Dear Editors and Reviews,

Thank you for your letter and for the comments concerning our manuscript entitled "Prompt Enhance API Recommendation: Visualize the User's Real Intention behind this Query". We have clarified the concerns of all reviewers in the current version. The main improvements are summarized as follows:

For easier reading. In the following reply letter, the reviewer suggested that we use blue characters, the reply uses black characters, and the revised part of the manuscript uses red characters.

We gratefully appreciate your consideration of our manuscript and look forward to receiving your feedback. Thank you and best regards.

Sincerely,

Yong Wang[1,2]

Linjun Chen[1]

Cuiyun Gao[3]

Yingtao Fang[1]

Yong Li[4]

[1]School of computer and information, Anhui Polytechnic University, Wuhu, Anhui, China

[2]Anhui Artificial Intelligence Laboratory, Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, High-tech Zone, Hefei,, Anhui, China

[3]School of Computer Science and Technology,Harbin Institute of Technology, Shenzhen, Guangdong, China

[4]College of Computer Science and Technology, Xinjiang Normal University, , Urumqi, Xinjiang, China.

January 3, 2024

---

1. Response to Reviewer #1:

We thank reviewer_1's constructive comments and guidance for the revision of this paper. The following is a point-to-point response.

Introduction:

(1) Comment#1: In this section, I see a critical point. The authors briefly describe the procedure to evaluate PTAPI. The description reports that several graduate students participated in a "survey" to provide feedback.

A user study like this one is useful. In the rest of the paper, there is no mention of this study.
Are the metrics calculated later on produced by the students?
If so, the authors should provide a questionnaire or an experience report and a description of the query tested, the students with their education experience (e.g. if they know JAVA, if they use JAVA, if they use API and libraries).
Answer: Thank you for your valuable suggestions. The approach we proposed does require user case studies to verify its effectiveness. For user study, we use a section to supplement this. We selected 12 students from the first author 's university to help us conduct user case studies. Based on the experience of 12 students, we divided them into three groups, one group using network resource search, one group using BIKER tool, and the last group using our tool PTAPI. The experimental results are analyzed from the correctness and time-consuming of the experimental results. See Section 6, Page 13 for details.

Motivating Example:
(2) Comment#2: I can personally agree with the author's claim: "By observing the question posts in SO..." but for a scientific paper, it is imperative to report references and formally justify it. Furthermore, the work is built on top of this perception.
" For users, they prefer that when they enter the above questions, there is a prompt telling them to sort the collection, array, or other data". This claim is reasonable for students or newbies. Is it still valid for experts?
As for the previous comment, is there some study supporting this claim?

Answer: Thank you for your valuable advice. About you put forward we lack of references, we study this. We added the corresponding references in the paper. First of all, in the paper [18,19], it is described that there are problems in the quality of SO question posts, and the quality of question posts needs to be improved. econdly, our method is mainly aimed at programming novices, because they are more prone to problems in the programming process. Studies have shown that [7], even programming experts may seek similar tips and guidance when facing problems outside a specific field. This shows that my point of view is not only applicable to novices and students, but also to a wider user group including experts.

[7] Li, Z., Wu, J., Li, M.: Study on key issues in api usage. Journal of software 29(6), 1716–1738 (2018), In Chinese.
[20] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza and D. Fullerton, "Improving Low Quality Stack Overflow Post Detection," 2014 IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, 2014, pp. 541-544, doi: 10.1109/ICSME.2014.90.
[21] K. Liu, X. Chen, C. Chen, X. Xie and Z. Cui, "Automated Question Title Reformulation by Mining Modification Logs From Stack Overflow," in IEEE

Transactions on Software Engineering, vol. 49, no. 9, pp. 4390-4410, 1 Sept. 2023, doi: 10.1109/TSE.2023.3292399.

