AutoPRTitle: A Tool for Automatic Pull Request Title Generation

Ivana Clairine Irsan*, Ting Zhang*, Ferdian Thung, David Lo and Lingxiao Jiang
School of Computing and Information Systems, Singapore Management University
Email: {ivanairsan, tingzhang.2019, ferdianthung, davidlo, lxjiang}@smu.edu.sg

***Abstract*—With the rise of the pull request mechanism in software development, the quality of pull requests has gained more attention. Prior works focus on improving the quality of pull request descriptions and several approaches have been proposed to automatically generate pull request descriptions. As an essential component of a pull request, pull request titles have not received a similar level of attention. To further facilitate automation in software development and to help developers draft high-quality pull request titles, we introduce AutoPRTitle. AutoPRTitle is specifically designed to generate pull request titles automatically. AutoPRTitle can generate a precise and succinct pull request title based on the pull request description, commit messages, and the associated issue titles. AutoPRTitle is built upon a state-of-the-art text summarization model, BART, which has been pre-trained on large-scale English corpora. We further fine-tuned BART in a pull request dataset containing high-quality pull request titles. We implemented AutoPRTitle as a stand-alone web application. We conducted two sets of evaluations: one concerning the model accuracy and the other concerning the tool usability. For model accuracy, BART outperforms the best baseline by 24.6%, 40.5%, and 23.3%, respectively. For tool usability, the evaluators consider our tool as easy-to-use and useful when creating a pull request title of good quality.**

Source code: https://github.com/soarsmu/Auto-PR-Title.

Video demo: https://tinyurl.com/AutoPRTitle.

***Index Terms*—Pull Request, GitHub, Summarization, Pre-trained Models**

I. Introduction

Collaborative coding platforms, such as GitHub, widely adopt the pull request mechanism. Developers make a local copy from the main repository and make changes in their local copy (i.e., branches). After they are satisfied with their changes, they can request to merge their branch to the main repository by opening a new pull request. A pull request consists of a pull request title, a pull request description (optional), and several commits. Similar to how an issue quality is important for bug triaging, the quality of a pull request also makes an impact on the decision of whether a pull request gets merged [1]. As an essential component of a pull request, the title has received less research attention compared to the description. Several prior works [2], [3] have proposed different approaches to generate pull request descriptions based on the information from the commits, i.e., commit messages and comments in code changes.

In our earlier work [4], as the first work on pull request title generation, we built a dataset named PRTiger containing pull requests from 495 popular GitHub repositories. We formulate the pull request title generation task as a one-sentence summarization task. The source sequence is the concatenation of the pull request description (if any), commit messages, and the related issue titles (if any). The target sequence is the pull request title. We utilized BART [5], which is a pre-trained sequence-to-sequence model that has achieved a remarkable performance in several summarization tasks. It achieved the best performance among the approaches that we evaluated.

In this paper, we present AutoPRTitle, which is a tool for automatically generating pull request titles. This is a demonstration paper accompanying our above-mentioned full research paper [4]. We implemented AutoPRTitle as a web application. AutoPRTitle takes a new pull request link, a pull request description, and related issue links from the user as inputs. AutoPRTitle can extract the commit messages from the given pull request link. If a related issue link is provided, AutoPRTitle will also extract the issue title. The commit messages, the issue title, and the pull request description are then fed to the BART model. AutoPRTitle will then provide a pull request title suggestion.

To evaluate our tool, including the underlying model, we performed an automatic evaluation to measure model accuracy, and a manual evaluation to measure the tool usability. We compared BART with existing approaches for software artifact generation. We consider two approaches for similar tasks as the baselines, i.e., PRSummarizer for pull request description generation [2], and iTAPE for issue title generation [6]. The experimental results from the automatic evaluation show that the fine-tuned BART [5] produced the best performance. It achieves the highest ROUGE-1, ROUGE-2, and ROUGE-L F1-scores of 47.22, 25.27, and 43.12, which outperform the best baseline by 24.6%, 40.5%, and 23.3%, respectively. For the manual evaluation, we asked 7 evaluators to rate our tool. They consider our tool to be easy-to-use and useful when drafting a good pull request title.

