower compared to single words or longest common sequences, which are captured by ROUGE-1 and ROUGE-L. The BERTscore values are consistently higher across all tasks compared to the ROUGE metrics. This is likely because BERTscore utilized contextual embeddings to calculate token similarity and importance weighting to assign more weight to rare words.

Overall, the results demonstrate that the summaries by the chatbot are consistently rated highly by the BERTscore. This suggests that the summaries effectively cover the main concepts or topics, although they may not always replicate the exact phrasing or specific language details as the original issues. As a result, traditional metrics such as ROUGE scores produce lower values. The tool is primarily designed to answer specific questions rather than to provide a comprehensive summary of all relevant issues. Consequently, it may omit context not directly related to the question, affecting the scores reported by the metrics. Although BERTscore evaluates semantic meaning, it cannot account for the question's context, thus influencing its evaluation.

Table 4.7: Precision, Recall, and F1-score of the generated summary in contrast to the reference issue. Accumulated mean and standard deviation scores of all summaries for each task. Using ROUGE-1, ROUGE-2, ROUGE-L and BERTscore.

Metric	Task	Precision	Recall	F1-score
Rouge-1	Task 1	0.33 (0.09)	0.61 (0.09)	0.41 (0.05)
	Task 2	0.42 (0.09)	0.23 (0.05)	0.29 (0.06)
	Task 3	0.67 (0.08)	0.53 (0.06)	0.58 (0.04)
Rouge-2	Task 1	0.08 (0.03)	0.15 (0.03)	0.10 (0.03)
	Task 2	0.10 (0.05)	0.05 (0.02)	0.07 (0.03)
	Task 3	0.33 (0.05)	0.26 (0.04)	0.29 (0.04)
Rouge-L	Task 1	0.14 (0.05)	0.26 (0.03)	0.18 (0.04)
	Task 2	0.25 (0.06)	0.13 (0.03)	0.17 (0.03)
	Task 3	0.34 (0.05)	0.28 (0.07)	0.30 (0.06)
BERTscore	Task 1	0.82 (0.04)	0.87 (0.01)	0.84 (0.03)
	Task 2	0.84 (0.02)	0.82 (0.01)	0.83 (0.01)
	Task 3	0.90 (0.01)	0.88 (0.01)	0.89 (0.01)

Table 4.8: Analysis of the summary length (in characters) in contrast to the issue length the task was based on. Summary length is shown as the mean + standard deviation accumulated for each task.

Task	Summaries Analyzed	Summary Length	Original Issue Length
Task 1	3x	940.33 (403.54)	380.00
Task 2	9x	518.11 (115.29)	882.00
Task 3	9x	735.67 (147.32)	876.00

To address **RQ6**, the results from the ROUGE and BERTscore metrics indicate that a brief summary that captures the essence of the issues is sufficient for making informed decisions. Although the ROUGE scores were relatively low due to the summary not necessarily capturing similar phrasings of the issues, this did not affect the ability to extract the correct solution. For example, the ROUGE-2 score for Task 2 reported a precision of 0.10, while nine out of ten participants provided the correct solution. This demonstrates that the summary captured the necessary information. The BERTscore reported higher scores, as it captures semantics more accurately and is more closely aligned with human evaluation. The findings indicate that the length of a summary does not necessarily correspond with its quality.

5 Discussion

5.1 Concise Presentation of Information is Important

During the experiment, I observed that participants frequently overlooked essential information while navigating through issues. The presence of excessive noise, such as potentially unrelated comments and discussions, made it more likely that important information would be overlooked. ITSs typically provide a list of issues with a keyword-based search function. However, identifying specific issues can be challenging and keyword-based searches have limited effectiveness in filtering relevant information. In contrast, participants of the chatbot group did not experience this limitation. The generated summary provided a compact version of relevant issues, effectively removing unnecessary noise. Furthermore, results of the user experience evaluation revealed that the chatbot achieved the highest scores both in *Attractiveness* and *Stimulation*. Additionally, the ease of use and the tool's fast responses were frequently mentioned as positive aspects of the chatbot. This observation is consistent with previous research indicating that developers value forums like Stack Overflow for their question-answer format [40] and prefer to receive information in smaller chunks [47]. However, this study extends those findings.