Methodology:
(3) Comment#3: I recognize different critical aspects:
Why did the authors decide to crawl StackOverflow? Crawling is an effective yet very fragile method to get data. A modest modification in the HTML can lead to an erroneous outcome.
There are stable datasets for this purpose:
• SOTorrent https://empirical-software.engineering/sotorrent/
• Archive.org https://archive.org/download/stackexchange/, scroll to StackOverflow
I can also cite works that employed Stackoverflow data:
•PostFinder https://www.sciencedirect.com/science/article/abs/pii/S0950584920301361
• PROMPTER https://ieeexplore.ieee.org/abstract/document/6976143
• Post2Vec https://ieeexplore.ieee.org/abstract/document/9469219
So, why not consider stable datasets?
"Additionally, a separate testing dataset consisting of 413 datasets was created". Is this a typo? Furthermore, is this the ground truth of your analysis? If so, the description is poor since the results rely on these 413 questions/datasets.
Concerning the baselines, the decision is fair. I wonder whether the tool exploited this dataset for tests only. If the tool can employ different datasets, the author should describe it better. It will be a relevant aspect.
Sec 4.1 "Additionally, a separate testing dataset consisting of 413 datasets was created". Datasets? Questions?

Answer: Thank you very much for pointing out these negligences, and also thank you for providing these data, which will be of great help to our future research. Our data is the same as the BIKER source, using a text corpus that downloads the official data dump of SO (published in: Dec 9th, 2017). We carefully read the manuscript and corrected these errors in the subection 4.1 Dataset Description. The revisions of the original manuscript are as follows: See subsection 4.1, Page 8 for details.

To test and evaluate the effectiveness of PTAPI, we reuse BIKER. BIKER downloaded the official data dump of SO (published in: Dec 9th, 2017) as its own SO text corpus. Next, we need an SO problem library to implement the second stage of prompt template generation and the third stage of similar problem retrieval. Because SO is an open question and answer website, the quality of the answers is often not high. Therefore, these data need to be strictly screened. It has the following screening criteria: 1) All scores in the library should be positive. 2) Each question has at least one API answer, and the score of the answer also needs to be positive. Finally, we need to select a test dataset to evaluate the effectiveness of PTAPI. This requires

higher quality of data, and we need to ensure that the API in the answer is the correct API. The screening criteria for the test database are as follows: 1) The score of the problem itself needs to be greater than 5 points. 2) There must be an answer in the question, and the score is positive. 3)The data is not in the SO problem library. In this way, a preliminary dataset is obtained. After manual labeling, a test dataset containing 413 problems and their ground truth API is obtained.

---

(4) Comment#4: Related to this comment, the authors explained that they employed the BIKER's dataset to perform the experiments.
The contribution of the crawler appears to be useless.
Furthermore, since the tests are based on the BIKER's dataset, what will happen to the metrics if a different dataset is exploited?

Answer: Thank you for your comments. In the subsection4.1 Dataset Description, we explain the origin of BIKER data and its screening process. It can be seen that the quality of BIKER data is relatively high, which is suitable for the experiment of our method. Thank you for your comments. In the subsection, we explain the origin of BIKER data and its screening process. It can be seen that the quality of BIKER data is relatively high, which is suitable for the experiment of our method. We also conducted experiments on the data sets used by NLP2API and RACK. We found that this dataset only supports class-level recommendations, so our team collected the top 40 data from this dataset and re-given the correct answer through the query. We use these 40 questions to test the formed test data set. The results show that the accuracy of the first 10 questions is only 0.45 at the method level. We further studied the experimental results and found that this is related to the quality of the post and the emergence of new APIs. In the future, we will continue to collect various data for experiments.

---

(5)Comment#5: Concerning the similarity measure, I wonder why the authors defined themself the similarity mechanism. Why not use a well-established tool like Lucene https://lucene.apache.org/core/?If the authors defined a custom parametrization to obtain better results is an added value to the work, why not describe it more deeply?

Answer: Thank you for your advice. Lucene is an open source full-text search engine. It is a core library of search and can provide powerful search functions. The similarity mechanism used in our paper is also a classical information retrieval method in software engineering, and has been used in many papers.

Ye, X., Shen, H., Ma, X., Bunescu, R., & Liu, C. (2016, May). From word

embeddings to document similarities for improved information retrieval in software engineering. In Proceedings of the 38th international conference on software engineering (pp. 404-415).
Huang, Q., Xia, X., Xing, Z., Lo, D., Wang, X.: Api method recommendation without worrying about the task-api knowledge gap. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, pp. 293–304 (2018)
Zhou, Y., Yang, X., Chen, T., Huang, Z., Ma, X., Gall, H.: Boosting api recommendation with implicit feedback. IEEE Transactions on Software Engineering 48(6), 2157–2172 (2022)

(6) Comment#6: Why do the authors focus only on the title to generate the prompt? Indeed, the title is relevant to finding a good post, why not consider the corpus of the question?

