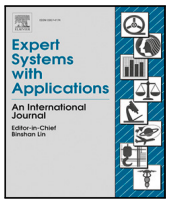
Expert Systems With Applications 267 (2025) 126113

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



An empirical study of ChatGPT-related projects and their issues on GitHub

Zheng Lin [a], Neng Zhang [b,*], Chao Liu [c], Zibin Zheng [a]

[a] School of Software Engineering, Sun Yat-sen University, Zhuhai, China
[b] School of Computer Science, Central China Normal University, Wuhan, China
[c] School of Big Data & Software Engineering, Chongqing University, Chongqing, China

ARTICLE INFO

Keywords:
Empirical study
ChatGPT
GitHub
Open-source project
Categorization
Topic modeling

ABSTRACT

Due to its powerful capabilities in natural language understanding and content generation, ChatGPT has received widespread attention since its launch in 2022. An increasing number of ChatGPT-related projects (that enhance the capabilities of ChatGPT, develop applications by calling ChatGPT APIs, etc.) are being released on GitHub and have sparked widespread discussions. However, GitHub does not provide a detailed classification of these projects to help users effectively explore interested projects. Additionally, the issues raised by users for these projects cover various aspects, e.g., installation, usage, and updates. It would be valuable to help developers prioritize more urgent issues and improve development efficiency. Unfortunately, there is currently no research focused on understanding the categories and issues of ChatGPT-related projects. To fill this gap, we retrieved 71,244 projects from GitHub using the keyword 'ChatGPT' and selected the top 200 representative projects with the highest numbers of stars as our dataset. By analyzing the project descriptions, we identified three primary categories of ChatGPT-related projects, namely *ChatGPT Implementation & Training*, *ChatGPT Application*, *ChatGPT Improvement & Extension*. We further built a classifier for automatically categorizing projects based on the 200 manually annotated projects. Next, we applied a topic modeling technique to 23,609 issues of those projects and identified ten issue topics, e.g., *model reply* and *interaction interface*. We analyzed the popularity, difficulty, and evolution of each issue topic within the three project categories and further proposed a method for recommending solutions for open issues by summarizing the pull requests associated with closed issues. Our main findings are: (1) The increase in the number of projects within the three categories is closely related to the development of ChatGPT; and (2) There are significant differences in the popularity, difficulty, and evolutionary trends of the issue topics across the three project categories. Based on these findings, we finally provided implications for project developers and platform managers on how to better develop and manage ChatGPT-related projects, such as offering more fine-grained tags to categorize projects to facilitate their exploration.

1. Introduction

In recent years, with the rapid development of artificial intelligence (AI), a considerable number of chat robots (Olujimi & Ade-Ibijola, 2023; Pérez-Soler et al., 2021; Song et al., 2023; Zhang et al., 2022) (also referred to as chatbots or conversational agents) have been developed. After receiving natural language input, chatbots can generate responses in various forms such as text, code, images, and video (Cañizares et al., 2022; Crawford & Paglen, 2021; Dagkoulis & Moussiades, 2023; Morris, 2023). Among these chatbots, ChatGPT[1] released by OpenAI on November 30, 2022 has received widespread attention. It is supported by the GPT-3.5 large language model (LLM). After its emergence, ChatGPT quickly gained popularity, surpassing one million users within five days and exceeding 100 million users within two months (Firat, 2023b; Firat & Kuleli, 2023). ChatGPT has numerous outstanding capabilities. For example, ChatGPT can capture contextual information from conversations and generate responses in the form of human conversation, greatly enhancing the interactive experience for users (Lund & Wang, 2023). ChatGPT also possesses powerful capabilities in code comprehension and generation (Tian et al., 2023). For instance, when a programmer expresses his/her functional requirements to ChatGPT and specifies the target programming language, ChatGPT can automatically generate the corresponding source code (Liu et al., 2023). In March 2023, OpenAI released an

* Corresponding author.
E-mail addresses: linzh228@mail2.sysu.edu.cn (Z. Lin), nengzhang@ccnu.edu.cn (N. Zhang), liu.chao@cqu.edu.cn (C. Liu), zhzibin@mail.sysu.edu.cn (Z. Zheng).

[1] https://chat.openai.com/.

https://doi.org/10.1016/j.eswa.2024.126113
Received 15 August 2024; Received in revised form 10 November 2024; Accepted 6 December 2024
Available online 13 December 2024

upgraded version GPT-4, which further enhanced the capabilities of ChatGPT (OpenAI, 2023).

Since the release of ChatGPT, a large number of ChatGPT-related projects (that enhance the capabilities of ChatGPT, develop applications by calling ChatGPT APIs, improve the usage of ChatGPT, etc.) have been increasingly created and hosted on GitHub, the largest open-source project hosting platform worldwide. These projects have also attracted a great number of discussions. By investigating the types of these emerging projects, it is possible to provide more fine-grained tags for the GitHub platform to categorize projects and enhance the project management capability. In addition, studying the popularity, difficulty, and evolutionary trends of issues raised by users can help developers prioritize key issues and improve development efficiency. However, there is no comprehensive study on investigating the types of these projects and their associated issues. To fill this gap, we collected 71,244 projects from GitHub using the keyword 'ChatGPT' and selected the top 200 projects with the highest number of stars. On GitHub, each project has an attribute called 'the number of stars', which reflects its popularity. Users can show their interest in a project by starring it (Cohen & Consens, 2018). Using these representative projects, we conducted several analyses to answer the following research questions (RQs).

RQ1. What categories of ChatGPT-related projects have been developed?

We manually examined the descriptions of 200 projects and identified three primary categories of ChatGPT-related projects, namely *ChatGPT Implementation & Training*, *ChatGPT Application*, *ChatGPT Improvement & Extension*, which include 8, 55, and 55 projects, respectively. Among the remaining 82 projects, 63 projects are irrelevant to ChatGPT (as explained in Section 3.2) and 19 projects belong to other minor categories, e.g., a collection of ChatGPT mirror websites or tools. In addition, we trained a classifier for automatic categorization of GitHub projects from the top 200 manually classified projects. These categories can help to understand how developers integrate ChatGPT into their projects and are also valuable for GitHub in improving the management and exploration of ChatGPT-related projects, as detailed in Section 5.1.

RQ2. What are the topics of issues discussed in ChatGPT-related projects?

We applied the Latent Dirichlet Allocation (LDA) (Blei et al., 2003) topic modeling technique to a set of 23,609 issues collected from 118 ChatGPT-related projects belonging to three primary categories. Since there is a significant portion (48.5%) of issues containing Chinese words, we propose a method to train a unified LDA model for issues expressed in both English and Chinese by semantically aligning the two kinds of words (see Sections 3.3 and 3.4), which results in ten issue topics, such as *interaction interface* and *gpt conversation*. We further analyzed the popularity of these issue topics within each project category. The results show that the popularity of each issue topic exhibits a significant difference across different categories. These issue topics and their popularity provide a comprehensive understanding of the problems that users encountered when using ChatGPT-related projects and also enable developers to improve the quality of their projects by prioritizing popular issues.

RQ3. How difficult is it to address the issues of each topic within different project categories?

We analyzed the difficulty of addressing the issues of each topic within the three project categories based on the average attention (i.e., the average number of comments and participants) received by the issues and closing rate of the issues. The results show significant differences across different project categories. For example, in the *ChatGPT Implementation & Training* category, although the *model reply* topic receives high attention, its closing rate is the lowest (57.2%), indicating that the issues of this topic are difficult to solve. In contrast, in the *ChatGPT Improvement & Extension* category, the closing rate of *model reply* is the highest (92.2%). Understanding the difficulty of issue topics, along with their popularity, can help developers better prioritize popular and challenging issues in their projects. In addition, to help developers address issues, we proposed a method to recommend solutions for open issues by summarizing and clustering solutions to similar issues.

RQ4. How do the issue topics evolve over time within different project categories?

We measured evolution trends of the issue topics using the Mann–Kendall trend test (Shourov & Mahmud, 2019). The issue topics show different evolution trends across the three project categories. For instance, the *npm runtime error* topic shows an increasing trend in the *ChatGPT Implementation & Training* category and a decreasing trend in the *ChatGPT Improvement & Extension* category, while there is no significant trend in the *ChatGPT Application* category. Based on the evolution trends of issue topics, developers can respond more quickly to changes in user requirements and adjust future development directions in a timely manner.

In summary, the main contributions of this study are listed as follows.

- We manually identified three primary categories of open-source projects closely related to ChatGPT on GitHub and trained a model for automatically categorizing projects. These categories can be used to better organize ChatGPT-related projects on GitHub or other project hosting platforms (e.g., Gitee[2]) and help users search for desired projects more effectively.
- We proposed a method for aligning Chinese words with English words and thus simultaneously building topic models for the issues of ChatGPT-related projects (or other text corpus) that are written in both English and Chinese.
- We discovered ten topics from the issues of ChatGPT-related projects and analyzed the popularity, difficulty, and evolution of these issue topics within different project categories. These results can facilitate the understanding and handling arrangements of the issues raised for ChatGPT-related projects.
- We proposed a method for generating and recommending solutions for open issues, which could help developers address the issues.
- We provided suggestions for developers and open-source platforms to improve the development and management of ChatGPT-related projects.

The rest of this paper is structured as follows. Section 2 reviews related work. Section 3 describes our research methodology. Section 4 presents answers to the four RQs. Section 5 provides implications for project developers and managers based on our findings and discusses threats to the validity of our research. Section 6 concludes the paper and discusses future work.

2. Related work

2.1. ChatGPT studies

There are many studies on the application of ChatGPT in different tasks, such as natural language processing (NLP), code analysis. Additionally, numerous studies have focused on the worries about ChatGPT.

NLP with ChatGPT. The natural language understanding and generation capabilities of ChatGPT have received increasing attention from researchers. Firat (2023a) believed that ChatGPT would become a promising tool in the field of open education. Due to its ability to understand and respond to natural language, ChatGPT could provide personalized guidance and assistance to learners, thereby improving

[2] https://gitee.com/.

their ability to independently solve problems and increasing their self-learning motivation. Búadóttir et al. (2023) developed a financial consulting tool that utilizes ChatGPT to answer questions. Users only need to submit their financial problems for consultation and resolution. Alessa and Al-Khalifa (2023) used the conversational interaction capability of ChatGPT to create conversation partners for the elderly, thereby helping the elderly reduce their loneliness and social isolation. Kozachek (2023) explored the potential of GPT-3, GPT-3.5, and GPT-4 in describing future scenarios of human society and provided several artificial scene examples to improve the quality of outputs. Therefore, providing high-quality prompts for ChatGPT is crucial for optimizing its generated results. White et al. (2023) examined several prompting patterns, such as improving requirements elicitation, improving code quality, improving system design, etc., and applied them to ChatGPT to help reduce and overcome common errors in software engineering tasks. However, possessing a certain amount of experience is essential for users to generate effective prompts. In order to reduce the time spent creating excellent prompts and improve the efficiency of using ChatGPT, Firat and Kuleli (2023) developed Auto-GPT, which automates the utilization of GPT-4. Users only need to describe their commands, and then Auto-GPT will automatically generate and execute prompts step by step to achieve the expected results.

Code analysis with ChatGPT. Nathalia et al. (2023) investigated the differences between ChatGPT, novice programmers and expert programmers in solving LeetCode competition problems in terms of performance and memory efficiency. They found that ChatGPT was better than novice programmers in solving simple and medium level problems. Haque and Li (2023) found that ChatGPT can identify bugs in code and provide suggestions, which not only reduces debugging time, but also makes it easier for developers to locate bugs in their programs. Tian et al. (2023) evaluated ChatGPT capabilities in accomplishing three code tasks: code generation, program repair, and code summarization. The results showed that ChatGPT performed well in these tasks, proving its potential as an assistant for programmers.

Worries about ChatGPT. Even though ChatGPT has excellent performance in natural language processing and code analysis, its use raises various concerns. Lo (2023) pointed out that since the training dataset of ChatGPT only goes up to 2021, it may yield biased or inaccurate knowledge when used in the education field. Moreover, the content generated by ChatGPT can bypass plagiarism detection programs, promoting a culture of student plagiarism. Chatterjee and Dethlefs (2023) found that ChatGPT can provide answers to questions that violate legal and ethical standards by altering the way of expression. Simultaneously, they provided an example to show how ChatGPT inadvertently introduces gender and racial discrimination in the coding process. Al-Hawawreh et al. (2023) explored how attackers can use ChatGPT to write malicious code snippets, circumvent system security measures and conduct network attacks.