Developers not only prefer smaller segments of information but encounter difficulties when confronted with an excessive amount of information [6]. Presenting information in smaller segments can mitigate the risk of overlooking important details. Furthermore, this approach can increase productivity since developers do not need to process excessive information to find essential details. However, it is essential to be cautious, as this approach carries the potential risk of omitting relevant information that could be beneficial. The chatbot tool addresses this concern by providing issue links, allowing users to access more detailed information when necessary. Therefore, a balanced approach is necessary to manage these aspects effectively. Additionally, chatbots capable of understanding and applying code examples to specific questions can be beneficial. Code and usage examples are essential for aiding comprehension and preventing errors [57, 41].

5.2 Developers do not trust LLMs yet

The experiment revealed that developers are hesitant to trust text generated by LLMs, particularly when this information is needed to assist users. This skepticism is mainly due to the black-box and non-deterministic nature of LLMs. Users are unable to see or comprehend the process behind the generation of answers. Furthermore, LLMs are known to produce hallucinations, further reducing trust in their output [25]. This concern is supported by the behavior of the participants, who frequently clicked on the provided issue links after providing an answer to the current task. This aligns with the findings by Sivaraman et al. [55]. In their study of the acceptance of AI-based support tools, the authors concluded that AI support was most accepted as an additional source of information, not as the primary source. The actions of the participants indicate a desire to confirm whether the model accurately utilized the provided information. However, to minimize the risk of generating inaccurate responses, I specifically designed the custom prompt to instruct the model to utilize the knowledge it has access to and refrain from generating responses in the absence of that knowledge. For the instructions given to the LLM, see Listing 3.2. Participants might not have been fully aware of this approach and probably compared the tool to ChatGPT, where such inaccuracies are more common. I informed participants that the tool uses information from issues, which were displayed as issue links below each response. However, as each experiment had a time limit, no detailed explanation of how the tool works was provided. This lack of information may have affected the participants' trust in the tool. According to Zhang et al., displaying confidence scores of the generated answer can increase trust in an LLM model, but only if the scores are high. Otherwise, it can have the opposite effect [71].

Interestingly, despite relying on the tool's summary, participants were confident in the correctness of their answers. Confidence levels were the highest in the chatbot group, with Task 1 being the exception. However, as previously noted, some participants cross-referenced the issue links to validate the tool's summaries. This reveals a crucial insight: although the tool often provided accurate information and participants were confident in their answers, there remained a hesitancy to fully trust the LLM-generated text. This hints at possible challenges for future research: enhancing models' ability to use context accurately, reducing false information (hallucinations), and, more importantly, building greater trust with users.

As some participants noted, it may be beneficial for the tool to be able to utilize the general knowledge of LLMs. However, it is important to minimize the risk of hallucinations and incorrect information by following strict instructions given by the prompt. The addition of an extra mode or switch could signal to users that the information provided may not be entirely accurate and that they should, therefore, exercise caution.

5.3 Metrics to Benchmark LLMs

In Section 4.6, I evaluated the summaries generated by the chatbot using metrics including ROUGE-1, ROUGE-2, ROUGE-L, and BERTscore. The results show that traditional metrics are not suitable for evaluating the quality of LLM-generated summaries. The ROUGE scores were consistently low, despite the summaries being sufficient for the given task, as demonstrated by the number of correct answers. In contrast, the BERTscore reported higher scores, as it is capable of capturing semantic meaning without relying on finding similar phrasings. Furthermore, it correlates more with human evaluation [69]. Essentially, these metrics evaluate the summaries by comparing them against reference texts. In my study, the summary generated by the chatbot was based on relevant issues and their content, retrieved from a database using a similarity function. However, it is possible that the issues selected by the similarity function contained more information than necessary to answer the user's question. This has influenced the reported scores, particularly the ROUGE scores. Kumar et al. [30] demonstrated that increasing the summary length correlates with a better ROUGE score. This occurs because more extended summaries tend to resemble the reference text closely. However, this approach does not effectively evaluate the quality of a summary. A concise summary can effectively convey the key points of the reference texts. LLM demonstrate exceptional text comprehension abilities [9]. Therefore, they may rightfully choose to 'ignore' content that is not relevant to the given question. Current metrics for text summarization, however, fail to capture this aspect of question-based content selection. BERTscore evaluates semantic meaning and effectively prioritizes essential context. However, it lacks awareness of the specific question being addressed and the objective of providing a relevant answer.

Ultimately, I was looking for a metric specifically designed to evaluate text summarization in a dialog context. Such a metric should not only assess the quality of the summary in relation to the reference texts but also evaluate how effectively the summary addresses the given question.