Answer: We know that the title is usually a concise summary of the problem, which can quickly convey the core content of the query. In many cases, the title itself is enough to reflect the main direction of the problem. Moreover, the titles in our dataset are generally of high quality and can provide a good overview of the problem content. The corpus of the question is indeed a good study, which has been studied by relevant personnel [1], however the experimental results are not very good. Nevertheless, using the corpus of questions is the right direction. We will consider it in future research.

[1] Zhang, Q., Zhang, J., Feng, T., Xue, J., & Huang, P. (2022, December). APIGAN: Learning to Recommend Java API in Real-Time Using Generative Adversarial Networks. In Journal of Physics: Conference Series (Vol. 2400, No. 1, p. 012055). IOP Publishing.

(7) Comment#7: It is not clear to me how long is going to be the initial input of the user.
What is the minimum size of input? What is the maximum?
The size is relevant since the tool will merge the input with the title which best describes the problem.

Answer: We do not set the initial input time of the user. When the user enters the query, PTAPI will give the answer in about 15s. In the experiment, we use the continuous bag-of-words model, where the window size is set to 5, so the minimum and maximum values we input are 5 and 100, respectively.

(8) Comment#8:The authors test only the effect of putting the title before or after the prompt.

Answer: Thank you for your comments. In the experiment of this paper, we first consider selecting the top-ranked prompts as templates separately, and find that selecting the top-ranked prompts can always get the best results. Then we consider the influence of the position of the template on the experiment and whether the selection of multiple templates affects the experiment. At present, prompt learning is a hot topic. In the design of prompts, approaches such as connection, fusion, and substitution are commonly used. The initial assumption of our work is that when the user enters a vague question, the model can give a prompt that meets the user 's wishes. However, due to the influence of the dataset, it is not always possible to get the real intention, so we retain the initial user 's input for the connection selection prompt template. Obviously, the selected prompt template can better express the user 's true intention, so we will study the influence of different weights of initial input and prompt template setting on the experiment in the future.

(9) Comment#9: The authors should describe better how the template is combined with the original input. Is the post of the proposed title filtered out at this stage? In other words, if a user picks a title, can the related post be found? Are the titles still used to find the pertinent post?

Answer:Our prompt templates are generated based on the question library, which is not intersected with our test library. So, we are filtering out posts with proposed titles. If a user selects a title, we are still able to find related posts. Since SO is a public Q&A platform where all people can ask questions, there are bound to be many similar questions which are simply different in description. The title is also used to find related posts, this is the flow of our method so it won't change.

(10) Comment#10:The only filter applied is the score of the answer. It is fine, but some questions can arise, for example:

- Do the authors consider the version of the API? (Java 1.5; 1.8; 11)
- Is there some mechanism to filter deprecated API or vulnerable?

Answer:Addressing the issue of API versioning is one of the challenges in API method recommendation, where there may be subtle but critical differences between different versions of a library or API. The issue of versioning is an important one that can be easily overlooked in API recommendation. This manuscript does not have a separate design for version compatibility, which does limit the usefulness of API

recommendation. We refer to the literature on API versions in this field, and propose some methods for API version compatibility and filtering unpopular APIs[3,4,5], which may be implemented in subsequent work.

Ideas for approaching the API versioning problem are as follows:

1), Version Matching: When making API recommendations, consider the API version used by the target system or project. Based on the requirements of the API version, an API method compatible with the target version is selected for recommendation. This can be achieved by comparing the version requirements of the API method with the API version of the target project.

2), Version Adaptation: When recommending API methods, take into account the differences between different versions and provide adapters or compatibility layers that adapt to different versions. This allows adapters to handle the differences between versions of the API method to ensure that the recommended method will work properly on the target version.