2.2. GitHub issue analysis

On GitHub, issues are important means of tracking bug reports, new features, and ideas of a project.[3] Users can communicate, discuss, and seek assistance from project developers through issues (Izadi et al., 2022). Currently, issues are widely used as datasets in various studies. The issues of GitHub projects have been extensively used as datasets in various research studies. Yi et al. (2022) identified vulnerabilities in blockchain systems from GitHub issues and conducted a unique analysis on these vulnerabilities, namely summarizing vulnerable modules, types, and patterns. Win et al. (2023) identified 15 unethical behaviors based on GitHub issues and designed a tool called Etor to automatically detect these behaviors. Kallis et al. (2019) developed a tool to tag open issues in GitHub projects by analyzing their titles and text descriptions. Izadi et al. (2022) proposed a method to classify issues into bug reports, feature requests, or product support based on their text information and analyzed the priority of these issues. Huang et al. (2021) collected 79 features of issues from three dimensions: clarity of issue description, complexity of changes, and skills required to address issues. After analyzing the correlation between these features and Good First Issues (GFIs), they developed a model to automatically predict GFIs for projects.

2.3. Topic modeling

Topic modeling is an unsupervised text mining technique used to discover abstract topics within text (Wan et al., 2021). The most popular topic modeling techniques include Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (pLSA), and Latent Dirichlet Allocation (LDA). LSA decomposes a document-word matrix into low-dimensional latent semantic space representations of documents and words through singular value decomposition (SVD) (Deerwester et al., 1990). Although LSA is relatively simple to implement, it does not explicitly model the document generation process, making it difficult to explain word frequency changes and the probabilistic distribution of synonyms. pLSA improves LSA by introducing a probabilistic model with a generative process, but it still lacks a global document generation process, making it unable to perform topic inference on new documents (Hofmann, 1999). LDA addresses these limitations by modeling the topic distribution of each document using the Dirichlet distribution, allowing for the inference of topics for new documents (Blei et al., 2003). Prior studies have demonstrated that LDA performs better than LSA and pLSA in various topic modeling tasks (Battal & Koç, 2023; Beldi et al., 2022; Hosseiny Marani & Baumer, 2023). Therefore, we chose LDA to identify issue topics within ChatGPT-related projects.

LDA has also been widely used in the field of software engineering. Li et al. (2018) approximated the functionality of code snippets using topics identified by LDA. They studied the relationship between the identified topics and the likelihood of statements appearing in log records, aiming to guide developers' logging decisions. Bagherzadeh and Khatchadourian (2019) developed a set of big data tags to identify and extract posts related to big data from Stack Overflow (SO). They used LDA to identify big data topics discussed by developers and measured the popularity, difficulty, and relevance of topics. Haque et al. (2020) used SO posts to create a Docker-related dataset and applied LDA to identify dominant topics. The objective is to better understand developers' interest in Docker technology and the challenges they encounter. Han et al. (2020) used LDA to identify discussion topics related to three deep learning frameworks (i.e., TensorFlow, PyTorch, and Theano) across SO and GitHub. They compared topics within the three frameworks and between the two platforms. Zhang et al. (2019) proposed a method to extract service objectives from RESTful service description. They first used LDA to group available services, and then clustered service objectives based on the topic modeling results, thereby enhancing the efficiency of service discovery.

3. Research methodology

Fig. 1 shows an overview of our research methodology. At first, we retrieved an initial set of open-source projects that might be relevant to ChatGPT from GitHub. For each project, we collected various attributes, e.g., the project description, number of stars, and number of issues. We filtered projects without issues and selected the top 200 projects with the highest numbers of stars as our research dataset. By manually examining the project descriptions, we identified and classified ChatGPT-related projects into three primary categories, answering RQ1. We further collected and preprocessed the issues from these ChatGPT-related projects. Next, we trained a topic model to discover the topics of issues, answering RQ2. After that, by measuring

[3] https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues.

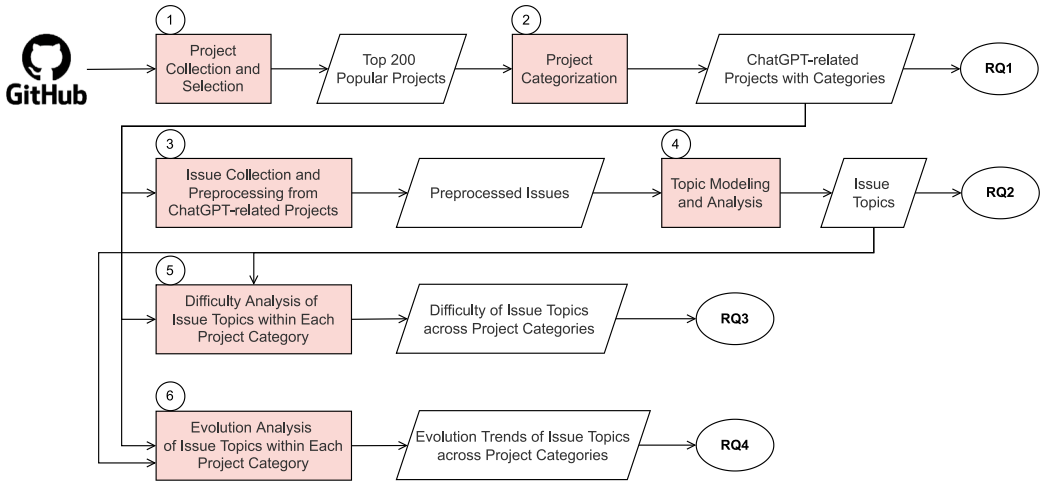


Fig. 1. Our research methodology.

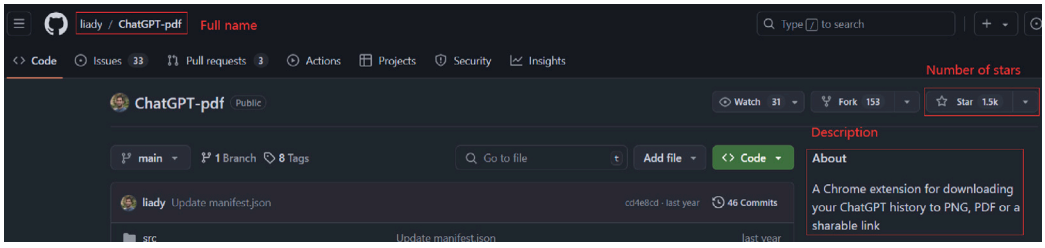


Fig. 2. Attributes collected from an example project 'liady/ChatGPT-pdf'.

the attention and the closed status of issues, we analyzed the difficulty in addressing issues across different topics, answering RQ3. Finally, we analyzed the evolution trend of each issue topic over time within the three categories, answering RQ4.

3.1. Project collection and selection

To conduct our study, we needed to collect a set of representative projects related to ChatGPT. As the world's largest open-source project platform, GitHub has attracted a large number of ChatGPT-related projects, and it offers a series of web APIs[4] to facilitate the retrieval of various kinds of data, e.g., projects and their issues.

We initially retrieved the projects created from Nov. 30, 2022 (the release date of ChatGPT) to Oct. 14, 2023, using the GitHub API with the search keyword 'ChatGPT', which resulted in 71,244 projects. For each project, we collected several attributes, including the full name, number of issues, number of stars, created time, and description (see Table 1). Specifically, the description of a project was extracted from the 'About' section and the README file, which summarize some important details, such as research intent, functionalities, implementation methods, and usage. Fig. 2 shows the attributes collected from an example project 'liady/ChatGPT-pdf'.[5]

We observed that a large number of the collected projects do not have any issues. Since one of our key objectives is to identify potential user requirements through the issues of ChatGPT-related projects, we removed the projects without any issues as they cannot provide valuable information. Table 2 presents several statistics on the number of issues of the retained 4541 projects. As can be seen, these projects have less than 16 issues on average; 50% projects have only one or two issues; and 75% projects have no more than eight issues. The projects with only a small number of issues may not be of much interest to users and may be unsuitable for our study. Generally, the number of stars can reflect the popularity of a project, and popular projects tend to have more issues reported by users (Cohen & Consens, 2018). To gather a set of representative projects with high popularity, we sorted the 4541 projects by the number of stars in descending order and selected the top 200 projects as our research dataset. In total, the top 200 projects have 39,985 issues, surpassing half of the total number of issues of the 4541 projects. Table 2 also presents several statistics calculated for the number of issues of these 200 projects.

In addition, we calculated the Pearson correlation coefficient (Kirch, 2008) between the created time and the number of stars for 200 projects using the `scipy.stats` package.[6] The value is 0.11, indicating that there is no significant linear correlation between the release time and the popularity of projects.

Table 1
Attributes collected for GitHub projects.

Attribute	Explanation
Full name	A unique identifier of a project, including the owner name and repository name, e.g., 'liady/ChatGPT-pdf'
Number of issues	The number of open/closed issues owned by a project
Number of stars	The number of stars owned by a project, which can reflect the popularity of the project
Created time	The created time of a project
Description	The content extracted from the 'About' section and the README file of a project, which summarizes the research intent, functionalities, implementation, and/or usage of the project

[4] https://api.github.com/.
[5] https://github.com/liady/ChatGPT-pdf.
[6] https://scipy.org/.

Table 2
Statistics calculated for the numbers of issues of two sets of GitHub projects.

Projects	#Total issues	Min	Max	Mean	Q1	Q2 (Media)	Q3
4541 projects with issues	72,510	1	1776	15.97	1	2	8
200 projects with the highest numbers of stars	39,985	1	1776	199.93	47	106.5	226.5

3.2. Project categorization

3.2.1. Manual classification

Based on our analysis of several projects, there are different types of projects about ChatGPT. For example, some projects propose methods to improve the capability or efficiency of ChatGPT, and some others focus on applying ChatGPT in specific task scenarios, e.g., code generation.

To help researchers, developers, and users understand what primary types of projects have been developed by leveraging ChatGPT, we decided to categorize the top 200 popular projects collected in the previous step. Since there were no pre-defined categories, the first and second co-authors of this paper performed the categorization task using an open card sorting approach in two phases (Gao et al., 2023). In the first phase, both co-authors manually and independently examined the description of each project and built a set of categories. After that, they discussed the categories together, which resulted in a common set of five categories: *ChatGPT Implementation & Training*, *ChatGPT Application*, *ChatGPT Improvement & Extension*, *Other ChatGPT-related Project*, and *Irrelevant Project*. The former three categories include the projects developed using ChatGPT. Their explanations can refer to Section 4.1. The *Other ChatGPT-related Project* category includes the projects relevant to ChatGPT but do not directly use ChatGPT, e.g., a project offering a collection of ChatGPT website mirrors or tools. The *Irrelevant Project* category includes the projects irrelevant to ChatGPT. Through our analysis, although all the projects were retrieved using the keyword 'ChatGPT', a number of projects are not really relevant, but are instead similar to ChatGPT. Such projects were retrieved due to the 'chatgpt' tag inaccurately assigned to them. For example, the project 'OpenLMLab/MOSS'[7] develops an open-source tool-enhanced conversational language model similar to ChatGPT, but the developer assigned the 'chatgpt' tag to it.

In the second phase, based on the determined categories, the two co-authors independently classified each project into these categories. There were 26 projects for which the co-authors assigned different categories. We measured the inter-rater agreement between the results of both co-authors using the Fleiss Kappa (Fleiss, 1971). The Kappa value was 0.82, which indicated almost perfect agreement. After discussing the disagreements together, both co-authors reached a consensus. There are 8, 55, 55, 19, and 63 projects assigned to the five categories, respectively.

3.2.2. Classification model for GitHub projects

To assist users and project management platforms in identifying whether a project is ChatGPT-related or not and which one of the three primary categories the project belongs to, we built an automatic classification model from the 200 manually categorized projects using the following four steps:

Step 1: Project description preprocessing. We preprocessed the project descriptions by removing several kinds of noise, including URLs, punctuations, numbers, and non-Chinese/English characters, using regular expressions, to reduce the impact of these noise on the classification model.

Step 2: Project description embedding. To build a classification model from a text corpus (e.g., the project descriptions), it is essential to represent each text as a vector. The traditional text vectorization methods, e.g., TF-IDF, rely on word frequency statistics and thus suffer from a loss of semantic information (Xu et al., 2016). We adopted a text embedding method to generate vectors that map text into a dense vector space while preserving semantic information (Carneiro et al., 2023). Specifically, we used the `SentenceTransformer` class of the `sentence_transformers` package[8] to generate embedding vectors for project descriptions.