6 Threats to Validity

6.1 Construct Validity

Many participants were overwhelmed by Task 1. This task required the use of React, with which most of the participants had no previous experience. Nonetheless, this scenario is typical at the Data Operation Center at EuXFEL, where support staff are often expected to assist users with problems in areas where they have limited expertise. Before each task, I asked participants if they understood the task and assisted them if further clarification was necessary.

Some participants mentioned that they would typically look for solutions on Stack Overflow or similar platforms, which could indeed contain the necessary information. However, I selected issues that users had raised, ensuring that the problems were addressed and resolved within actual issue threads.

The similarity function used to retrieve the relevant issues was dependent on specific keywords. This was particularly evident in repositories with many issues. However, ITSs encounter this challenge to the same extent, if not greater.

6.2 Internal Validity

The tool was developed and evaluated exclusively using the OpenAI models, specifically GPT-4. Employing different models could potentially influence the quality of answers, leading to perceived improvements and a better overall user experience. The thesis focused on the development of a tool that leverages LLMs to enhance the knowledge retrieval process. Although some effort was dedicated to prompt engineering, it wasn't the primary focus of this thesis. However, allocating additional attention to prompt engineering has the potential to produce improved results.

The selected questions in the experiment could influence the validity of the findings. The tool was evaluated using three tasks to keep the experiment brief and ensure participant engagement in the survey.

The search input field of GitHub utilizes reserved keywords to provide additional functionality. The `'NOT'` keyword is used to exclude results containing a specific word [18]. This feature proved problematic for some participants in the issue list group during Task 2, which required them to translate a `'NOT IN'` SQL query. Specifically, when the command `'not in'` was entered into the GitHub search input field, it succeeded, but `'NOT IN'` failed.

The chatbot tool lacked support for follow-up questions that could leverage context from previous interactions, primarily due to concerns about context length. Consequently, the interaction differed from that of ChatGPT, assuming participants were familiar with it. Nevertheless, this limitation did not significantly impact the overall experience, as most participants were able to obtain the correct information using a single prompt.

Additionally, most participants were familiar with the QWERTY keyboard layout. However, for the experiment, a keyboard with the QWERTZ layout was provided, which caused some inconveniences for participants. I assisted participants who were looking for a particular shortcut or key. Nevertheless, this issue can be mitigated because it was a common issue for all participants.

6.3 External Validity

The research was conducted primarily at EuXFEL and involved a relatively small sample size. Consequently, the sample may not be highly representative of the overall population. In order to derive conclusions that are more broadly applicable, it is necessary to conduct a more extensive evaluation.

The majority of participants (24 out of 30) were from EuXFEL. Therefore, the experience with LLMs and ITSs might be limited to the technologies employed at EuXFEL. To mitigate this, I included participants from different groups at EuXFEL, like Data Analysis, Software Development, Electronics, and IT Infrastructure. Furthermore, the participants were distributed as evenly as possible across the tool groups in order to reduce homogeneity within a group.

The number of issues in an ITS varies depending on the project. The repository of the react-phone-number-input library contained numerous issues, although only one was relevant for Task 1. In other repositories, multiple issues may contain the necessary information. Three issues from two repositories have been included to account for this variation.

6.4 Conclusion Validity

Inferential statistics were employed on relatively small sample sizes. Furthermore, I also excluded data points when participants did not provide any answer or for similar reasons. This resulted in uneven group sizes, reducing the statistical significance of the tests applied in this study. A larger sample size may lead to different results.

The selection of relatively simple tasks may limit the findings of the study. The inclusion of more complex tasks that require information from multiple issues and potentially involve numerous ITSs could result in more meaningful insights. However, I opted for more straightforward tasks to reduce cognitive load and the risk of participants being unable to solve the tasks.

7 Conclusion

In this study, I conducted a controlled experiment on knowledge retrieval with the assistance of LLMs from multiple ITSs. The experiment included three test groups and a total of 30 participants. For the experiment, I developed a chatbot tool that fetches issues from various ITSs, processes the issues' information, and then stores it in a database. The participants in the experiment interacted with this tool in natural language and received either a summary (group 1) or a list (group 2) of relevant issues. Group 3 interacted with the usual ITS interface.