3), version compatibility assessment: Before making API recommendations, assess the compatibility of the target project's API version. Evaluate the compatibility between the API method and the target version, including the usability of the method, differences in parameters, etc. Based on the evaluation results, select API methods compatible with the target version for recommendation.

4), Dynamic Version Detection: Dynamically detect the API version of the target system or project at runtime and select a suitable API method for recommendation based on the detection results. This can be achieved by using a version detection tool or querying the API information of the target system.

[1] Hao Xia, Yuan Zhang, Yingtian Zhou, Xiaoting Chen, Yang Wang, Xiangyu Zhang, Shuaishuai Cui, Geng Hong, Xiaohan Zhang, Min Yang, and Zhemin Yang. 2020. How Android developers handle evolution-induced API compatibility issues: a large-scale study. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE '20). Association for Computing Machinery, New York, NY, USA, 886–898.

[2] L. Qi, W. Lin, X. Zhang, W. Dou, X. Xu and J. Chen, "A Correlation Graph Based Approach for Personalized and Compatible Web APIs Recommendation in Mobile

APP Development," in IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 6, pp. 5444-5457, 1 June 2023, doi: 10.1109/TKDE.2022.3168611.
[3] Lianyong Qi, Houbing Song, Xuyun Zhang, Gautam Srivastava, Xiaolong Xu, and Shui Yu. 2021. Compatibility-Aware Web API Recommendation for Mashup Creation via Textual Description Mining. ACM Trans. Multimedia Comput. Commun. Appl. 17, 1s, Article 20 (January 2021), 19 pages.

---

(11) Comment#11: How does the tool exploit the official documentation? (Javadoc? Crawling the documentation website?)

Answer:PTAPI directly crawls the data on the official documentation website to build a corpus of official questions and answers. API full name extracted from SO posts are reordered using the API description information in the documentation.

---

(12) Comment#12: Are third-party libraries supported in some way?

Answer: The answers in the posts in Stack Overflow will contain APIs in third-party libraries. When the user enters the problem, PTAPI will find similar posts in Stack Overflow, and get a candidate API sequence from these posts. However, JAVA official documents do not directly support third-party libraries, and PTAPI need to use official documents to rearrange candidate API sequences. The final API ranking score will take the average of the SO post similarity score and the official document similarity score. Therefore, the API of third-party libraries will be recommended, but generally ranked lower. We discuss it in the threat to validity. see subsection 7.2, page 15 for more details.

---

(13) Comment#13: In the implementation details section, the concept of parameter is introduced. Are the authors referring to the possible recommendation size? That is the number of possible recommendations.

Answer: Thank you for your comments. Parameter refers to the number of similar problems used to retrieve top-k. We have revised the manuscript. see subsection 4.4, page 9 for more details.

We run our experiments on a computer with AMD Ryzen 7 2700X 3.7GHz, 32GB RAM. When searching for top-k similar problems to detect candidate APIs, we note that the number of retained similar problems has an important impact on the

recommendation results. If you retain too few similar problems, you may miss the target API required by the user, and if you retain too much, you may introduce too much noise data. In subection 5.1, we set the parameter to 50 (retaining 50 similar questions) to compare with BIKER and BRAID. In subsection 5.2, we manually adjust the size of the parameters to verify the impact of the number of similar problems on the experiment. In subsection 5.3, according to the user 's input, we feedback to the user 10 prompts (10 prompts can be found in the probability of meeting the user 's intentions) as a user selection prompt template.

---

(14) Comment#14: Why does this "parameter" value range between 30 and 1000?
If the authors conducted preliminary tests to retrieve these values, it would be interesting to show in the work.

Answer: We have added an analysis of the effect of parameter variations on the experimental results in the revised version. see subsection 5.2, page 11 for more details.

By analyzing the data in the table, we can find that the performance of PTAPI will first improve and then decrease with the increase of parameters. This is because in the beginning, we kept fewer similar questions, and the scope of the search was not large enough, which caused a lot of important relevant API information to be lost. With the increase of similar problems, the accuracy of the model will gradually improve. However, when the performance of the model reaches the optimal value, the introduction of similar problems will lead to the increase of a large number of noisy data, and the performance of the model will gradually decrease.