Step 3: Oversampling. The numbers of projects manually classified into the five categories are quite different (see Table 8). For example, the *ChatGPT Implementation & Training* category contains only eight projects while *ChatGPT Application* and *ChatGPT Improvement & Extension* categories both contain 55 projects. This issue, well-known as '**class imbalance**' in the field of machine learning, impacts the overall performance of a statistical model trained from the dataset. Before training the classification model, we addressed this issue using the Synthetic Minority Oversampling Technique (SMOTE) (Lemaître et al., 2017). SMOTE is commonly used for handling imbalanced classification problems. It expands the minority classes by generating new samples based on the original samples and their random neighbors (Lemaître et al., 2017). Specifically, we used the `SMOTE` class of the `imblearn.over_sampling` package[9] to synthesize 55, 8, 8, and 44 samples, respectively, for the four categories: *ChatGPT Implementation & Training*, *ChatGPT Application*, *ChatGPT Improvement & Extension*, and *Other ChatGPT-related Project*. The final dataset contains 315 samples.

Step 4: Model training and evaluation. We used 80% of the oversampled dataset as the training set and the remaining 20% as test set. We trained classification models using five commonly used classifiers: Decision Tree (DT), Random Forest (RF), SVM, Logistic Regression (LR), and Naive Bayes (NB). These classifiers are implemented in the `sklearn` package[10] and have been widely used in previous work (Ho et al., 2024; Mathur et al., 2020; Misra et al., 2020; Rebro et al., 2023).

After applying the trained classification models to the test set, we first calculated the precision, recall, and F_1 score for each model with respect to each category.

$$Precision = \frac{|TP|}{|TP| + |FP|}$$

$$Recall = \frac{|TP|}{|TP| + |FN|}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

where TP, FN, FP, and FN, respectively, represent the numbers of true positives, false negatives, false positives, and false negatives predicted for a category by a model.

Next, we measured the macro-precision, macro-recall, and macro-F_1 score of a model as the arithmetic mean of the precision, recall, and F_1 score of the model achieved on all the five categories. Table 3 shows the results of five models. As can be seen, the RF model achieved the optimal performance and was determined as our final classifier.

3.3. Issue collection and preprocessing from ChatGPT-related projects

3.3.1. Issue collection

To investigate what kinds of issues have been reported in the projects developed using ChatGPT, we further collected the issues

[7] https://github.com/OpenLMLab/MOSS.

[8] https://sbert.net/.

[9] https://imbalanced-learn.org/stable/references/index.html.

[10] https://scikit-learn.org/stable/api/index.html.

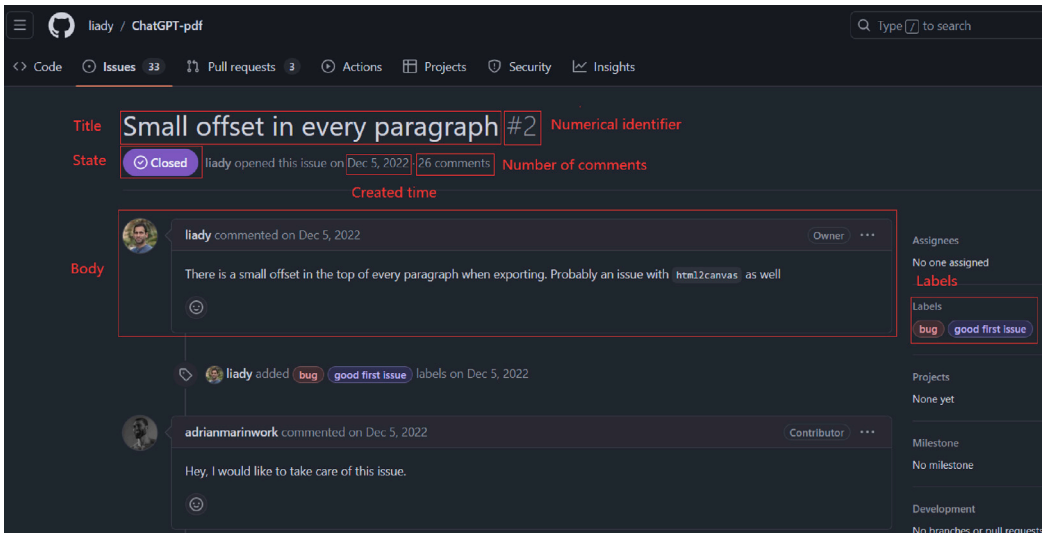


Fig. 3. Attributes collected from an example issue #2 of the project 'liady/ChatGPT-pdf'.

Table 3
The macro-precision, macro-recall, and macro-F_1 score results of five classifiers.

Classifier	macro-precision	macro-recall	macro-F_1
Decision Tree	0.634920635	0.633721181	0.637762238
Random Forest	0.80952381	0.819473684	0.8175
SVM	0.746031746	0.733690476	0.73986014
Logistic Regression	0.793650794	0.803754579	0.805151515
Naive Bayes	0.777777778	0.80634278	0.773846154

Table 4
Attributes collected for the issues of ChatGPT-related projects.

Attribute	Explanation
Numerical identifier	A unique identifier of an issue
Title	The title of an issue, which briefly summarizes the issue
Labels	A few keywords or phases (e.g., bug and enhancement) of an issue, which are defined by GitHub for issue classification and management
Body	A detailed description of an issue, offering specific content and contextual information about the issue, e.g., the related code, operations, and results
Created time	The created time of an issue
Number of comments	The number of comments associated with an issue
Number of participants	The number of participants who have commented on an issue
State	The state, i.e., open or closed, of an issue
Closed time	The closed time of an issue

of the 118(= 8 + 55 + 55) ChatGPT-related projects belonging to the three primary categories, i.e., *ChatGPT Implementation & Training*, *ChatGPT Application*, and *ChatGPT Improvement & Extension*. In total, we obtained 23,609 issues. For each issue, we collected its numerical identifier, title, labels, body, created time, number of comments, number of participants, state and closed time (see Table 4). Fig. 3 shows the attributes collected from an example issue #2[11] of the project 'liady/ChatGPT-pdf'.

3.3.2. Issue preprocessing

In this study, we decided to utilize the popular LDA topic modeling technique to discover the discussion topics of issues about ChatGPT-related projects. The main content of an issue is described in its title and body. However, through observation, there are several problems with the content, which may affect the quality of topic modeling. Specifically, (1) there are various kinds of noise, such as code snippets, bug reports, logs, HTML tags, and URLs. (2) Issues are written primarily in either Chinese or English. (3) There are stopwords (e.g., 'a' and 'the') and morphologically varied words, e.g., 'train' and 'training'. Therefore, before applying the LDA technique to the issues, we preprocessed the title and body of each issue using the following five steps. Tables 5 and 6 present the preprocessing results of two issues, an English issue[12] and a Chinese issue.[13]

Step 1: Noise content removal. When reporting an issue, the user often attaches code snippets, bug reports, or logs to clarify the related program and its execution results or errors. Although these information can help developers understand the issues, our experiments showed that retaining such details often results in generating topics with many meaningless keywords (e.g., 'line', 'java', 'py', 'python', and 'package'). Thus, similar to previous work (Han et al., 2020; Izadi et al., 2022; Pérez-Verdejo et al., 2021), we removed these information using regular expressions based on the markdown syntax of issues. Moreover, we removed several other kinds of noise, including HTML tags (e.g., '<p>'), URLs, file paths, default labels of the markdown-style issue templates (e.g., [BUG] and [FEAT]), numbers, punctuation marks, and non-Chinese/English characters.

Step 2: Word segmentation and lemmatization. Our analysis showed that a large proportion (48.5%) of issues containing Chinese words. We used the `posseg` module of jieba[14] to segment such issues. For the issues that only contain English words, we segmented them using the `word_tokenize` module of NLTK.[15] In addition, to reduce the interference of grammatical forms of English words when translating Chinese words to English words in step 4, we employed the `WordNetLemmatizer` module of NLTK to convert English words into their basic forms (aka. lemma) (Miller, 1995; Pérez-Verdejo et al., 2021). For instance, the lemmas of 'creating' and 'created' are both 'create'.

Step 3: Stopword removal. There are stopwords in both Chinese and English that can prevent LDA from identifying meaningful topics (Manning et al., 2008). We removed stopwords using the Chinese stopword list[16] and the Mallet's English stopword list.[17] These two

[11] https://github.com/liady/ChatGPT-pdf/issues/2.

[12] https://github.com/C-Nedelcu/talk-to-chatgpt/issues/116.
[13] https://github.com/fuergaosi233/wechat-chatgpt/issues/661.
[14] https://github.com/fxsjy/jieba.
[15] https://www.nltk.org.
[16] https://github.com/goto456/stopwords.
[17] https://github.com/mimno/Mallet/blob/master/stoplists/en.txt.

Table 5
Preprocessing of an English issue.

Preprocessing process	Result content
Raw text	Widget obstructs the edit button I love this extension, but the widget hovers exactly over the prompt edit button: ⟨img width = "90" alt = "image" src = "https://github.com/C-Nedelcu/talk-to-chatgpt/assets/64134453/57b7306c-450c-4d01-8a5f-64033389db42"⟩ This wouldn't be a big issue normally, but since dragging the widget around behaves very strangely, this is quite annoying. Regards!
After step 1	Widget obstructs the edit button i love this extension but the widget hovers exactly over the prompt edit button this wouldn t be a big issue normally but since dragging the widget around behaves very strangely this is quite annoying regards
After steps 2–3	Widget; obstructs; edit; button; love; extension; widget; hovers; prompt; edit; button; wouldn; big; issue; drag; widget; behaves; strangely; annoy; regard;
After steps 4–5	Widget; obstruct; edit; button; love; extens; widget; hover; prompt; edit; button; wouldn; big; issu; drag; widget; behav; strang; annoy; regard;

stopword lists are widely used in previous work (Ai et al., 2024; Cheng et al., 2023; Gao et al., 2023; Härtel et al., 2018).

Step 4: Chinese word translation. To uniformly build a topic model for issues expressed in both English and Chinese, we proposed a method for translating Chinese words to semantically equivalent English words. We first collected all Chinese words and all English words from the issues and sorted these two sets of words in descending order by their frequencies, which resulted in a Chinese word list, CW, and an English word list, EW. Since the Baidu Translation APIs[18] provide mature documentation for developers and have good performance in translating contextual words (Xie, 2022), we used the Baidu Translation APIs to translate each Chinese word $w \in CW$ to its corresponding English word, w'. If w' exists in EW, we replaced w with w' in all issues.

Step 5: Stemming. Stemming can effectively handle morphological variations of words and improve the quality of issue texts (Zhang et al., 2019). For instance, 'create', 'creation', 'creating', and 'created' are all stemmed to 'creat'. Therefore, similar to previous work (Gao et al., 2023; Silva et al., 2021; Zhang et al., 2019), we used the `SnowballStemmer` module of NLTK to reduce each English word to its root form (aka. stem), thereby eliminating morphologically varied words.

3.4. Topic modeling and analysis

We wanted to discover the discussion topics of issues in ChatGPT-related projects. Although GitHub allows users to annotate issues with some pre-defined labels or tags (e.g., 'bug' and 'enhancement'), these labels are often coarse-grained and cannot support fine-grained analysis (Han et al., 2020). For example, issues labeled as 'bug' may be related to API calls, login, pre-training, etc. Moreover, users may not always label issues consistently or correctly (Siddiq & Santos, 2023). Therefore, we employed the well-known LDA topic modeling technique to discover fine-grained topics from the issues of ChatGPT-related projects. For a corpus of textual documents, LDA assumes that each document has a probability distribution over a set of latent topics, and each topic has a probability distribution over the set of words in the corpus. It uses a bag-of-words model to estimate the topic distribution of each document and the word distribution of each topic based on the co-occurrence of words in the documents (Barua et al., 2014).

In LDA, there is a key parameter, namely the number of latent topics (denoted as K), which controls the granularity of the discovered topics (Barua et al., 2014). A larger or smaller K may both result in poor performance. To determine the optimal K, we utilized the coherence score to evaluate the quality of the trained topic model. The coherence score measures the semantic consistency between words within a topic (Newman et al., 2010). A higher coherence score indicates stronger semantic associations among words, that is, the topic is more understandable and meaningful.

We applied the `LdaModel` module provided in the Gensim library[19] to the entire set of preprocessed issues. We trained different topic models by setting K from 5 to 50 with a step 1. Based on the coherence scores of the models measured using the `CoherenceModel` module (see Fig. 4), we identified the optimal $K = 10$ and selected the corresponding model as the final topic model. The first and second co-authors collaboratively named the topics based on the top ten keywords of each topic and some issues belonging to the topic. The topics (e.g., *request method* and *model reply*) with their top ten keywords are presented in Table 10.