II. Compared Methods

As pre-trained models have produced remarkable performances in many tasks (e.g., [7], [8]), we utilize BART (`facebook/bart-base`), which has demonstrated its ability to solve text summarization tasks in the natural language processing field [5]. We also presented some existing approaches that solve similar tasks, i.e., PRSummarizer [2] for pull request description generation and iTAPE [6] for issue

*Both authors contributed equally to this research.

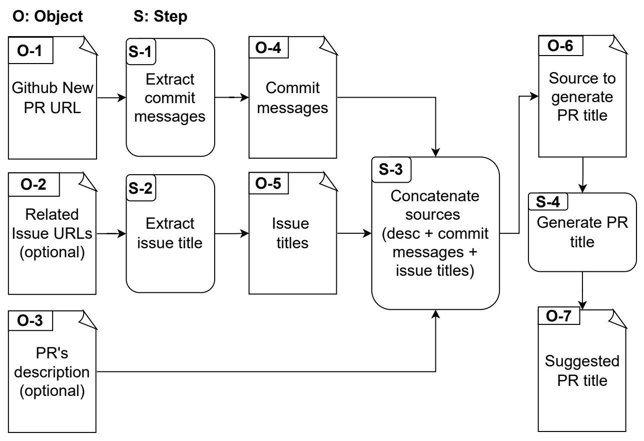

O: Object
S: Step
O-1 Github New PR URL
S-1 Extract commit messages
O-4 Commit messages
O-2 Related Issue URLs (optional)
S-2 Extract issue title
O-5 Issue titles
S-3 Concatenate sources (desc + commit messages + issue titles)
O-3 PR's description (optional)
O-6 Source to generate PR title
S-4 Generate PR title
O-7 Suggested PR title

Fig. 1. The architecture of AUTOPRTITLE

title generation as our baseline methods. We briefly describe these approaches as follows:

- *BART* [5] adopts a standard Transformer [9] architecture. BART was pre-trained by (1) corrupting the source sequence with noise functions and then (2) learning to reconstruct the original text. It was pre-trained in the same corpora as RoBERTa [10]. The pre-training corpora cover 160GB of text, including news, books, stories, and web text. BART achieved state-of-the-art performance in several text generation tasks, such as dialogue response generation on CONVAI2 [11].
- *PRSummarizer* [2] leverages the pointer generator [12] to handle the out-of-vocabulary (OOV) issue. Besides, to mitigate the gap between the loss function and the evaluation metrics, i.e, ROUGE scores, PRSummarizer adopts reinforcement learning to optimize the performance in terms of ROUGE scores directly.
- *iTAPE* [6] was originally proposed to generate issue titles. The source sequence is the issue description, and the target sequence is the issue title. iTAPE leverages a sequence-to-sequence model. To cope with the OOV issue, it combines two techniques, i.e., a lightweight tagging method and a copy mechanism [13] with a pointer generator.

III. Tool Architecture

A. Overview

The architecture of AUTOPRTITLE is presented in Figure 1. AUTOPRTITLE is built upon simplicity. It is packaged as a web application that could be easily deployed using a Docker container. This tool has three input sources to generate a pull request title suggestion: a new pull request URL (O-1), related issues' URLs (O-2), and a pull request description (O-3). The new pull request URL (O-1) is a mandatory input, as it is needed to extract (S-1) the commit messages (O-4). On the other hand, related issue URLs (O-2) are not mandatory as a pull request is not necessarily a fix to an issue. If related issue URLs are given, AUTOPRTITLE will extract

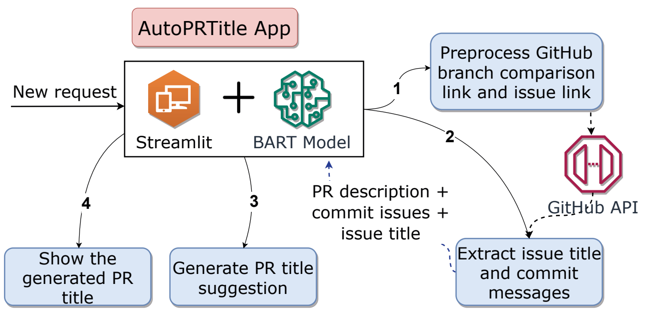

AutoPRTitle App
New request
Streamlit
BART Model
1
2
3
4
Preprocess GitHub branch comparison link and issue link
GitHub API
Extract issue title and commit messages
PR description + commit issues + issue title
Generate PR title suggestion
Show the generated PR title

Fig. 2. The workflow of AUTOPRTITLE

the corresponding issue titles (S-2, O-5). While a pull request description (O-3) is an optional input, users are encouraged to fill it in to get a better-generated title. All of the three sources are then concatenated (S-3) before being used as a textual source sequence (O-6) to generate a suggested pull request title (S-4, O-7).