I utilized the UEQ to evaluate the user experience of the test groups. My study revealed that practitioners prefer using a chatbot tool for retrieving knowledge from ITSs. The chatbot outperformed the other tools in *Attractiveness*, *Efficiency*, *Stimulation*, and *Novelty* (RQ1). Furthermore, I conducted inferential statistics to analyze the task completion time and the confidence of participants in their answers. No significant differences were observed among the three test groups. However, the small sample size may be responsible for this outcome. The average response time for participants using the chatbot tool was slightly longer compared to the other groups, while it was the fastest for those using the link list tool (RQ2). Interestingly, participants who used the chatbot tool reported the highest confidence in their responses. They utilized the generated summary and the provided issue links for further verification. The issue list group participants expressed the lowest confidence ratings (RQ3). Analysis of prompt patterns revealed that the chatbot group used significantly longer and more detailed prompts compared to the link list group. The chatbot's conversational mode likely encouraged participants to utilize dialog-like prompts (RQ4). The inclusion of issue links in addition to the chatbot summary was not essential for task completion. The issue links served as a verification step and slightly increased the confidence of participants (RQ5). Finally, I analyzed the summaries generated by the chatbot using automatic metrics. My results indicate that traditional metrics are not well-suited to evaluate LLM-generated text. Utilizing metrics capable of capturing semantic meaning has been demonstrated to result in more effective outcomes (RQ6).

Furthermore, the experiment highlighted the importance of presenting information in a concise format, as participants were prone to overlooking it otherwise. Moreover, the results indicate that practitioners have not yet established trust in LLMs. However, participants were not fully familiar with the tool's implementation, which may have increased their trust.

My study introduced a novel approach for knowledge retrieval from multiple ITSs. Future work could explore the integration and evaluation of various LLMs. Furthermore, exploring various methods for retrieving information to integrate as custom knowledge might be beneficial. An interesting evolution of the tool could involve summarizing issues in a specified format prior to storing them in the database. Additionally, the output format could also be specified. This approach might lead to more predictable responses and further enhance the retrieval process. However, it is important to note that this results in longer processing times. Implementing a periodic task to retrieve new issues would be advantageous for the future use of this tool.

A larger-scale experiment involving more participants is necessary to derive more reliable conclusions. Additionally, conducting observations in real-world settings where the tool is employed and connected to an actively used ITS can provide valuable insights. This would reveal the tool's practicality and effectiveness in enhancing the knowledge retrieval process. An interesting starting point could be a user support setting where ITSs are utilized to document inquiries. In this scenario, the tool would retrieve previous inquiries, facilitating more efficient responses from the user support staff.

Appendix

A Experiment

Appendix A contains the precise wording of the three tasks used in the controlled experiment conducted in this study.

A.1 Task 1

Imagine you are using the **react-phone-number-input** library in your web application. It provides a React component that streamlines the process of inputting phone numbers. Based on the phone number in the input field, it automatically identifies the country, applies the appropriate format, and displays the corresponding country flag. For added flexibility, users can manually set the country by selecting the available options from a dropdown list.

When using the dropdown list (to manually select a country) it will only show the country names without a flag. You aim to have the emoji flags (unicode) displayed in the dropdown list as well. Find out how this can be achieved by using our provided knowledge retrieval tool.

A.2 Task 2

Imagine you are using **postgrest-py**, a PostgresREST client for Python. PostgREST is a standalone web server that turns your PostgreSQL database directly into a RESTful API. **postgrest-py** enables developers to formulate SQL queries in Python syntax using a predefined set of methods. Imagine you have the following query:

SELECT id, title, body, tags FROM issues WHERE status = 'OPEN';

This query would look like this in **postgrest-py**:

await client.from(”table”).select(”id”, “title”, “body”, "tags").eq(”status”, “OPEN”).execute()

Your task is to translate the following SQL query to the postgrest-py syntax:

SELECT id, title, body, tags FROM issues WHERE tags NOT IN ('wont fix', 'docs');

Please use our provided tool to retrieve the solution.

A.3 Task 3

Imagine you are using **postgrest-py**, a PostgresREST client for Python. PostgREST is a standalone web server that turns your PostgreSQL database directly into a RESTful API. **postgrest-py** enables developers to formulate SQL queries in Python syntax using a predefined set of methods. You are facing an issue while using **postgrest-py**. You are executing this query:

data = supabase.table('users').select('email, api_key').eq('api_key', 'fakekey').execute().dict()

This query should return elements where the **api_key** column matches the value **fakekey**.

You expect **data** to be empty because there is no **api_key** that is called **fakekey**. However, **data** contains database entries after executing this query. Your task is to use our tool and find out why the query is not working as expected.