---

(15) Comment#15: For the RQ3 Table 5, the authors experimented to understand the effects of putting random templates in the query.
How many experiments do the authors conduct?
Why did the authors decide on a random number between 3 and 10?

Answer: First of all, our initial idea was to get more accurate results by selecting multiple prompts. If the user chooses “I like apple.”, “I like oranges.”, “I like watermelon.”. We hope that through these three tips, the recommendation system can give the answer of “fruit platter “. For API recommendation, when the user selects “sort the collection.”, “Find the collection.”, “Reverse the set.”. In these three prompts, we hope that the recommendation system can give the answer of java.util.Collections. In previous experiments, we have concluded that in the top 10 recommendation

prompts, the accuracy rate has been in line with the user 's choice basis. In this experiment, we set three different random numbers within 10, and each group was subjected to 10 experiments. The final result was the average of 10 results. We have supplemented the manuscript, as shown in Comment 16

(16) Comment#16:The authors raise an interesting point concerning the bag-of-words. If the tool employed the cosine similarity and the IDF, how can the authors exploit the template order? This question is related to RQ3; I trust the evaluation, but it is unclear how technically the position affects the results.

Answer: The model of our experiment uses the Continuous bag of words (CBOW), which is a neural network model for generating word vectors. It was proposed by Tomas Mikolov et al.1[1] in 2013 and is often used in natural language processing tasks, such as word embeddings. The basic idea of CBOW is to predict the word itself given the context of a word (that is, other words around it). Specifically, for a given context, the CBOW model attempts to predict the target word by other words in the context. This context can be determined by defining a window size, and the words in the window are considered as context words.

In the experiment, we set the window of the continuous bag-of-words model to 5. Although the word order problem has always been the shortcoming of the bag-of-words model, in the paper, we combine the original input and the prompt into a new sentence. Experiments show that the difference in the order of the two sentences has a small impact on the results. We reformulated RQ3 as follows : see subsection 5.3, page 12 for more details.

[1] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Table 4 The influence of the position of the template on the experiment was suggested

	S@1	**S@3**	**S@10**	**MAP**	**MRR**
30	**0.561**	**0.782**	**0.864**	**0.633**	**0.679**
$\widehat{30}$	0.540	0.760	0.845	0.618	0.659
40	**0.552**	**0.804**	**0.898**	**0.641**	**0.682**
$\widehat{40}$	0.542	0.775	0.862	0.622	0.666
50	**0.538**	**0.794**	**0.906**	**0.632**	**0.673**
$\widehat{50}$	0.525	0.775	0.881	0.617	0.661
100	**0.528**	**0.809**	**0.935**	**0.629**	**0.667**
$\widehat{100}$	0.511	0.833	0.918	0.614	0.656

1000	**0.462**	**0.814**	**0.973**	**0.571**	**0.618**
$\widehat{1000}$	0.453	0.801	0.954	0.567	0.611

Table 5 The influence of using multiple prompt templates on the experiment

	S@1	**S@3**	**S@10**	**MAP**	**MRR**
30_{method}	0.552	0.782	0.847	0.623	0.665
50_{method}	**0.542**	0.789	0.884	0.631	0.670
45_{class}	0.678	**0.903**	**0.959**	0.758	0.787
50_{class}	0.682	**0.906**	**0.969**	**0.763**	0.791

Table 4 shows the experimental results at the method level (at the class level, the sequence of the prompt template does not affect the experimental results), where K indicates that the prompt template is in front and the user ’s input problem is in the back. ˆK indicates that the user ’s input problem is ahead and the prompt template is behind. We found that regardless of the number of parameters, the result of putting the prompt template in front is always the best. Table 5 is to verify whether the more templates are added, the better the experimental results. In this experiment, we first get 10 prompts based on the user ' s input. Then, three different prompts were randomly selected within the range of 10 prompts to combine with the original input and form a new input. The experiment was repeated 10 times in each group, and the final results were averaged.