According to the trained topic model, each issue possesses a probability distribution of K topics. We denoted the probability of a specific topic $z_k (k = 1, \ldots, K)$ in the issue d_i as $\theta(d_i, z_k)$, where $0 \leq \theta(d_i, z_k) \leq 1$ and $\sum_{i=1}^{k} \theta(d_i, z_k) = 1$. To answer RQ2, we defined the topic with the highest probability as the *dominant topic* of d_i, i.e., $dominant\_topic(d_i) = z_k : \theta(d_i, z_k) = \max(\theta(d_i, z_j)), 1 \leq j \leq K$. Next, for each of the three primary categories of ChatGPT-related projects (see Section 4.1), e.g., c_j, we measured the *popularity* of an issue topic z_k within c_j as the proportion of issues whose dominant topic is z_k, i.e.,

$$popularity(z_k, c_j) = \frac{|Issues(c_j, z_k)|}{|Issues(c_j)|} \tag{1}$$

where $Issues(c_j)$ represents the set of all issues from the projects within c_j; and $Issues(c_j, z_k)$ represents the subset of issues whose dominant topic is z_k, i.e., $\{d_i | d_i \in Issues(c_j) \wedge dominant\_topic(d_i) = z_k\}$.

Based on the trained LDA topic model, we can help developers quickly identify the topic of a newly created issue. Specifically, we first preprocessed the issue text using the method described in Section 3.3.2. Next, we converted the issue text into a vector using the same dictionary used to train LDA model. Finally, we inferred the topic distribution using the trained LDA model and determined the dominant topic based on the probability distribution of topics.

3.5. Difficulty analysis of issue topics within each project category

3.5.1. Evaluation indicators

Intuitively, if an issue has received much attention but it has not been closed yet, then the issue should be probably difficult to address. The attention of an issue can be reflected by the number of comments and the number of participants associated with the issue. Based on these ideas, for each primary category of ChatGPT-related projects, c_j, we investigated the difficulty in addressing the issues of each topic z_k within c_j by measuring the average attention and closing rate of all issues belonging to z_k, as described as follows.

Average number of comments. Within c_j, we calculated the average number of comments for z_k as the ratio of the total number of comments on the issues belonging to z_k to the number of all issues belonging to z_k, i.e.,

$$avg\_num_{comments}(c_j, z_k) = \frac{\sum_{d_i \in Issues(c_j, z_k)} num_{comments}(d_i)}{|Issues(c_j, z_k)|} \tag{2}$$

where $num_{comments}(d_i)$ represents the number of comments on the issue d_i.

Average number of participants. Within c_j, we calculated the average number of participants for z_k as the ratio of the total number

[18] https://fanyi-api.baidu.com/product/11.

[19] https://radimrehurek.com/gensim/.

Table 6
Preprocessing of a Chinese issue.

Preprocessing process	Result content
Raw text	登 录 问 题 ![image](https://user-images.githubusercontent.com/103298985/221082393-591af1e2-82de-4c09-b9f2-e984568a0c2b.png) 给 了 正确的账号和密码登录不了，是不是要挂打码平台？，能不能给一个手动登录的?
After step 1	登录问题 给了正确的账号和密码登录不了 是不是要挂打码平台 能不能给一个手动登录的
After steps 2-3	登录;正确;账号;密码;登录;挂;码;平台;手动;登录;
After steps 4-5	login; correct; account; password; login; hang; code; platform; manual; login;

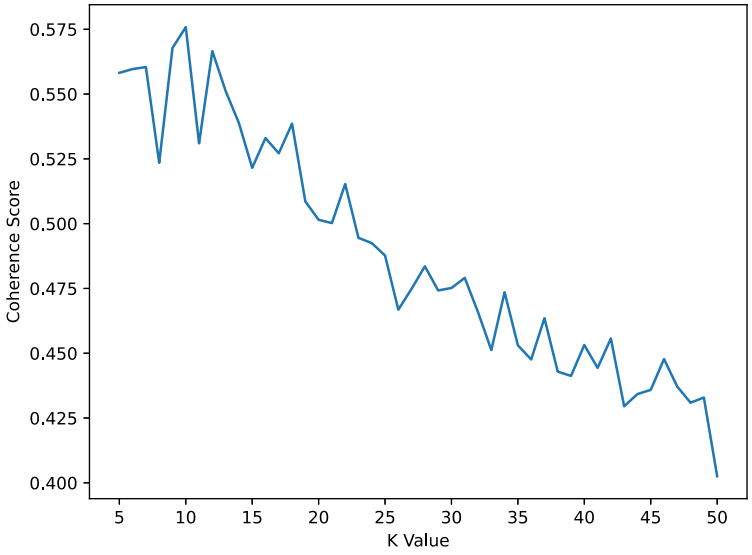


Fig. 4. The coherence scores corresponding to different numbers of topics, i.e., K.

of participants who commented on the issues belonging to z_k to the number of all issues belonging to z_k, i.e.,

$$avg\_num_{participants}(c_j, z_k) = \frac{\sum_{d_i \in Issues(c_j, z_k)} num_{participants}(d_i)}{|Issues(c_j, z_k)|} \quad (3)$$

where $num_{participants}(d_i)$ represents the number of participants who commented on the issue d_i.

Closing rate. Within c_j, we calculated the closing rate for z_k as the ratio of the total number of closed issues belonging to z_k to the number of all issues belonging to z_k, i.e.,

$$avg\_rate_{closed}(c_j, z_k) = \frac{\sum_{d_i \in Issues(c_j, z_k)} I(state(d_i) == closed)}{|Issues(c_j, z_k)|} \quad (4)$$

where $state(d_i)$ represents the state of the issue d_i; and $I(condition)$ is an indicator function whose value is 1 if the $condition$ is true, otherwise 0.

We calculated the Pearson correlation coefficient between the attention to issues (i.e., the number of comments and the number of participants) and their survival time (i.e., the difference between closed time and created time). The values are 0.09 and 0.14 respectively, indicating that there is no significant linear correlation between the attention and survival time of issues.

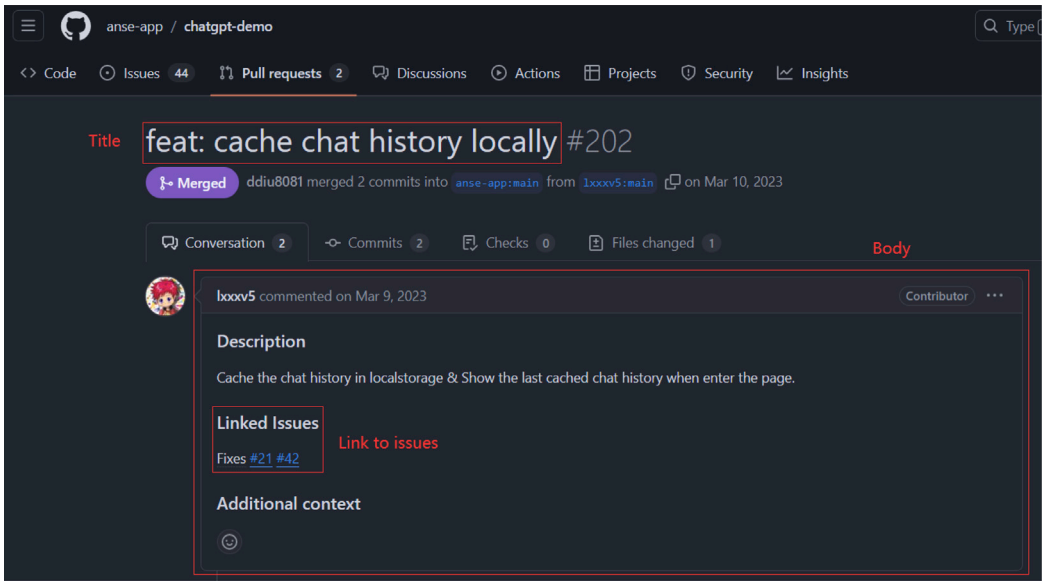
3.5.2. Solution recommendation for open issues

To help developers and users address difficult issues more effectively, we proposed a method to automatically recommend solutions for open issues, which contains the following four steps.

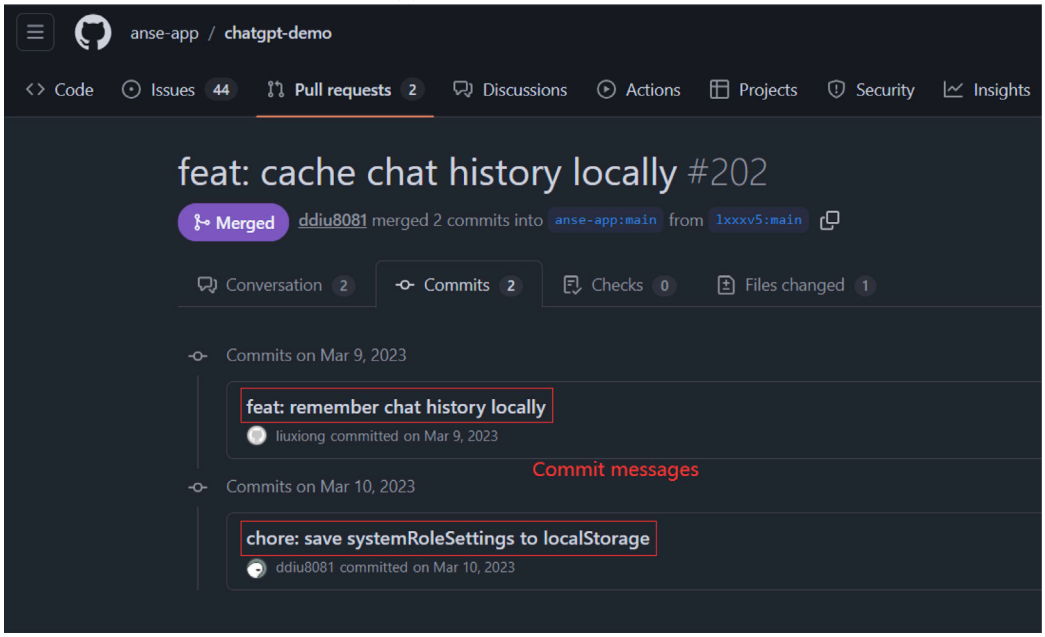
Step 1: Collect pull requests associated with closed issues. On GitHub, pull requests are important means for developers to add functionality or fix errors within a project. The body and commit messages of pull requests provide potential solutions to issues. Based on the link relationships between pull requests and their corresponding issues,[20] we collected pull requests, including their bodies and commit messages, for each closed issue. Fig. 5 shows the body and commit messages collected from an example pull request #202[21] of the project 'anse-app/chatgpt-demo' and the linked issues.

Step 2: Identify similar issues. For an open issues, we searched for similar issues with pull requests. Specifically, we preprocessed the issue text and generated its embedding vector using the method described in Section 3.2.2. Next, we calculated the Cosine similarity between the

[20] https://docs.github.com/en/issues/tracking-your-work-with-issues/using-issues/linking-a-pull-request-to-an-issue.
[21] https://github.com/anse-app/chatgpt-demo/pull/202.



(a) Body and linked issues



(b) Commit messages

Fig. 5. Body and commit messages collected from an example pull request #202 of the project 'anse-app/chatgpt-demo'.

open issue and every closed issue which has pull requests. To avoid noises, we considered the closed issues with the similarity ≥0.7 as similar to the open issue.

Step 3: Generate pull request summaries. To recommend concise solutions, we designed a prompt (see Fig. 6) to generate summaries for pull requests using the open-source large language model 'DeepSeek-V2.5' offered by the DeepSeek platform.[22]

Step 4: Cluster and recommend solutions. We applied the spectral clustering algorithm, which is particularly well-suited for handling complex-shaped clusters and small or medium-sized datasets (Yang et al., 2024), to cluster the pull request summaries of similar closed issues. Specifically, we used the `SpectralClustering` class from the `sklearn.cluster` package. We set the number of clusters from 2 to the total number of pull requests and determined the optimal number of clusters by measuring the silhouette coefficients of different cluster models. The silhouette coefficient evaluates the compactness and separation of clustering results by calculating the relationship between the average distance of a sample to other samples in the same cluster and the average distance of the sample to samples in the nearest cluster (Belyadi & Haghighat, 2021). Finally, the clustered pull request summaries were recommended as potential solutions to the open issue.

As an example, Table 7 shows the solutions recommended for the open issue #81[23] of the project 'yakGPT/yakGPT'. For this issue, we retrieved three similar issues associated with four pull requests. There are two solutions clustered from the pull request summaries.

[22] https://platform.deepseek.com.

[23] https://github.com/yakGPT/yakGPT/issues/81.

```
I will provide the body and commit messages of a pull request associated with an issue in JSON format:
{
    "body": "_",
    "commit messages": "_"
}
Please read these carefully and help me generate a summary.

Requirements:
1.Return a summary in JSON format: {"summary": ""}
2.Do not output any irrelevant content or explanatory descriptions.
3.Limit the summary content to 50 words.
```

Fig. 6. Prompt template used to generate pull request summaries.

Table 7
Solutions recommended for the open issue #81 of the project 'yakGPT/yakGPT'.