Acronyms

AI Artificial Intelligence. 8, 40

API Application Programming Interface. 4, 5, 16, 20

EuXFEL European XFEL. 15, 20, 21, 42, 43

ITS Issue Tracking System. i, ii, 1--3, 6, 7, 9, 10, 13, 15, 16, 20, 21, 25--32, 34, 35, 39, 42--46

KB Knowledge Base. 9, 18

LLM Large Language Model. i, ii, v, 2, 3, 7--9, 11, 13, 16, 19--21, 23, 26, 27, 29, 31, 32, 36, 40--43, 45, 46

RAG Retrieval-Augmented Generation. 3, 16

RSC React Server Components. 16

SDK Software Development Kit. 19

UEQ User Experience Questionnaire. 13, 23, 45

UI user interface. 2, 13, 18, 20, 27--30

Bibliography

[1] Emad Aghajani et al. “Software Documentation Issues Unveiled”. In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 2019, pp. 1199–1210. DOI: 10.1109/ICSE.2019.00122.

[2] Emad Aghajani et al. “Software Documentation: The Practitioners' Perspective”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. ICSE '20. Seoul, South Korea: Association for Computing Machinery, 2020, pp. 590–601. ISBN: 9781450371216. DOI: 10.1145/3377811.3380405.

[3] Obead Alhadreti and Pam Mayhew. “Rethinking Thinking Aloud: A Comparison of Three Think-Aloud Protocols”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–12. ISBN: 9781450356206. DOI: 10.1145/3173574.3173618.

[4] Deeksha Arya et al. “Analysis and Detection of Information Types of Open Source Software Issue Discussions”. In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 2019, pp. 454–464. DOI: 10.1109/ICSE.2019.00058.

[5] AWS. *What is Retrieval-Augmented Generation?* 2024. URL: https://aws.amazon.com/what-is/retrieval-augmented-generation/ (visited on 04/13/2024).

[6] Olga Baysal, Reid Holmes, and Michael W. Godfrey. “No issue left behind: reducing information overload in issue tracking”. In: *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE 2014. Hong Kong, China: Association for Computing Machinery, 2014, pp. 666–677. ISBN: 9781450330565. DOI: 10.1145/2635868.2635887.

[7] Dane Bertram et al. “Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams”. In: CSCW '10. Savannah, Georgia, USA: Association for Computing Machinery, 2010, pp. 291–300. ISBN: 9781605587950. DOI: 10.1145/1718918.1718972.

[8] Tegawendé F. Bissyandé et al. “Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub”. In: *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*. 2013, pp. 188–197. DOI: 10.1109/ISSRE.2013.6698918.

[9] Yupeng Chang et al. *A Survey on Evaluation of Large Language Models*. 2023. DOI: 10.48550/arXiv.2307.03109.

[10] The SciPy community. *Statistical functions*. 2024. URL: https://docs.scipy.org/doc/scipy/reference/stats.html (visited on 01/28/2024).

[11] Ozren Dabic, Emad Aghajani, and Gabriele Bavota. “Sampling Projects in GitHub for MSR Studies”. In: *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. 2021, pp. 560–564. DOI: 10.1109/MSR52588.2021.00074.

[12] Themistoklis Diamantopoulos, Dimitrios-Nikitas Nastos, and Andreas Symeonidis. “Semantically-enriched Jira Issue Tracking Data”. In: *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 2023, pp. 218–222. DOI: 10.1109/MSR59073.2023.00039.

[13] Thu T. H. Doan et al. “Too long; didn’t read: Automatic summarization of GitHub README.MD with Transformers”. In: EASE '23. Oulu, Finland: Association for Computing Machinery, 2023, pp. 267–272. DOI: 10.1145/3593434.3593448.

[14] Thomas Dohmke. *100 million developers and counting*. 2023. URL: https://github.blog/2023-01-25-100-million-developers-and-counting/ (visited on 01/14/2024).

[15] *Embeddings*. 2024. URL: https://platform.openai.com/docs/models/embeddings (visited on 03/24/2024).

[16] Neil A. Ernst and Gail C. Murphy. “Case studies in just-in-time requirements analysis”. In: *2012 Second IEEE International Workshop on Empirical Requirements Engineering (EmpiRE)*. 2012, pp. 25–32. DOI: 10.1109/EmpiRE.2012.6347678.