After analyzing the above results, first of all, for the study of the position of the prompt template, we initially thought that the order of words in the bag-of-words model would not affect the experimental results. After subsequent research, we found that we use the continuous word bag model, and set the size of the window to 5, the original input is connected to the prompt word, and the two sentences will affect each other. The experimental results show that it is better to put the prompt in front of the text. Secondly, for using multiple prompt templates, our initial guess is that the more tips, the better the experimental results. Because we use the bag of words model, the more tips can provide more keywords to query, which can enhance the expression of words, so as to achieve better results. Meanwhile, multiple prompts may provide multiple different keywords, and multiple keywords can be combined into the user ’s true intention. But in the end, it was found that the experimental results did not have obvious advantages. Later, we found that the experimental results of multiple templates have a particularly high quality requirement for user input problems. As shown in Figure 1 (b), we encountered the problem of ’using Java to sort content’. Due to the poor quality of the problem, multiple templates will give keywords with

very different meanings, resulting in a poor recommendation effect. The experimental results show that, first, putting the prompt in front of the sentence position will get better results. Second, the number of input prompts is not the more the better.

---

(17) Comment#17: The threats to validity section shall be extended.
Discussions about third-party libraries, crawling the websites, vulnerable and outdated APIs are interesting.

Answer: Thank you for your advice. We have extended the threat to effectiveness section in the manuscript as follows: see subsection 7.2, page 15 for more details.

In this paper, we generate prompts and form new problems based on user input, and combine SO posts and API documents for similarity calculation to obtain the final API ranking. Due to the variety of APIs and the rapid update of their versions, PTAPI is not very comprehensive in considering third-party libraries and API quality issues. Meanwhile, the model using a regularized expression to extract hyperlinks and API names may be error-prone. However, this paper mainly studies the impact of prompt learning on API recommendation. The training dataset and the test dataset we used are of high quality and the collected time is similar, so these threats do not affect our experiment.

---

(18) Comment#18: Finally but still relevant, there is not any replication package or accessible repository to test the tool or study the results.

Answer: Thank you for your advice. In the future, we will package our project and publish it to GitHub.

---

(19) Comment#19: Fig. 2 Recommendationl -> recommendation
Sec 6.1 The two words in the bag of words are similar in grammar, but there may be a big difference in grammar. ???

Answer: Thank you very much for pointing out these errors. We went over the manuscript carefully and corrected these errors in presentation. The revisions are as follows:
Fig.2 Recommendationl idea diagram → Recommendation idea diagram
Sec6.1 The two words in the bag of words are similar in grammar, but there may be a big difference in grammar, so only using the context of the word is not enough to

distinguish between queries. → The bag-of-words model only focuses on the presence or absence of words, and cannot capture the contextual relationship between words. Context is important for understanding the meaning of text.

2. Response to Reviewer #2:

We would like to thank reviewer_2 for recognizing our work and for his valuable comments.

(1) Comment#1: Comparison with Baselines:** The paper focuses on API recommendation but fails to compare PTAPI with existing baselines in the field, such as MAPO, GAPI, or FOCUS. A comparison with these approaches is essential to establish the relevance and effectiveness of PTAPI.
Comparison with existing baselines like FOCUS is not addressed, which is crucial for understanding the novelty and effectiveness of PTAPI.

Answer: We are very grateful for your advice ! At present, the research on API recommendation is mainly divided into two types : one is to query similar projects based on the existing code context to recommend subsequent APIs. Another question based on user input to get the API answer. This paper belongs to the second research direction. FOCUS, MAPO and GAPI belong to the first category of research directions.

(2) Comment#2:Prompt Template and Prompt Learning:** The proposal to introduce a prompt template for API recommendation lacks clarity. A comprehensive explanation of the approach, including prompt template creation, is necessary for reader understanding. The concept of prompt learning is introduced abruptly; its relevance should be clarified from the outset. It remains unclear if the combination of the prompt template with the user description is a one-time process or can undergo several iterations.

Answer: Thank you for your suggestion. We add prompt learning, prompt template and the role of prompt learning in the introduction. In this paper, the combination of prompt template and user description is a one-time process. We have made changes in the manuscript as follows: see Section 1 Introduction, page 2 for more details.

To deal with this uncertainty, we adopt the method of prompt learning to help users express their real needs more clearly. Prompt learning [18, 19] is a new paradigm in natural language processing. It can design a series of prompts to guide the model to better understand and process tasks. Among them, the template is a framework for building prompts. Through the design of the template, the potential of the upstream pre-training model can be tapped.