Issue text	Summaries clustering
LocalAI support? I would be glad if this allowed not needing to use OpenAI API Key. This, with the custom base URL, would allow the use of https://github.com/go-skynet/LocalAI which extends these clients to allow more and even custom models without the need to depend on OpenAI.	**Cluster1:** Added support for a base OpenAI URL to enable use of external proxies or Azure OpenAI. **Cluster1:** Adds support for custom HTTP proxies to OpenAI API requests. **Cluster2:** Refactor openai-edge to openai, add logs, fix baseURL error, and update CI types. **Cluster2:** Support Azure OpenAI, refactor settings data structure, and fix various CI and styling issues.

3.6. Evolution analysis of issue topics within each project category

The hotspots of issues will change during the progress of a project. We explored the evolution of the issue topics within each of the three primary project categories. For each issue topic, z_k, within a specific project category, c_j, we first defined and calculated the monthly variation of the impact of z_k based on prior work (Wan et al., 2021) as

$$impact(c_j, z_k, m) = \frac{1}{|Issues(c_j, z_k, m)|} \sum_{d_i \in Issues(c_j, z_k, m)} \theta(d_i, z_k) \quad (3)$$

where $Issues(c_j, z_k, m)$ represents the issues belonging to z_k that are raised for the projects in c_j in a specific month, m.

Afterwards, we performed the Mann–Kendall trend test (MK test) to determine the evolution trend in the impact of each issue topic at a significance level of 0.05 using the `pymannkendall` package.[24] The MK test is a non-parametric statistical test method for assessing trend changes in time series data. Additionally, we calculated the Theil–Sen slope (Shourov & Mahmud, 2019) to quantify the magnitude of the monotonic trend, which is often used in conjunction with the MK test. Based on the results, we identified the issue topics that exhibit increasing or decreasing trends and inferred their causes.

4. Results

4.1. RQ1. What categories of ChatGPT-related projects have been developed?

According to the categorization method described in Section 3.2, we identified four categories of ChatGPT-related projects: *ChatGPT Implementation & Training*, *ChatGPT Application*, *ChatGPT Improvement & Extension*, *Other ChatGPT-related Project*, which are explained as follows.

ChatGPT Implementation & Training. The projects in this category focus on studying the underlying model of ChatGPT and implementing it through other large language models. Additionally, they explore ways to improve the training process of ChatGPT, including optimizing training methods for model parameters, fine-tuning the model to adapt to different downstream tasks, and constructing datasets for model training. For example, the project 'PhoebusSi/Alpaca-CoT'[25] builds a fine-tuning platform which is easy to get started and use.

ChatGPT Application. The projects in this category aim to unlock the potential capabilities of ChatGPT by leveraging it to solve various tasks, e.g., text translation and code generation. Moreover, calling ChatGPT APIs to develop conversational robots has also received widespread attention. For example, the project 'gragland/chatgpt-chrome-extension'[26] develops a tool for integrating ChatGPT into text boxes on the Internet.

ChatGPT Improvement & Extension. The projects in this category improve ChatGPT from different aspects, including the calling of ChatGPT APIs and the functional expansion and performance enhancement of ChatGPT. Specifically, developers encapsulate the ChatGPT APIs for easier application calls, assist users in resolving various network issues that may arise when accessing ChatGPT, provide prompt templates to optimize ChatGPT's replies, and use network search results to enhance the accuracy of results generated by ChatGPT. For example, the project 'Yidadaa/ChatGPT-Next-Web'[27] develops a cross-platform user interface for ChatGPT.

Other ChatGPT-related Project. This category contains the other ChatGPT-related projects that cannot be classified into the three categories above, e.g., a collection of ChatGPT mirror websites, survey reports about ChatGPT, etc. For example, the project 'LiLittleCat/awesome-free-chatgpt'[28] collects a list of free ChatGPT mirror websites.

Table 8 presents the number of projects assigned to the four categories. In addition, we counted the number of projects growing in every month for the three primary categories: *ChatGPT Implementation & Training*, *ChatGPT Application*, *ChatGPT Improvement & Extension*, as depicted in Fig. 7. As can be seen, the monthly increase in the number of projects within the *ChatGPT Application* and *ChatGPT Improvement & Extension* categories peaked in December 2022 and March 2023. This

[24] https://pypi.org/project/pymannkendall/.

[25] https://github.com/PhoebusSi/Alpaca-CoT.

[26] https://github.com/gragland/chatgpt-chrome-extension.

[27] https://github.com/Yidadaa/ChatGPT-Next-Web.

[28] https://github.com/LiLittleCat/awesome-free-chatgpt.

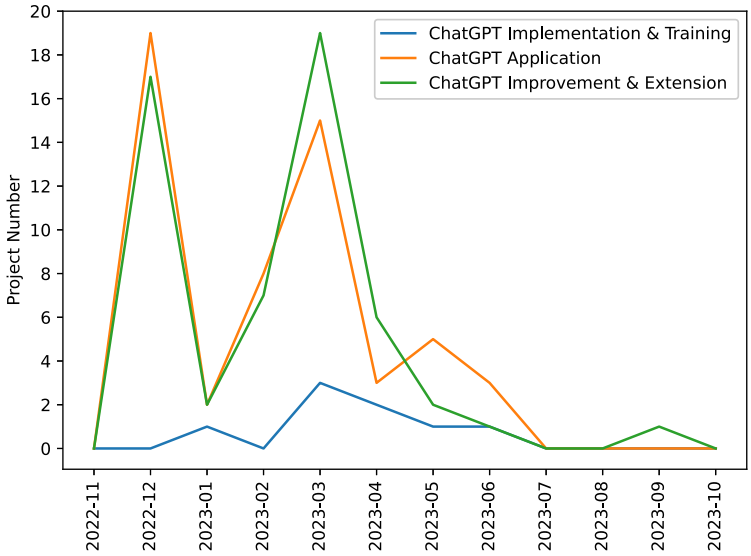


Fig. 7. The monthly increased numbers of projects within three primary categories.

Table 8
The numbers of ChatGPT-related projects within four categories.

Category	#Projects
ChatGPT implementation & Training	8
ChatGPT application	55
ChatGPT improvement & Extension	55
Other ChatGPT-related project	19

Table 9
The classification results of the Random Forest classifier.

Category	#Projects
ChatGPT implementation & Training	116
ChatGPT application	16,878
ChatGPT improvement & Extension	11,913
Other ChatGPT-related project	1016
Irrelevant project	6327

phenomenon is closely related to the evolution of ChatGPT. Specifically, after OpenAI launched the first version of ChatGPT built on the GPT-3.5 model on November 30, 2022, there was a significant increase in the number of ChatGPT-related projects on GitHub. The first projects in the two categories were created on December 2, 2022 and December 1, 2022, respectively, followed by a rapid growth in that month. Subsequently, OpenAI launched the multi-modal model GPT-4 in March 2023, which significantly enhanced the capabilities of GPT-3.5 and further stimulated the enthusiasm of researchers and developers. For example, in the issue #471[29] (created on March 15, 2023) of the project 'transitive-bullshit/chatgpt-api' (created on December 3, 2022), the user hoped that GPT-4 could be incorporated into the project. The project 'enricoros/big-agi'[30] created on March 19, 2023 is a personal AI application powered by GPT-4. As a result, the growing number of projects in the *ChatGPT Application* and *ChatGPT Improvement & Extension* categories reached the second peak in March 2023. Later, OpenAI launched the iOS version of ChatGPT in May 2023, which also promoted the secondary development of ChatGPT to a certain extent. For example, in the issue #142[31] (created on May 26, 2023) of the project 'sunner/ChatALL', the user hoped that developers could add the iOS client models of ChatGPT to their project. Therefore, in May 2023, the number of projects in the *ChatGPT Application* category saw another increase.

Moreover, when ChatGPT was initially released, most developers focused on improving the use of ChatGPT or exploring ways to conduct secondary development based on ChatGPT. As time progressed, an increasing number of open-source large language models began to emerge, making customized implementation and training of ChatGPT possible. For example, the project 'mymusise/ChatGLM-Tuning'[32] (created on March 16, 2023) used the Tsinghua's ChatGLM-6B model as an affordable solution for ChatGPT implementation. Thus, a peak in the growing number of projects in the *ChatGPT Implementation & Training* category occurred in March 2023.

We applied the RF classifier trained in Section 3.2.2 to the remaining 36,250 projects with descriptions. Table 9 presents the classification results. Nearly half (46.6%) of the projects belong to the *ChatGPT Application* category.

4.2. RQ2. What are the topics of issues discussed in ChatGPT-related projects?

As described in Section 3.4, in order to understand the issues reported about ChatGPT-related projects, we applied the LDA topic model to the dataset of 23,609 issues collected from the projects in the three primary categories identified in RQ1. Table 10 presents the ten discovered issue topics and their top 10 keywords. Moreover, we calculated the popularity of each topic in the overall dataset and the three primary project categories using Eq. (1), as shown in Fig. 8.

Fig. 8(a) shows the popularity of issue topics in the overall dataset. We can see that the popularity of the four topics, namely *request method*, *interaction interface*, *npm runtime error*, and *gpt conversation*, is greater than 13%. The popularity of the other four topics, i.e., *model reply*, *docker deployment*, *login access*, and *api key*, is greater than 8%. The popularity of *project language* and *module import & result export* is the lowest, with only 3.9% and 2.2%, respectively. Due to the fact that the *ChatGPT Application* and *ChatGPT Improvement & Extension* categories account for a large proportion of ChatGPT-related projects, the issues are largely relevant to the improvement and secondary development

[29] https://github.com/transitive-bullshit/chatgpt-api/issues/471.
[30] https://github.com/enricoros/big-agi.
[31] https://github.com/sunner/ChatALL/issues/142.
[32] https://github.com/mymusise/ChatGLM-Tuning.

Table 10
The issue topics with their top ten keywords.

Topic name	Top ten keywords
request method	request featur function support chatgpt add user api share enhanc
model reply	model answer amd output inform prompt interfac time paramet data
interaction interface	bug click open button window time display file input work
project language	project languag ai model version chines support pictur chrome work
docker deployment	error chatgpt docker deploy api messag bug send openai web
npm runtime error	error run npm instal version work file warn build code
module import & result export	import export line administr client arm imag slow error speed
gpt conversation	gpt chat convers prompt add model messag user set make
login access	error server login access fail proxi connect agent password http
api key	api token key dialogu word alloc access mode modifi prompt

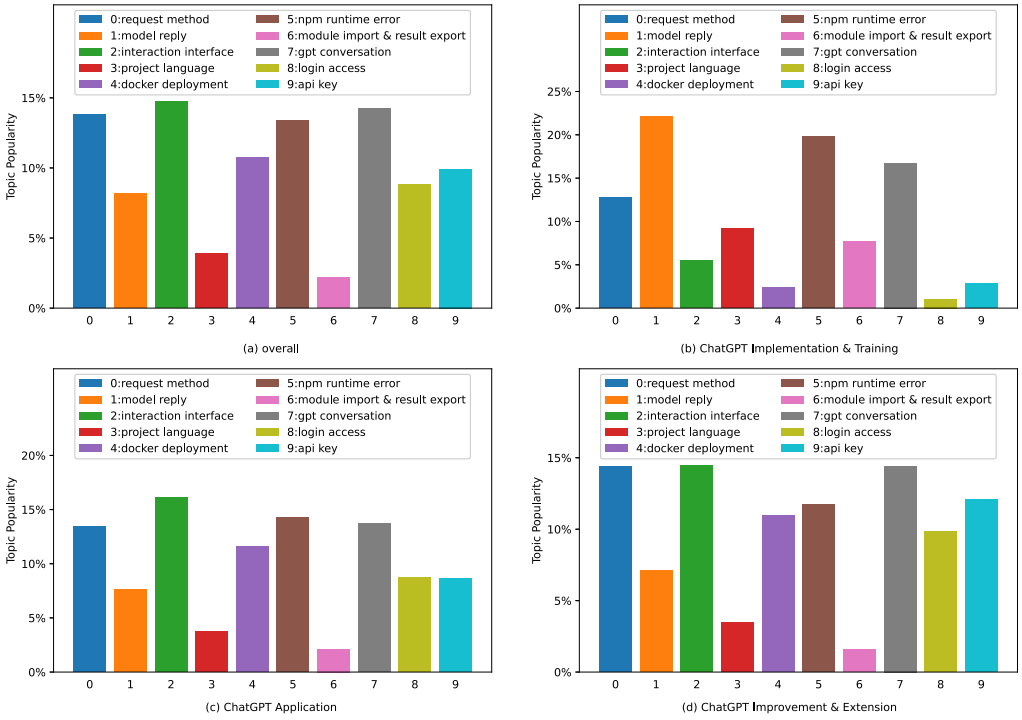


Fig. 8. The popularity of issue topics in the overall dataset and within the three primary project categories.

of ChatGPT. For example, there is significant focus on improving the interactive interface of ChatGPT, the methods of sending requests to ChatGPT, and communicating with ChatGPT to obtain the conversation results, etc.