B. Implementation Details

Figure 2 shows the detailed workflow of AUTOPRTITLE. AUTOPRTITLE is developed on top of `Streamlit` [14], an open-source app framework that is built as a Python3 library. It enables us to integrate the fine-tuned BART model with a web interface seamlessly. AUTOPRTITLE uses a BART Model as the underlying model to generate the pull request title. The BART Model is loaded once at start up, and will be called every time there is a request to generate a pull request title.

First, AUTOPRTITLE preprocesses the given new pull request link (e.g., https://github.com/microsoft/vscode/compare/main...TylerLeonhardt/copy-after-action) and issue links. For both links, AUTOPRTITLE then replaces the "github.com" in the link with "api.github.com/repos" to create a GitHub API's request URL. After the GitHub API's request URLs are obtained, AUTOPRTITLE makes REST API calls to these URLs.

Second, AUTOPRTITLE extracts the issue title and commit messages from the GitHub API response. The response associated with the GitHub branch comparison link contains a list of commits ready to be merged to the target branch. We extract commit messages from these commits' details. For the response associated with the issue link, the corresponding issue details are returned. We will extract the issue title from these issue details. At this point, as no pre-processing step is done toward the pull request description, all the input sources are ready to be concatenated. AUTOPRTITLE proceeds to concatenate the pull request description, the commit messages, and the issue titles.

Third, using the BART Model, AUTOPRTITLE generates the pull request title based on the concatenated text. While the model is running in the background, the web User Interface (UI) will show an animation to indicate that the process is still running.

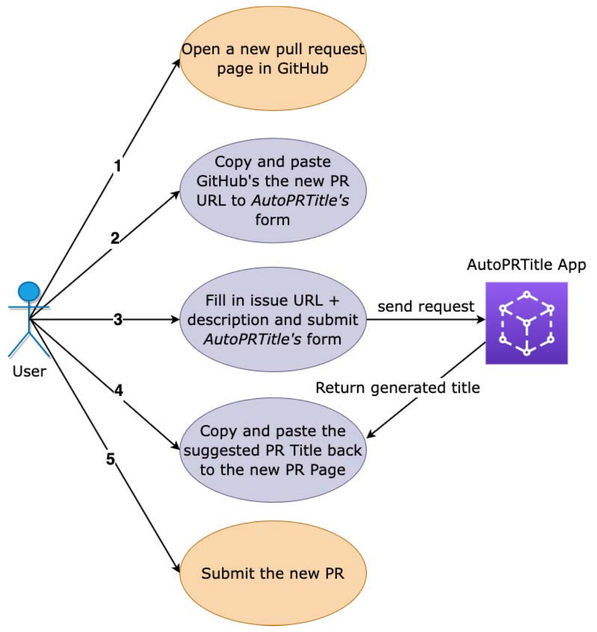
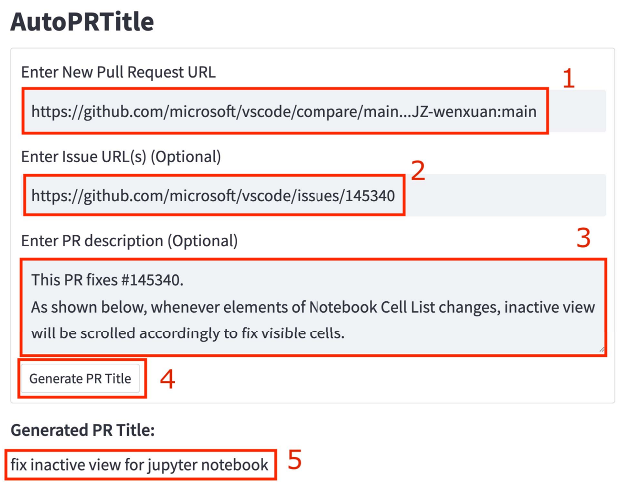