In this paper, we propose a task-based user intent visualization approach PTAPI(use Prompt Template to enhance API recommendation performance). This approach introduces SO as a third-party information source, and uses similar post problems in SO as a prompt template for task description, so that developers can understand their real intentions. That is, the developer enters the problem to be solved, finds similar problems from the SO post, and the developer selects the most similar problem as the prompt template, and combines the two sentences into a new input through a connection process.

---

(3) Comment#3: Language Model Construction:** The process of constructing a language model for similarity calculation requires a more in-depth explanation. Details on technology used, challenges faced in dataset creation, and the method supporting the search, especially concerning Stack Overflow posts, are crucial for clarity.

Answer: Thank you for your suggestion, our revised draft gives a more in-depth explanation of the process of building a language model for similarity calculation. The following is the revision: see subsection 3.2 Generate prompt template, page6 for more details.

In this subsection, first, we need to use SO posts to build a corpus for calculating the similarity between queries and SO posts or API descriptions. Get the text content from the SO posts in the HTML page and process it. Because some SO posts contain long code fragments, this will increase the burden of training. Therefore, we delete the posts containing long code fragments, retain the posts containing short code fragments, and use the NLTK package to label the sentences. The NLTK package can identify the part of speech of each word in the sentence. Secondly, we need to use Word2Vec to train a word embedding model, which is the basic model for measuring word similarity. Then, we construct the word IDF (inverse document frequency) vocabulary based on the SO corpus. The IDF of the word represents the reciprocal of the number of posts containing the word. Among them, the more posts a word appears in, the less likely it is to carry important semantic information, so its IDF value is lower. Each word in the text is abbreviated into a root form. For instance, the stem of 'running', 'runs', and 'run' is 'run', and the stem of 'am', 'is', and 'are' is 'be' Words with the same root have the same IDF value. Finally, because the amount of text in the SO corpus is much larger than that in the API document, the words in the API document directly

select the word embedding model and the IDF vocabulary.

---

(4) Comment#4: Data Extraction and Regular Expressions:** The choice to extract data from Stack Overflow HTML pages rather than using dumps needs justification. Additionally, the use of regular expressions to extract hyperlinks and API names may be error-prone and should be addressed cautiously.

Thank you very much for your advice! Our data comes from SO 's official data dump (published in: December 9, 2017), and we have revised this error in subsection 4.1 of the original. In addition, we have filtered the quality of posts in the dataset. At the same time, we retrieve the top K similarity questions to detect candidate APIs. We set the K value to be relatively large. Although the regularization expression extracts links and API names may be error-prone, the target API can still be obtained with a high probability. Finally, we clarify in the threat that the regularized expression will bring a certain threat.

subsection 4.1 Dataset Description

To test and evaluate the effectiveness of PTAPI, we reuse BIKER. BIKER downloaded the official data dump of SO (published in: Dec 9th, 2017) as its own SO text corpus. Next, we need an SO problem library to implement the second stage of prompt template generation and the third stage of similar problem retrieval. Because SO is an open question and answer website, the quality of the answers is often not high. Therefore, these data need to be strictly screened. It has the following screening criteria: 1) All scores in the library should be positive. 2) Each question has at least one API answer, and the score of the answer also needs to be positive. Finally, we need to select a test dataset to evaluate the effectiveness of PTAPI. This requires higher quality of data, and we need to ensure that the API in the answer is the correct API. The screening criteria for the test database are as follows: 1) The score of the problem itself needs to be greater than 5 points. 2) There must be an answer in the question, and the score is positive. 3) The data is not in the SO problem library. In this way, a preliminary dataset is obtained. After manual labeling, a test dataset containing 413 problems and their ground truth API is obtained.

subsection 6.2 Threats to Validity

In this paper, we generate prompts and form new problems based on user input, and combine SO posts and API documents for similarity calculation to obtain the final API ranking. Due to the variety of APIs and the rapid update of their versions, PTAPI is not very comprehensive in considering third-party libraries and API quality issues.