[17] Michael A. Fligner and Timothy J. Killeen. “Distribution-Free Two-Sample Tests for Scale”. In: *Journal of the American Statistical Association* 71.353 (1976), pp. 210–213. DOI: 10.1080/01621459.1976.10481517.

[18] Github. *Understanding the search syntax*. 2024. URL: https://docs.github.com/en/search-github/getting-started-with-searching-on-github/understanding-the-search-syntax#exclude-results-with-specific-keywords (visited on 03/18/2024).

[19] Alon Halevy and Jane Dwivedi-Yu. *Learnings from Data Integration for Augmented Language Models*. 2023. DOI: 10.48550/arXiv.2304.04576.

[20] Hideaki Hata et al. “9.6 Million Links in Source Code Comments: Purpose, Evolution, and Decay”. In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 2019, pp. 1211–1221. DOI: 10.1109/ICSE.2019.00123.

[21] Yingying He et al. “Understanding and Enhancing Issue Prioritization in GitHub”. In: *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2023, pp. 813–824. DOI: 10.1109/ASE56229.2023.00044.

[22] Chenxu Hu et al. *ChatDB: Augmenting LLMs with Databases as Their Symbolic Memory*. 2023. DOI: 10.48550/arXiv.2306.03901.

[23] Xing Hu et al. “Correlating Automated and Human Evaluation of Code Documentation Generation Quality”. In: 31.4 (July 2022). ISSN: 1049-331X. DOI: 10.1145/3502853.

[24] Md Athikul Islam, Rizbanul Hasan, and Nasir U. Eisty. “Documentation Practices in Agile Software Development: A Systematic Literature Review”. In: *2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA)*. 2023, pp. 266–273. DOI: 10.1109/SERA57763.2023.10197828.

[25] Jean Kaddour et al. “Challenges and Applications of Large Language Models”. In: (2023). DOI: 10.48550/arXiv.2307.10169.

[26] Nikhil Kandpal et al. *Large Language Models Struggle to Learn Long-Tail Knowledge*. 2023. DOI: 10.48550/arXiv.2211.08411.

[27] Junaed Younus Khan and Gias Uddin. “Automatic Code Documentation Generation Using GPT-3”. In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. ASE '22. Rochester, MI, USA: Association for Computing Machinery, 2023. ISBN: 9781450394758. DOI: 10.1145/3551349.3559548.

[28] Mojtaba Komeili, Kurt Shuster, and Jason Weston. “Internet-Augmented Dialogue Generation”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 8460–8478. DOI: 10.18653/v1/2022.acl-long.579.

[29] William H. Kruskal and W. Allen Wallis. “Use of Ranks in One-Criterion Variance Analysis”. In: *Journal of the American Statistical Association* 47.260 (1952), pp. 583–621. DOI: 10.1080/01621459.1952.10483441.

[30] Abhishek Kumar, Partha Pratim Das, and Partha Pratim Chakrabarti. “Summarize Me: The Future of Issue Thread Interpretation”. In: *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2023, pp. 341–345. DOI: 10.1109/ICSME58846.2023.00042.

[31] Bettina Laugwitz, Theo Held, and Martin Schrepp. “Construction and Evaluation of a User Experience Questionnaire”. In: *HCI and Usability for Education and Work"*. Berlin, Heidelberg: Springer Berlin Heidelberg, Nov. 2008, pp. 63–76. ISBN: 978-3-540-89349-3. DOI: 10.1007/978-3-540-89350-9_6.

[32] Alon Lavie and Abhaya Agarwal. “Meteor: an automatic metric for MT evaluation with high levels of correlation with human judgments”. In: StatMT '07. Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 228–231.

[33] Howard Levene. “Robust tests for equality of variances”. In: *Contributions to probability and statistics* (1960), pp. 278–292.

[34] Chin-Yew Lin. “ROUGE: A Package for Automatic Evaluation of Summaries”. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013.

[35] Clara Marie Lüders, Abir Bouraffa, and Walid Maalej. “Beyond Duplicates: Towards Understanding and Predicting Link Types in Issue Tracking Systems”. In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 48–60. DOI: 10.1145/3524842.3528457.

[36] Clara Marie Lüders, Tim Pietz, and Walid Maalej. “Automated Detection of Typed Links in Issue Trackers”. In: *2022 IEEE 30th International Requirements Engineering Conference (RE)*. 2022, pp. 26–38. DOI: 10.1109/RE54965.2022.00010.