Open a new pull request page in GitHub
Copy and paste GitHub's the new PR URL to *AutoPRTitle's* form
Fill in issue URL + description and submit *AutoPRTitle's* form
Copy and paste the suggested PR Title back to the new PR Page
Submit the new PR
User
AutoPRTitle App
send request
Return generated title
1
2
3
4
5

Fig. 3. The scenario of using AUTOPRTITLE


AutoPRTitle
Enter New Pull Request URL
Enter Issue URL(s) (Optional)
Enter PR description (Optional)
Generate PR Title
Generated PR Title:
1
2
3
4
5

https://github.com/microsoft/vscode/compare/main...JZ-wenxuan:main

https://github.com/microsoft/vscode/issues/145340

This PR fixes #145340.
As shown below, whenever elements of Notebook Cell List changes, inactive view will be scrolled accordingly to fix visible cells.

fix inactive view for jupyter notebook

Fig. 4. The user interface of AUTOPRTITLE

Fourth, the generated title is displayed in the AUTOPRTITLE's web UI, and is ready to be copied to GitHub's new pull request page.

C. Deployment

We provide a Dockerfile in our replication package[1] for easy deployment. Users only need to build the image and run it as a container on every machine that supports Docker usage. This practice will enable non-Python developers to use our apps, as no Python knowledge (i.e., environment set up, execution, etc.) is needed to deploy AUTOPRTITLE in their machines.

IV. USAGE SCENARIO

We illustrate how a user can use AUTOPRTITLE in Figure 3. We also show the user interface of AUTOPRTITLE in Figure 4. Firstly, a user opens a new pull request page on GitHub: they can open a pull request page in a repository and click the *New Pull Request* button. Secondly, the user needs to copy and paste the new pull request URL to an input field (the red box 1 in Figure 4). They can then put the issue links, which are resolved by the new pull request, to another input field (the red box 2 in Figure 4). Furthermore, they can provide a description of this pull request on another input field (the red box 3 in Figure 4). Since the new pull request URL is the only mandatory input, the user can leave the other inputs empty. After clicking *Generate PR Title* (the red box 4 in Figure 4), the generated pull request title will be output below it (the red box 5 in Figure 4). The user can copy this generated title to the new pull request page and submit the pull request.

In Figure 4, we show an example of the generated pull request title produced by AUTOPRTITLE on an existing pull request in `Microsoft/vscode` repository.[2] This example contains all three input sources. The original pull request title is `Fixes #145340`. This title is not informative and asks for an extra effort from pull request reviewer to check what the issue is actually about. The pull request title generated by AUTOPRTITLE is `fix inactive view for jupyter notebook`, which summarized the related commit messages and the linked issue title. AUTOPRTITLE can save pull request reviewers' time since they can understand the purpose of the pull request just by reading the generated pull request title.

[1]https://github.com/soarsmu/Auto-PR-Title

[2]The pull request is available at https://github.com/microsoft/vscode/pull/146125

V. EVALUATION

A. Dataset

We built a dataset to facilitate automatic pull request title generation. We considered a diverse set of repositories. We first collected pull request from the Top-100 most-starred, Top-100 most-forked GitHub repositories regardless of programming languages. We further included the Top-100 most-starred repositories that are written primarily in any of the following programming languages: JavaScript, Python, Java, C, and C++. In total, we have a list of 700 repositories. After filtering out the duplicate repositories, we are left with 578 distinct repositories. We crawled the pull request of these repositories that were published before the year of 2022. We then cleaned the dataset, and 83 distinct repositories were removed because all the corresponding PRs were filtered out. The detailed steps can be found in our research paper [4]. In the final dataset, we have 43,816 pull request from 495 distinct GitHub repositories. We split them in the ratio of 8:1:1 for training, validation, and testing purposes.

B. Evaluation Setup and Results

We conducted two sets of evaluation: (1) *Model accuracy*: we calculated the ROUGE scores [15] of the generated pull

TABLE I
AUTOMATIC EVALUATION RESULTS ON THE TEST DATASET

Approach	ROUGE-1	ROUGE-2	ROUGE-L
BART [5]	**47.22**	**25.27**	**43.12**
PRSummarizer [2]	37.91	17.99	34.98
iTAPE [6]	32.23	12.91	29.31

request titles with the original high-quality pull request titles. (2) *Tool usability*: we invited evaluators to evaluate our tool usability.