Meanwhile, the model using a regularized expression to extract hyperlinks and API names may be error-prone. However, this paper mainly studies the impact of prompt learning on API recommendation. The training dataset and the test dataset we used are of high quality and the collected time is similar, so these threats do not affect our experiment.

---

(5) Comment#5: PTAPI framework description:** Figure 3 requires a detailed description. Currently, it is presented as a sketch, lacking the necessary details for readers to comprehend its content fully.

Answer: Thank you for your suggestion. For the detailed description of Figure 3, we have made modifications in the original draft, as shown below:

Subsection 3.3 Generate prompt template

In this subsection, PTAPI generates prompts based on the user 's input, and the newly generated prompts can better express the user 's true intention. First, through the similarity language model constructed in the previous section, PTAPI finds several titles that are most similar to the user input from the SO problem library. We use formula 1 to calculate the similarity between the query and the SO problem library:

Subsection 3.4 generate relevant API recommendations

In the previous subsection, we find the prompt template according to the user input description, and combine the generated template with the user input description into a new input statement. Continue the operation of the previous stage to calculate the similarity between the newly generated sentence and the sentence in the SO problem library. Through calculation, we find k problems that are most similar to the input from the SO problem library. After retrieving the top-k similar questions, PTAPI uses two heuristic rules to retrieve APIs from each question. One detects the full name of the corresponding API method from the hyperlink using a regularized expression by using each hyperlink in each answer. Another uses a dictionary to store the names of all APIs from official documents. Then, PTAPI checks the plain text contained in the HTML tags in each answer. If the API in the plain text matches the one in the dictionary, it is marked as a candidate API.

Evaluation Concerns:

(6) Comment#6: RQ1 needs rephrasing for clarity, aligning with the standard that experiments answer research questions. The meaning and interpretation of "how far" in the context of the experiment remain unclear.

Comparison results with BIKER and BRAID lack qualitative discussion, necessitating additional insights into the obtained results.

Answer: Thank you for your comments! We have re-stated RQ1 in the original and the following revisions have been made:

It is worth noting that PTAPI has the most obvious increase in S @ 1 at the method level and class level. This means that PTAPI can recommend more than half of the correct APIs in the S @ 1 results. This is consistent with our expectation that prompts can help users better choose APIs. We find that BRAID accepts the user 's initial input and the results recommended by the existing API as input on the basis of BIKER. It uses user history selection as feedback information and uses active learning technology to establish a new API recommendation model. This makes BRAID may be overly dependent on historical data and vulnerable to initial data bias. Meanwhile, the choice of user history may not fully reflect their long-term intentions. PTAPI can give users prompts in real time, and these prompts can also best reflect the user 's most real intentions at present, so our method can be greatly improved in accuracy.

PTAPI has a great advantage compared with other baselines, which indicates that the display information has a positive effect on improving the accuracy of the recommendation system.

---

(7) Comment#7: The reference to "top 10 most similar problems as templates" requires further explanation.
The term "parameter" lacks context and clarification regarding its specific mention in the evaluation section.

Answer: Thank you for your comments! We have made changes to the original text in subsection 4.4 Implementation Details, explaining why we chose 10 prompt templates and the explanation of ' parameters '.

We run our experiments on a computer with AMD Ryzen 7 2700X 3.7GHz, 32GB RAM. When searching for top-k similar problems to detect candidate APIs, we note that the number of retained similar problems has an important impact on the recommendation results. If you retain too few similar problems, you may miss the target API required by the user, and if you retain too much, you may introduce too much noise data. In subsection 5.1, we set the parameter to 50 (retaining 50 similar questions) to compare with BIKER and BRAID. In subsection 5.2, we manually adjust the size of the parameters to verify the impact of the number of similar

problems on the experiment. In subsection 5.3, according to the user 's input, we feedback to the user 10 prompts (10 prompts can be found in a large probability of the user 's intention) as a user selection prompt template.

(8) Comment#8: The claimed effectiveness of PTAPI at both the method and class levels needs clearer supporting evidence in the experimental evaluation.

Answer: Thanks for your comments! First, we rephrase RQ1 to clarify why PTAPI can perform better at the method and class levels. Secondly, we add a section of user research in the manuscript to prove the effectiveness of PTAPI in practice through user research. Please see Section 6, page 13 for more details.