Fig. 8(b) shows the popularity of each topic within the *ChatGPT Implementation & Training* category, which notably differs from the topic popularity in the overall dataset. Specifically, the popularity of the *model reply*, *project language*, *npm runtime error*, and *module import & result export* topics is obviously higher than the others. In particular, *model reply* has the highest popularity of 22.1%. Conversely, the popularity of *interaction interface*, *docker deployment*, *login access*, and *api key*, is much lower than that shown in Fig. 8(a). This is because the projects in the *ChatGPT Implementation & Training* category primarily focus on implementing and training ChatGPT-like models. Consequently, they pay more attentions to the replies generated by the model and the storage of output results. For example, when LoRA technology[33] was applied to fine-tune the model, a user raised an issue about the responses being too short.[34] Developers can fine-tune or retrain the model based on the exported results to make it closer to ChatGPT.

Fig. 8(c) shows the popularity of issue topics within the *ChatGPT Application* category. Compared to the results shown in Fig. 8(a), except for *interaction interface* (whose popularity is higher) and *api key* (whose popularity is lower), there is no significant difference in the popularity of the other topics. The projects in this category typically use ChatGPT as the underlying model or call ChatGPT APIs for development, thereby integrating the capabilities of ChatGPT into their application tools. As an application tool, users pay more attention to the interaction interface and may also propose new interaction requirements. For example, in the issue #1006[35] of the project 'binary-husky/gpt_academic', the user hoped to add a small button next to the submit button to decide whether to carry historical information during conversations. Although the projects in this category frequently call ChatGPT APIs to utilize its functionalities, which might lead to more issues about *api key*, the official documentation from OpenAI provides detailed instructions on how to apply for and use the ChatGPT APIs with examples. Therefore, users can solve most of *api key* related issues, which may explain the low popularity of the *api key* topic.

Fig. 8(d) shows the popularity of issue topics within the *ChatGPT Improvement & Extension* category. Compared to the results shown in

[33] Low-Rank Adaptation of Large Language Models (LoRA) technology is a low-dimensional reparameterization fine-tuning technique that improves efficiency by reducing the number of fine-tuning parameters (Hu et al., 2021).

[34] https://github.com/PhoebusSi/Alpaca-CoT/issues/111.

[35] https://github.com/binary-husky/gpt_academic/issues/1006.

Table 11
Attention and solution status of each issue topic within the *ChatGPT Implementation & Training* category.

Topic name	#Issues	avg_num$_{comments}$	avg_num$_{participants}$	avg_rate$_{closed}$
request method	168	1.86	1.33	68.5%
model reply	292	3.37	2.18	57.2%
interaction interface	72	2.86	1.81	72.2%
project language	121	2.4	1.68	70.2%
docker deployment	31	2.94	1.77	80.6%
npm runtime error	262	3.05	1.88	81.7%
module import & result export	101	3.68	2.13	77.2%
gpt conversation	221	2.14	1.34	75.6%
login access	14	3.29	2.14	71.4%
api key	38	2.53	1.68	63.2%

Fig. 8(a), the popularity of *login access* and *api key* is higher, while the popularity of *model reply* and *npm runtime error* is lower. This is because the goal of *ChatGPT Improvement & Extension* is to facilitate the use of ChatGPT. Therefore, providing convenient login and access methods, as well as improving the methods for calling ChatGPT APIs, are important in the projects of this category. For example, in the issue #363[36] and #1022[37] of the project 'Yidadaa/ChatGPT-Next-Web', users expressed a desire to add multi-api key polling functionality.

4.3. RQ3. How difficult is it to address the issues of each topic within different project categories?

We defined and measured three metrics, i.e., average number of comments (avg_num$_{comments}$), average number of participants (avg_num$_{participants}$), and closing rate (avg_rate$_{closed}$), to investigate the difficulty in addressing the issues of each topic within the three primary ChatGPT-related project categories, as described in Section 3.5. These metrics reflect the attention and solution status of the issue topics. Generally, a topic with higher attention and a lower closing rate probably indicates that the issues of the topic are more difficult to address.

Table 11 presents the metric results measured for each issue topic within the *ChatGPT Implementation & Training* category. Particularly, the *model reply* topic has the highest popularity (see Fig. 8(b)) and also leads in both average number of comments and average number of participants, however its closing rate is the lowest. This result means that the issues related to this topic are difficult to address since they received high attention, but only a low proportion of them have been resolved. We believe this phenomenon occurs because different underlying models and datasets could be used for training, which leads to unexpected results in the implementation of models similar to ChatGPT. For example, in the issue #617[38] of the project 'OptimalScale/LMFlow', the user used a LLM ('Chinese-Llama2-7b') to test the fine-tuning ability of the tool. However, this model is different from the underlying model of ChatGPT, which results in output errors. Additionally, the training process requires multiple rounds of fine-tuning which depends on the dataset and the model response. This not only requires a considerable amount of time, but may also yield different results during every training session, which increases the difficulty in solving related issues.

Table 12 presents the metric results of each issue topic within the *ChatGPT Application* category. The *interaction interface*, the most popular topic in this category (see Fig. 8(c)), ranks about halfway among all topics in terms of average number of comments and average number of participants, indicating that the issues related to this topic received relatively high attention. Additionally, the closing rate of *interaction interface* reaches 72.7%, showing that most of the issues raised by users have been addressed. This could be explained by the reason that the projects in the *ChatGPT Application* category are primarily application tools. Addressing the issues related to the interaction interface is beneficial for the promotion and usability of these tools.

In contrast, another popular topic *request method* has the lowest closing rate. This may be because the projects often need to send various types of requests. For example, the project 'sunner/ChatALL'[39] is able to communicate simultaneously with many large language models, such as ChatGPT, Bing Chat, ChatGLM, and MOSS, to find the best answer. However, different types of requests have different parameter settings and rules, which leads to many unpredictable issues. For example, in the issue #301[40] of the project 'sunner/ChatALL', the user found that 'claude-v1' model disconnects every ten minutes and requires a manual refresh to solve this issue. Therefore, the issues related to this topic are often difficult to address.

Table 13 presents the metric results of each issue topic within the *ChatGPT Improvement & Extension* category. The average number of comments and participants of each topic are higher than those in the other two categories, indicating that these topics received higher attention. Additionally, except for *gpt conversation*, the closing rates of the other topics are all close to or above 75% and significantly exceed those of the corresponding topics in the other two categories. This shows that most issues in this category have been resolved. This could be explained by the reason that the developers in this category need to have a good understanding of ChatGPT and thus can quickly address the issues.

Moreover, the popular topic *gpt conversation* (see Fig. 8(d)) has the lowest closing rate, i.e., 61.9%. This is because in the *ChatGPT Improvement & Extension* category, most issues of this topic concern improving the conversation process. Users often request new features to enhance their experience. For example, users may wish to have multiple sessions simultaneously rather than following the same pattern as on the ChatGPT official website.[41] Implementing new features could be a challenge for developers, which increases the difficulty in addressing these issues.

From the results presented in Tables 11, 12, and 13, there are significant differences in the difficulty addressing the ten issue topics within each project category. Moreover, the difficulty addressing the same issue topic also varies across different project categories. For example, the closing rate of *model reply* topic in the *ChatGPT Implementation & Training* category is only 57.2%, whereas it reaches 92.2% in the *ChatGPT Improvement & Extension* category. We attribute this to the different research focus of the three project categories. In the *ChatGPT Implementation & Training* category, the issues of *model reply* are often related to underlying models and datasets used to implement ChatGPT. However, different underlying models yield different training effects. For example, in the issue #197[42] of the project 'OptimalScale/LMFlow', the user experienced poor training results due to the poor model 'gpt_neo_2.7B'. Addressing such issues requires developers to have rich

[36] https://github.com/Yidadaa/ChatGPT-Next-Web/issues/363.
[37] https://github.com/Yidadaa/ChatGPT-Next-Web/issues/1022.
[38] https://github.com/OptimalScale/LMFlow/issues/617.
[39] https://github.com/sunner/ChatALL.
[40] https://github.com/sunner/ChatALL/issues/301.
[41] https://github.com/mckaywrigley/chatbot-ui/issues/107.
[42] https://github.com/OptimalScale/LMFlow/issues/197.

Table 12
Attention and solution status of each issue topic within the *ChatGPT Application* category.

Topic name	#Issues	$avg\_num_{comments}$	$avg\_num_{participants}$	avg_rate_{closed}
request method	1497	2.28	1.47	66.3%
model reply	854	3	1.83	75.5%
interaction interface	1795	2.76	1.7	72.7%
project language	414	2.24	1.5	67.4%
docker deployment	1292	3.57	2.26	81.7%
npm runtime error	1594	2.96	1.85	77.7%
module import & result export	238	2.64	1.62	72.3%
gpt conversation	1530	2.45	1.5	76.9%
login access	976	3.76	2.31	78.3%
api key	965	2.88	1.97	74.9%

Table 13
Attention and solution status of each issue topic within the *ChatGPT Improvement & Extension* category.

Topic name	#Issues	$avg\_num_{comments}$	$avg\_num_{participants}$	avg_rate_{closed}
request method	1600	2.86	1.87	74.8%
model reply	790	3.25	2.16	92.2%
interaction interface	1608	3.08	1.95	75.7%
project language	388	2.65	1.77	78.4%
docker deployment	1218	4.25	2.5	84.9%
npm runtime error	1308	3.31	2.04	76.4%
module import & result export	178	2.98	1.97	83.1%
gpt conversation	1606	2.69	1.74	61.9%
login access	1096	4.1	2.61	88.0%
api key	1342	3.35	2.2	88.8%

experience in training large language models. In contrast, projects in the *ChatGPT Application* and *ChatGPT Improvement & Extension* categories typically implement functionality by calling ChatGPT APIs. As a result, most issues of *model reply* stem from the interactions with ChatGPT, including slow response speeds,[43] incomplete replies,[44] and crafting prompts for specific responses.[45] Resolving these issues generally requires only proficiency in using ChatGPT.

4.4. RQ4. How do the issue topics evolve over time within different project categories?

As described in Section 3.6, we first calculated the monthly changes in the impact of each issue topic within the three primary ChatGPT-related project categories. Then, we used the MK test to determine the evolution trend of each topic.

Table 14 presents the evolution trend of the impact of each issue topic within the *ChatGPT Implementation & Training* category. The impact of *model reply* shows a decreasing trend, while the impact of *npm runtime error* and *gpt conversation* exhibits an increasing trend. The impact of the other topics does not show significant trends. This is because the *model reply* topic is related to the training phase of a ChatGPT-like model. In the early stages of a project, there is a significant gap between the open-source model used to implement ChatGPT and the actual ChatGPT model. Both developers and users need to train and fine-tune the model based on its responses. As the process improves, the model gradually becomes similar to the ChatGPT model. Thus, the impact of *model reply* gradually decreases; thereafter users shift their attention to the configuration and use of the model, which leads to more issues about *npm runtime error* and *gpt conversation*. For example, users can install the JavaScript packages of the project through npm commands.[46] In the issue #35[47] of the project 'Instruction-Tuning-with-GPT-4/GPT-4-LLM', the user reported that there exist a considerable number of low-quality dialogue samples in 'unnatural_instruction_gpt4_data.json'.

Table 14
Evolution trend of each issue topic within the *ChatGPT Implementation & Training* category.

Topic name	Evolution trend	p-value	Sen's slope
request method	No trend	0.063486531	0.020925277
model reply	Decreasing	0.004434008	−0.046970927
interaction interface	No trend	0.265510389	0.00461322
project language	No trend	0.10776229	−0.006612922
docker deployment	No trend	0.901538627	7.2633E−05
npm runtime error	Increasing	0.035447893	0.011221673
module import & result export	No trend	0.063486531	−0.009560359
gpt conversation	Increasing	0.009374768	0.040220779
login access	No trend	0.10776229	−0.001600849
api key	No trend	0.901538627	−0.001691792

Table 15
Evolution trend of each issue topic within the *ChatGPT Application* category.

Topic name	Evolution trend	p-value	Sen's slope
request method	Increasing	0.003093449	0.005350561
model reply	No trend	0.086768173	0.001541284
interaction interface	Increasing	0.003093449	0.010677818
project language	No trend	0.119470987	0.001846343
docker deployment	Decreasing	0.000613905	−0.01015641
npm runtime error	No trend	0.275757848	−0.003741957
module import & result export	No trend	0.876269664	5.49696E−05
gpt conversation	Increasing	0.00506931	0.008503926
login access	Decreasing	0.001845721	−0.006764936
api key	Decreasing	0.012731364	−0.004841313

Therefore, the impact of these two issue topics increases accordingly. Moreover, since the other topics are almost unrelated to the early training and later use of the model, there are no significant trends in their impact.