[37] Walid Maalej and Martin P. Robillard. “Patterns of Knowledge in API Reference Documentation”. In: *IEEE Transactions on Software Engineering* 39.9 (2013), pp. 1264–1282. DOI: 10.1109/TSE.2013.12.

[38] Walid Maalej et al. “On the Comprehension of Program Comprehension”. In: *ACM Trans. Softw. Eng. Methodol.* 23.4 (Sept. 2014). ISSN: 1049-331X. DOI: 10.1145/2622669.

[39] Ritch Macefield. “Usability studies and the Hawthorne effect”. In: *J. Usability Studies* 2 (May 2007), pp. 145–154.

[40] Michael Meng, Stephanie Steinhardt, and Andreas Schubert. “Application Programming Interface Documentation: What Do Software Developers Want?” In: *Journal of Technical Writing and Communication* 48.3 (2018), pp. 295–330. DOI: 10.1177/0047281617721853.

[41] Michael Meng, Stephanie Steinhardt, and Andreas Schubert. “How Developers Use API Documentation: An Observation Study”. In: *Commun. Des. Q. Rev* 7.2 (2019), pp. 40–49. DOI: 10.1145/3358931.3358937.

[42] Thorsten Merten et al. “Software Feature Request Detection in Issue Tracking Systems”. In: *2016 IEEE 24th International Requirements Engineering Conference (RE)*. 2016, pp. 166–175. DOI: 10.1109/RE.2016.8.

[43] Thorsten Merten et al. “Requirements Communication in Issue Tracking Systems in Four Open-Source Projects”. en. In: Matulevičius, Weyer et al. (Eds.): Joint Proceedings of REFSQ-2015 Workshops, Research Method Track, and Poster Track co-located with the 21st International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2015). Essen, Germany, March 23, 2015. CEUR Workshop Proceedings, Vol-1342. CEUR, 2015, pp. 114–125.

[44] *Models*. 2024. URL: https://platform.openai.com/docs/models/ (visited on 01/21/2024).

[45] Nornadiah Mohd Razali and Bee Yap. “Power Comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling Tests”. In: *Journal of Statistical Modeling and Analytics* 2 (Jan. 2011), pp. 21–33.

[46] Lloyd Montgomery, Clara Lüders, and Walid Maalej. “An Alternative Issue Tracking Dataset of PublicJira Repositories”. In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 73–77. DOI: 10.1145/3524842.3528486.

[47] Mathieu Nassif and Martin P. Robillard. “A Field Study of Developer Documentation Format”. In: *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI EA '23. Hamburg, Germany: Association for Computing Machinery, 2023. ISBN: 9781450394222. DOI: 10.1145/3544549.3585767.

[48] Anh Nguyen Duc et al. “Impact of Stakeholder Type and Collaboration on Issue Resolution Time in OSS Projects”. In: *Open Source Systems: Grounding Research*. Ed. by Scott A. Hissam et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–16. ISBN: 978-3-642-24418-6. DOI: 10.1007/978-3-642-24418-6_1.

[49] Arjun Pesaru, Taranveer Singh Gill, and Archit Reddy Tangella. “AI assistant for document management Using Lang Chain and Pinecone”. In: *International Research Journal of Modernization in Engineering Technology and Science* (2023). DOI: 10.56726/IRJMETS42630.

[50] *Prompt engineering*. 2024. URL: https://platform.openai.com/docs/guides/prompt-engineering (visited on 01/22/2024).

[51] Tim Puhlfürß, Lloyd Montgomery, and Walid Maalej. “An Exploratory Study of Documentation Strategies for Product Features in Popular GitHub Projects”. In: *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2022, pp. 379–383. DOI: 10.1109/ICSME55016.2022.00043.

[52] Mikko Raatikainen et al. “Improved Management of Issue Dependencies in Issue Trackers of Large Collaborative Projects”. In: *IEEE Transactions on Software Engineering* 49.4 (2023), pp. 2128–2148. DOI: 10.1109/TSE.2022.3212166.

[53] Paul Ralph et al. “ACM SIGSOFT Empirical Standards”. In: *CoRR* abs/2010.03525 (2020). DOI: 10.48550/arXiv.2010.03525.

[54] Sid Sijbrandij. *From 2/3 of the self-managed Git market, to the next-generation CI system, to Auto DevOps*. 2017. URL: https://about.gitlab.com/blog/2017/06/29/whats-next-for-gitlab-ci/ (visited on 01/24/2024).

[55] Venkatesh Sivaraman et al. “Ignore, Trust, or Negotiate: Understanding Clinician Acceptance of AI-Based Treatment Recommendations in Health Care”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI '23. Hamburg, Germany: Association for Computing Machinery, 2023. ISBN: 9781450394215. DOI: 10.1145/3544548.3581075.

[56] Larissa Rocha Soares et al. “Exploring Feature Interactions without Specifications: A Controlled Experiment”. In: *SIGPLAN Not.* 53.9 (Apr. 2020), pp. 40–52. ISSN: 0362-1340. DOI: 10.1145/3393934.3278127.

[57] S M Sohan et al. “A study of the effectiveness of usage examples in REST API documentation”. In: *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 2017, pp. 53–61. DOI: 10.1109/VLHCC.2017.8103450.

[58] UEQ team. *User Experience Questionnaire*. 2024. URL: https://www.ueq-online.org (visited on 02/12/2024).

[59] Haoye Tian et al. *Is ChatGPT the Ultimate Programming Assistant -- How far is it?* 2023. DOI: 10.48550/arXiv.2304.11938.

[60] Boxin Wang et al. *DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models*. 2024. DOI: 10.48550/arXiv.2306.11698.

[61] Xintao Wang et al. *KnowledGPT: Enhancing Large Language Models with Retrieval and Storage Access on Knowledge Bases*. 2023. DOI: 10.48550/arXiv.2308.11761.

[62] Yawen Wang et al. “What are Pros and Cons? Stance Detection and Summarization on Feature Request”. In: *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2023, pp. 1–12. DOI: 10.1109/ESEM56168.2023.10304865.

[63] Claes Wohlin et al. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012. ISBN: 3642290434. DOI: 10.1007/978-3-642-29044-2.

[64] Di Wu et al. “Retrieving API Knowledge from Tutorials and Stack Overflow Based on Natural Language Queries”. In: *ACM Trans. Softw. Eng. Methodol.* 32.5 (2023). ISSN: 1049-331X. DOI: 10.1145/3565799.

[65] Tao Xiao et al. “18 million links in commit messages: purpose, evolution, and decay”. In: *Empirical Software Engineering* 28 (2023), pp. 1–29. DOI: 10.1007/s10664-023-10325-8.

[66] Shengbin Xu et al. “Combining Code Context and Fine-grained Code Difference for Commit Message Generation”. In: Internetware '22. Hohhot, China: Association for Computing Machinery, 2022, pp. 242–251. ISBN: 9781450397803. DOI: 10.1145/3545258.3545274.

[67] Weizhe Yuan, Graham Neubig, and Pengfei Liu. “BARTScore: Evaluating Generated Text as Text Generation”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 27263–27277. URL: https://proceedings.neurips.cc/paper_files/paper/2021/hash/e4d2b6e6fdeca3e60e0f1a62fee3d9dd-Abstract.html.

[68] Jian Zhang et al. “Retrieval-based neural source code summarization”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. ICSE '20. Seoul, South Korea: Association for Computing Machinery, 2020, pp. 1385–1397. ISBN: 9781450371216. DOI: 10.1145/3377811.3380383.

[69] Tianyi Zhang et al. *BERTScore: Evaluating Text Generation with BERT*. 2019. DOI: 10.48550/arXiv.1904.09675.

[70] Ting Zhang et al. “Duplicate Bug Report Detection: How Far Are We?” In: *ACM Trans. Softw. Eng. Methodol.* 32.4 (May 2023). ISSN: 1049-331X. DOI: 10.1145/3576042.

[71] Yunfeng Zhang, Q. Vera Liao, and Rachel K. E. Bellamy. “Effect of confidence and explanation on accuracy and trust calibration in AI-assisted decision making”. In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. FAT* '20. Barcelona, Spain: Association for Computing Machinery, 2020, pp. 295–305. ISBN: 9781450369367. DOI: 10.1145/3351095.3372852.

[72] Weiqin Zou et al. “How Practitioners Perceive Automated Bug Report Management Techniques”. In: *IEEE Transactions on Software Engineering* 46.8 (2020), pp. 836–862. DOI: 10.1109/TSE.2018.2870414.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe.

Hamburg, den 1. Mai 2024

L. Ciepielowski

Lukas Ciepielowski

Erklärung zur Veröffentlichung

Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den 1. Mai 2024

L. Ciepielowski

Lukas Ciepielowski