Table 15 presents the evolution trend of the impact of each issue topic within the *ChatGPT Application* category. As time goes by, the impact of *request method*, *interaction interface*, and *gpt conversation* shows an increasing trend. Conversely, the impact of *docker deployment*, *login access*, and *api key* shows a decreasing trend. The impact of the other topics does not show significant trends. This is because the projects in the *ChatGPT Application* category aim to apply ChatGPT in various tasks. After developers complete the development of

[43] https://github.com/josStorer/chatGPTBox/issues/166.
[44] https://github.com/lencx/ChatGPT/issues/601.
[45] https://github.com/dair-ai/Prompt-Engineering-Guide/issues/61.
[46] https://github.com/embedchain/embedchain/issues/122.
[47] https://github.com/Instruction-Tuning-with-GPT-4/GPT-4-LLM/issues/35.

Table 16
Evolution trend of each issue topic within the *ChatGPT Improvement & Extension* category.

Topic name	Evolution trend	p-value	Sen's slope
request method	No trend	0.161124949	0.000929693
model reply	No trend	0.533416513	0.002768807
interaction interface	No trend	0.161124949	−0.002702788
project language	Increasing	0.042960146	0.001112902
docker deployment	No trend	0.640428787	0.000223462
npm runtime error	Decreasing	0.042960146	−0.008559295
module import & result export	No trend	0.876269664	0.000103658
gpt conversation	No trend	0.350201389	−0.002809677
login access	No trend	0.350201389	0.001377567
api key	No trend	0.086768173	0.008431346

a project, users usually need to install and deploy the project based on its README file and then configure the ChatGPT APIs to login and use the project. Therefore, in the early stages of a project, users mainly focus on issues related to deployment, configuration, and login access. For example, the issues #127,[48] #130,[49] and #250[50] of the project 'fuergaosi233/wechat-chatgpt' are all related to the docker deployment. As the project improves and developers provide answers to these initial issues, they are no longer a hindrance for users. Thus, users start to pay more attention to the interaction interface and the quality of conversation. This shift leads to a decreasing trend in the impact of *docker deployment*, *login access*, and *api key*, and an increasing trend in the impact of *request method*, *interaction interface*, and *gpt conversation*.

Table 16 presents the evolution trend of the impact of each issue topic within the *ChatGPT Improvement & Extension* category. Over time, the impact of *project language* shows an increasing trend, while the impact of *npm runtime error* shows a decreasing trend. The impact of the other topics does not show significant trends. This is because the projects in the *ChatGPT Improvement & Extension* category focus on enhancing various aspects of ChatGPT, such as usage, functionality extension, and performance improvement. Consequently, these projects often introduce additional software packages to facilitate the use of ChatGPT and extend its functionalities. However, software packages in the npm module repository may undergo updates and rollbacks, leading to incompatibility and crashes. Therefore, the issues related to *npm runtime error* often occur in the early stages of a project. For example, a user encountered an error 'the module could not be found' when executing the command 'npm run dev'.[51] But with the improvement of the project and the introduction of more stable npm software packages, the issues gradually decrease. Moreover, ChatGPT interacts with humans through dialogue, which requires accepting various types of language and text inputs. Thus, the impact of *project language* is increasing. For example, a user expressed a desire for the project 'ztjhz/BetterChatGPT' to provide a LaTeX rendering switch to render inline formulas to improve the readability.[52] As for the topics *request method*, *interaction interface*, *gpt conversation*, *login access*, and *api key*, which are important aspects of ChatGPT improvement, their impact does not show significant trends.

5. Discussion

From the results of RQ1, ChatGPT-related projects on GitHub can be broadly classified into three primary categories: *ChatGPT Implementation & Training*, *ChatGPT Application*, and *ChatGPT Improvement & Extension*. The number of projects in each category is closely tied to the evolution of ChatGPT. In RQ2, we calculated the popularity of the ten issue topics identified by LDA and observed significant differences in popularity within each category. Additionally, the popularity of the same topic (e.g., *api key*) varies across different categories. According to the results of RQ3, the most difficult topics to address in the *ChatGPT Implementation & Training*, *ChatGPT Application*, and *ChatGPT Improvement & Extension* categories are *model reply*, *request method*, and *gpt conversation*, respectively. Finally, from the results of RQ4, the impact of topics (e.g., *npm runtime error*) has different evolution trends across the three primary categories. Based on these findings, we discuss the implications for project management platforms and developers of ChatGPT-related projects.

5.1. Implications

Implications for project management platforms. Our identified categories of ChatGPT-related projects and trained classification model can effectively facilitate the organization and categorization of projects on GitHub or other platforms (e.g., Gitee). For example, when developers write a description for a project, the platform can use our classification model to categorize this new project and suggest a more fine-grained tag, e.g., *ChatGPT Implementation & Training* or *ChatGPT Application*, if the project is related to ChatGPT. This can not only optimize the management of ChatGPT-related projects but also refine the search results for users, thereby enhancing their retrieval efficiency.

Implications for developers. There are several implications for developers to develop, improve, and promote their ChatGPT-related projects. (i) **Focusing on the application and improvement of ChatGPT.** According to the classification results in Section 4.1, most ChatGPT-related projects on GitHub are associated with developing ChatGPT-based applications and improving the use of ChatGPT. These projects enable practitioners in various fields to benefit from ChatGPT, which has high economic and practical value and is worthy of further research. (ii) **Staying updated with ChatGPT developments.** The development of ChatGPT is crucial for developing and improving ChatGPT-related projects. For example, after the release of GPT-4, developers should promptly integrate it into their projects promptly and continuously stay aligned with the development of ChatGPT, which can quickly meet user requirements and enhance project visibility. (iii) **Attention should vary across different categories of ChatGPT-related projects.** The popularity and difficulty of issue topics are different within different project categories. Developers should pay more attention to the popular and difficult issues of projects in each category. For example, in the *ChatGPT Implementation & Training* category, developers should focus more on issues related to *model reply*. It is advisable to specify the dataset, underlying model, and parameter settings used during training in the README file to help users replicate the process. In the *ChatGPT Application* category, developers should timely satisfy new requirements raised by users regarding *interaction interface* and ensure that the descriptions of *request method* in the README file are clear and complete. In the *ChatGPT Improvement & Extension* category, developers can enhance the experience of using ChatGPT by providing diverse interfaces and extending API calling methods. In addition, the topic inference method for newly created issues (see Section 3.4) and the solution recommendation method for open issues (see Section 3.5.2) can help developers more effectively identify issue topics and address difficult issues. (iv) **Adjusting development focus over time.** The development focus of a project should be adjusted over time since the impact of issue topics will evolve during the progress of the project. For example, in the *ChatGPT Implementation & Training* category, as the project progresses, the proportion of issues related to the *gpt conversation* topic will increase. Developers need to focus on issues that contain keywords such as 'gpt', 'chat', and 'conversation'. In the *ChatGPT Application* category, the impact of *interaction interface* topic will increase and thus developers need to prioritize addressing issues that contain keywords like 'click', 'open', and 'button'.

[48] https://github.com/fuergaosi233/wechat-chatgpt/issues/127.
[49] https://github.com/fuergaosi233/wechat-chatgpt/issues/130.
[50] https://github.com/fuergaosi233/wechat-chatgpt/issues/250.
[51] https://github.com/pashpashpash/vault-ai/issues/47.
[52] https://github.com/ztjhz/BetterChatGPT/issues/251.

5.2. Threats to validity

Threats to internal validity relate to the errors in the implementation of our research methodology as well as the subjective bias of participants in the manual categorization of ChatGPT-related projects. To ensure the correctness of our research methodology, we double-checked the implementation code and execution of every step shown in Fig. 1. For the categorization of the top 200 projects with the highest numbers of stars, the first and second co-authors of the paper performed the task using an open-card sorting approach in two phases, i.e., category determination and project classification, as described in Section 3.2. To minimize the subjective bias, both co-authors independently conducted each phase task and then resolved any inconsistencies through discussion to reach a common decision.

Threats to external validity relate to the generalizability of our results. In this work, we selected the top 200 open-source projects (searched using the keyword 'ChatGPT') with the highest numbers of stars from GitHub, the world's largest project hosting platform, as our candidate dataset of ChatGPT-related projects. These projects cover a representative set of the most popular projects related to ChatGPT. Moreover, we collected a relatively large dataset of 23,609 issues from the 118 ChatGPT-related projects within three primary categories. These two popular and large datasets could help improve the generalizability of our results.

6. Conclusion and future work

With the rapid development and widespread dissemination of ChatGPT, developers have hosted a large number of ChatGPT-related projects on GitHub. These projects have sparked extensive discussions. In this study, we investigated the categories of ChatGPT-related projects and further analyzed the popularity, difficulty, and evolution of ten issue topics discovered using the LDA topic model within three primary project categories. Based on our findings, we provide useful suggestions for project developers and hosting platforms (such as GitHub and Gitee) to better develop and manage ChatGPT-related projects. For example, the platforms can use our categories as fine-grained tags to label their projects, thus promoting the exploration experience for users. According to the popularity and difficulty of the issue topics, developers can devote more efforts and resources to solve popular and challenging issues raised for their projects. Developers can also adjust the development focus of their projects during the progress in a timely manner based on the evolution trends of issue topics.

In future work, we plan to establish a specific metric for assessing the difficulty addressing issues within each topic and conduct a quantitative analysis of these difficulties. This will enable us to better identify difficult issues, enhance the efficiency of resolving issues, and promote project development.

CRediT authorship contribution statement

Zheng Lin: Performs the design, implementation, and experiments of the study as well as the writing and revision of the paper. **Neng Zhang:** Leads the design, implementation, and experiments of the study as well as the writing and revision of the paper. **Chao Liu:** Participates in the study design and paper revision. **Zibin Zheng:** Provides suggestions on the paper revision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62302536) and the Guangdong Basic and Applied Basic Research Foundation (2023A1515012292).

Data availability

Data will be made available on request.

References

Ai, Z., Chiu, D. K. W., & Ho, K. K. W. (2024). Social media analytics of user evaluation for innovative digital cultural and creative products: Experiences regarding dunhuang cultural heritage. *Journal on Computing and Cultural Heritage*, *17*(3), URL https://doi.org/10.1145/3653307.

Al-Hawawreh, M., Aljuhani, A., & Jararweh, Y. (2023). Chatgpt for cybersecurity: practical applications, challenges, and future directions. *Cluster Computing*, *26*(6), 3421–3436, URL https://doi.org/10.1007/s10586-023-04124-5.

Alessa, A., & Al-Khalifa, H. (2023). Towards designing a ChatGPT conversational companion for elderly people. In *Proceedings of the 16th international conference on pervasive technologies related to assistive environments* (pp. 667–674). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3594806.3596572.

Bagherzadeh, M., & Khatchadourian, R. (2019). Going big: a large-scale study on what big data developers ask. In *ESEC/FSE 2019, Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering* (pp. 432–442). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3338906.3338939.

Barua, A., Thomas, S. W., & Hassan, A. E. (2014). What are developers talking about? An analysis of topics and trends in stack overflow. *Empirical Software Engineering*, *19*(3), 619–654, URL https://doi.org/10.1007/s10664-012-9231-y.

Battal, O. M., & Koç, A. (2023). Automatic construction of sememe knowledge bases from machine readable dictionaries. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *32*, 1023–1035, URL https://doi.org/10.1109/TASLP.2023.3347927.

Beldi, A., Sassi, S., & Jemai, A. (2022). Learn2Sum: A new approach to unsupervised text summarization based on topic modeling. In *Proceedings of the 14th international conference on management of digital ecosystems* (pp. 136–143). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3508397.3564853.

Belyadi, H., & Haghighat, A. (2021). Chapter 4 - unsupervised machine learning: clustering algorithms. In H. Belyadi, & A. Haghighat (Eds.), *Machine learning guide for oil and gas using python* (pp. 125–168). Gulf Professional Publishing, URL https://www.sciencedirect.com/science/article/pii/B9780128219294000020.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Búadóttir, T., Mascio, O., & Eckroth, J. (2023). Kira: A financial chatbot using ChatGPT and data obfuscation. *Journal of Computing Sciences in Colleges*, *39*(3), 277–294.

Cañizares, P. C., Pérez-Soler, S., Guerra, E., & de Lara, J. (2022). Automating the measurement of heterogeneous chatbot designs. In *Proceedings of the 37th ACM/SIGAPP symposium on applied computing* (pp. 1491–1498). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3477314.3507255.

Carneiro, G., Ferreira, J., Ramalho, F., & Massoni, T. (2023). Similar bug reports recommendation system using BERT. In *Proceedings of the XXXVII Brazilian symposium on software engineering* (pp. 378–387). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3613372.3613396.

Chatterjee, J., & Dethlefs, N. (2023). This new conversational AI model can be your friend, philosopher, and guide ... and even your worst enemy. *Patterns*, *4*(1), Article 100676, URL https://www.sciencedirect.com/science/article/pii/S2666389922003233.

Cheng, Y., Zhang, C., Sangaiah, A. K., Fan, X., Wang, A., Wang, L., & Liu, Y. (2023). Efficient low-resource medical information processing based on semantic analysis and granular computing. *ACM Transactions on Asian and Low-Resource Language Information Processing*, URL https://doi.org/10.1145/3626319.

Cohen, E., & Consens, M. P. (2018). Large-scale analysis of the co-commit patterns of the active developers in github's top repositories. In *Proceedings of the 15th international conference on mining software repositories* (pp. 426–436). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3196398.3196436.

Crawford, K., & Paglen, T. (2021). Excavating AI: the politics of images in machine learning training sets. *AI & SOCIETY*, *36*(4), 1105–1116, URL https://doi.org/10.1007/s00146-021-01162-8.

Dagkoulis, I., & Moussiades, L. (2023). A comparative evaluation of chatbot development platforms. In *Proceedings of the 26th pan-hellenic conference on informatics* (pp. 322–328). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3575879.3576012.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407.

Firat, M. (2023a). How chat GPT can transform autodidactic experiences and open education? URL https://doi.org/10.31219/osf.io/9ge8m.

Firat, M. 2023. What ChatGPT means for universities: Perceptions of scholars and students. 6, 1–22. URL https://doi.org/10.37074/jalt.2023.6.1.22.

Firat, M., & Kuleli, S. (2023). What if GPT4 became autonomous: The auto-GPT project and use cases. *Journal of Emerging Computer Technologies*, *3*(1), 1–6, URL https://doi.org/10.57020/ject.1297961.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, *76*, 378–382, URL https://api.semanticscholar.org/CorpusID:143544759.

Gao, K., Wang, Z., Mockus, A., & Zhou, M. (2023). On the variability of software engineering needs for deep learning: Stages, trends, and application types. *IEEE Transactions on Software Engineering*, *49*(2), 760–776, URL https://doi.org/10.1109/TSE.2022.3163576.

Han, J., Shihab, E., Wan, Z., Deng, S., & Xia, X. (2020). What do programmers discuss about deep learning frameworks. *Empirical Software Engineering*, *25*, 2694–2747, URL https://doi.org/10.1007/s10664-020-09819-6.

Haque, M. U., Iwaya, L. H., & Babar, M. A. (2020). Challenges in docker development: A large-scale study using stack overflow. In *Proceedings of the 14th ACM / IEEE international symposium on empirical software engineering and measurement*. New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3382494.3410693.

Haque, M. A., & Li, S. (2023). The potential use of ChatGPT for debugging and bug fixing. *EAI Endorsed Transactions on AI and Robotics*, *2*, URL https://publications.eai.eu/index.php/airo/article/view/3276.

Härtel, J., Aksu, H., & Lämmel, R. (2018). Classification of APIs by hierarchical clustering. In *Proceedings of the 26th conference on program comprehension* (pp. 233–243). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3196321.3196344.

Ho, C. F., Liew, J., & Lim, T. M. (2024). Exploring optimality and consistency of supervised machine learning algorithms in sentiment analysis. In *Proceedings of the 2024 9th international conference on intelligent information technology* (pp. 48–54). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3654522.3654531.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 50–57). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/312624.312649.

Hosseiny Marani, A., & Baumer, E. P. S. (2023). A review of stability in topic modeling: Metrics for assessing and techniques for improving stability. *ACM Computing Surveys*, *56*(5), URL https://doi.org/10.1145/3623269.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-rank adaptation of large language models. arXiv:2106.09685. URL https://arxiv.org/abs/2106.09685.

Huang, Y., Wang, J., Wang, S., Liu, Z., Wang, D., & Wang, Q. (2021). Characterizing and predicting good first issues. In *Proceedings of the 15th ACM / IEEE international symposium on empirical software engineering and measurement*. New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3475716.3475789.

Izadi, M., Akbari, K., & Heydarnoori, A. (2022). Predicting the objective and priority of issue reports in software repositories. *Empirical Software Engineering*, *27*(2), URL https://doi.org/10.1007/s10664-021-10085-3.

Kallis, R., Di Sorbo, A., Canfora, G., & Panichella, S. (2019). Ticket tagger: Machine learning driven issue classification. In *2019 IEEE international conference on software maintenance and evolution* (pp. 406–409). URL https://doi.org/10.1109/ICSME.2019.00070.

Kirch, W. (2008). Pearson's correlation coefficient. In *Encyclopedia of public health* (pp. 1090–1091). Dordrecht: Springer Netherlands, URL https://doi.org/10.1007/978-1-4020-5614-7_2569.

Kozachek, D. (2023). Investigating the perception of the future in GPT-3, -3.5 and GPT-4. In *C&c '23*, *Proceedings of the 15th conference on creativity and cognition* (pp. 282–287). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3591196.3596827.

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, *18*(17), 1–5, URL http://jmlr.org/papers/v18/16-365.html.

Li, H., Chen, T.-H. P., Shang, W., & Hassan, A. E. (2018). Studying software logging using topic models. *Empirical Software Engineering*, *23*(5), 2655–2694, URL https://doi.org/10.1007/s10664-018-9595-8.

Liu, C., Bao, X., Zhang, H., Zhang, N., Hu, H., Zhang, X., & Yan, M. (2023). Improving ChatGPT prompt for code generation. URL https://arxiv.org/abs/2305.08360.

Lo, C. K. (2023). What is the impact of ChatGPT on education? A rapid review of the literature. *Education Sciences*, *13*, 410, URL https://doi.org/10.3390/educsci13040410.

Lund, B., & Wang, T. (2023). Chatting about ChatGPT: How may AI and GPT impact academia and libraries? *Library Hi Tech News*, *40*, URL https://doi.org/10.1108/LHTN-01-2023-0009.

Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press, URL https://books.google.com.sg/books?id=t1PoSh4uwVcC.

Mathur, A., Vinodh, S. A., & Urolagin, S. (2020). Classifying paintings into movements using HOG and LBP features. In *Proceedings of the 3rd international conference on big data research* (pp. 147–151). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3372454.3372483.

Miller, G. A. (1995). WordNet: a lexical database for english. *Communications of the ACM*, *38*(11), 39–41, URL https://doi.org/10.1145/219717.219748.

Misra, V., Reddy, J. S. K., & Chimalakonda, S. (2020). Is there a correlation between code comments and issues? an exploratory study. In *Proceedings of the 35th annual ACM symposium on applied computing* (pp. 110–117). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3341105.3374009.

Morris, M. R. (2023). *Scientists' perspectives on the potential for generative AI in their fields*: *Technical report*, URL https://arxiv.org/abs/2304.01420.

Nathalia, N., Paulo, A., & Donald, C. (2023). Artificial intelligence vs. Software engineers: An empirical study on performance and efficiency using ChatGPT. In *Proceedings of the 33rd annual international conference on computer science and software engineering* (pp. 24–33). USA: IBM Corp..

Newman, D., Lau, J. H., Grieser, K., & Baldwin, T. (2010). Automatic evaluation of topic coherence. In *Human language technologies: the 2010 annual conference of the North American chapter of the association for computational linguistics* (pp. 100–108). USA: Association for Computational Linguistics.

Olujimi, P. A., & Ade-Ibijola, A. (2023). NLP techniques for automating responses to customer queries: a systematic review. *Discover Artificial Intelligence*, *3*(1), 20, URL https://doi.org/10.1007/s44163-023-00065-5.

OpenAI (2023). GPT-4 technical report. URL https://arxiv.org/abs/2303.08774.

Pérez-Soler, S., Guerra, E., & de Lara, J. (2021). Creating and migrating chatbots with conga. In *Proceedings of the 43rd international conference on software engineering: companion proceedings* (pp. 37–40). IEEE Press, URL https://doi.org/10.1109/ICSE-Companion52605.2021.00030.

Pérez-Verdejo, J. M., Sánchez-García, Á. J., Ocharán-Hernández, J. O., Mezura-Montes, E., & Cortés-Verdín, K. (2021). Requirements and GitHub issues: An automated approach for quality requirements classification. *Programming and Computer Software*, *47*(8), 704–721, URL https://doi.org/10.1134/S0361768821080193.

Rebro, D. A., Chren, S., & Rossi, B. (2023). Source code metrics for software defects prediction. In *Proceedings of the 38th ACM/SIGAPP symposium on applied computing* (pp. 1469–1472). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3555776.3577809.

Shourov, M. M., & Mahmud, I. (2019). PyMannKendall: a python package for non parametric Mann Kendall family of trend tests. *Journal of Open Source Software*, *4*, 1556, URL https://doi.org/10.21105/joss.01556.

Siddiq, M. L., & Santos, J. C. S. (2023). BERT-based GitHub issue report classification. In *Proceedings of the 1st international workshop on natural language-based software engineering* (pp. 33–36). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3528588.3528660.

Silva, C. C., Galster, M., & Gilson, F. (2021). Topic modeling in software engineering research. *Empirical Software Engineering*, *26*(6), URL https://doi.org/10.1007/s10664-021-10026-0.

Song, Y., Mahmud, J., De Silva, N., Zhou, Y., Chaparro, O., Moran, K., Marcus, A., & Poshyvanyk, D. (2023). Burt: A chatbot for interactive bug reporting. In *Proceedings of the 45th international conference on software engineering: companion proceedings* (pp. 170–174). IEEE Press, URL https://doi.org/10.1109/ICSE-Companion58688.2023.00048.

Tian, H., Lu, W., Li, T. O., Tang, X., Cheung, S.-C., Klein, J., & Bissyandé, T. F. (2023). Is ChatGPT the ultimate programming assistant – how far is it? URL https://arxiv.org/abs/2304.11938.

Wan, Z., Xia, X., & Hassan, A. E. (2021). What do programmers discuss about blockchain? A case study on the use of balanced LDA and the reference architecture of a domain to capture online discussions about blockchain platforms across stack exchange communities. *IEEE Transactions on Software Engineering*, *47*(7), 1331–1349, URL https://doi.org/10.1109/TSE.2019.2921343.

White, J., Hays, S., Fu, Q., Spencer-Smith, J., & Schmidt, D. C. (2023). ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. URL https://arxiv.org/abs/2303.07839.

Win, H. M., Wang, H., & Tan, S. H. (2023). Towards automated detection of unethical behavior in open-source software projects. In *ESEC/FSE 2023*, *Proceedings of the 31st ACM joint European software engineering conference and symposium on the foundations of software engineering* (pp. 644–656). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3611643.3616314.

Xie, S. (2022). A comparative study on the quality of english-Chinese machine translation in the era of artificial intelligence. In *AIAM2021*, *2021 3rd international conference on artificial intelligence and advanced manufacture* (pp. 1261–1264). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3495018.3495378.

Xu, L., Sun, S., & Wang, Q. (2016). Text similarity algorithm based on semantic vector space model. In *2016 IEEE/aCIS 15th international conference on computer and information science* (pp. 1–4). URL https://doi.org/10.1109/ICIS.2016.7550928.

Yang, Z., Zhang, H., Yang, C., Li, B., Zhao, X., & Long, Y. (2024). Scale fairness on spectral clustering. In *Proceedings of the 36th international conference on scientific and statistical database management*. New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3676288.3676301.

Yi, X., Wu, D., Jiang, L., Fang, Y., Zhang, K., & Zhang, W. (2022). An empirical study of blockchain system vulnerabilities: modules, types, and patterns. In *ESEC/FSE 2022*, *Proceedings of the 30th ACM joint European software engineering conference and symposium on the foundations of software engineering* (pp. 709–721). New York, NY, USA: Association for Computing Machinery, URL https://doi.org/10.1145/3540250.3549105.

Zhang, N., Huang, Q., Xia, X., Zou, Y., Lo, D., & Xing, Z. (2022). Chatbot4QR: Interactive query refinement for technical question retrieval. *IEEE Transactions on Software Engineering*, *48*(4), 1185–1211, URL https://doi.org/10.1109/TSE.2020.3016006.

Zhang, N., Wang, J., He, K., Li, Z., & Huang, Y. (2019). Mining and clustering service goals for restful service discovery. *Knowledge and Information Systems*, *58*(3), 669–700, URL https://doi.org/10.1007/s10115-018-1171-4.