(1) Model accuracy. We report ROUGE-N (N=1,2) and ROUGE-L F1-scores. ROUGE-1 and ROUGE-2 measure the overlap of uni-grams (1-grams) and bi-grams (2-grams), respectively. On the other hand, ROUGE-L measures the longest common sub-sequence between the reference summary (i.e., the original pull request title) and the generated summary (i.e., the model generated pull request title). The formulas to calculate ROUGE-N (N=1,2) scores are as follows:

$$R_{rouge-n} = \frac{Count(overlapped\_N\_grams)}{Count(N\_grams \in reference\_summary)}$$

$$P_{rouge-n} = \frac{Count(overlapped\_N\_grams)}{Count(N\_grams \in generated\_summary)}$$

$$F1_{rouge-n} = 2 \times \frac{R_{rouge-n} \times P_{rouge-n}}{R_{rouge-n} + P_{rouge-n}}$$

Note that $overlapped\_N\_grams$ refers to the n-grams that appear in both the reference summary and the generated summary. ROUGE F1-score ($F1_{rouge-n}$) seeks the balance between ROUGE Recall ($R_{rouge-n}$) and ROUGE Precision ($P_{rouge-n}$). Thus, we use ROUGE F1-scores as the main metrics in our automatic evaluation.

Result. Table I shows the model accuracy. It indicates that BART outperforms both prior approaches. BART outperforms PRSummarizer, which is a pull request description generation approach, by 24.6%, 40.5%, and 23.3% in terms of ROUGE-1, ROUGE-2, and ROUGE-L F1-score, respectively.

(2) Tool usability. We invited 7 people from our research group. They all have at least 5 years of experience in programming and at least 4 years of experience using GitHub. We give them access to our tool. We also provided an example on how to use our tool. However, we did not provide a list of existing pull requests for them. They are free to test our tool with any pull request in public repositories. The rationale is that they can select repositories which they are more familiar with so that they can better judge the quality of the recommended pull request title. We then asked them to provide two scores regarding the tool's usability. Each score ranges from 1 to 5, indicating strongly disagree, slightly disagree, neutral, slightly agree, and strongly agree. The two scores relate to two aspects, i.e., (1) *Ease-of-use:* the tool is easy to use; (2) *Usefulness:* The tool is useful and can help me to come up with a good pull request title.

Result. The averaged score in terms of *ease-of-use* is 4.3. While the average score about *usefulness* is 4.4. It shows that the evaluators perceive our tool to be useful and easy to use.

VI. Related Work

Our work belongs to the broader research topic of automatically generating software artifacts. In the past years, various tools have been proposed to generate different types of artifacts automatically. For instance, AutoComment was proposed to automatically generate source code comments by leveraging Stack Overflow posts [16]. These automatic approaches, especially the ones related to summarization, are typically aimed to improve the quality of software artifacts and save practitioners' time [6]. Among the existing summarization tasks, the two most similar tasks to ours are Stack Overflow post title generation and issue title generation. CODE2QUE [17] introduced a tool to generate a Stack Overflow post title based on the given code snippet. CODE2QUE was intended to help developers write high-quality Stack Overflow question titles. Besides generating titles, CODE2QUE also recommends related Stack Overflow questions based on the similarity between code snippets. The other related tool is iTIGER [18], which is aimed to automatically generate an issue title based on the given issue description. iTIGER has been implemented as a Userscript and provides the suggested title directly on GitHub's new issue page. AUTOPRTITLE differs from these tools as it requires several input sources: a pull request description, commit messages, and linked issue titles. In comparison, the two relevant tools only require one type of input, i.e., code snippets or issue descriptions. Thus, we implemented AUTOPRTITLE as a web application to better capture the different sources of information in a pull request.

VII. Conclusion and Future Work

In this work, we demonstrate an automatic pull request title generation tool, AUTOPRTITLE. Upon creating a new pull request, developers need to copy and paste the new pull request URL into our web application. They can also add the associated issue links and a pull request description to the web application. After clicking on the *Generate PR Title*, our tool will return a suggested title. Developers can then copy this title back to the new pull request page and submit it. AUTOPRTITLE uses BART, a state-of-the-art summarization approach. In our evaluation, BART outperforms existing baselines. Moreover, evaluators agree that AUTOPRTITLE is easy to use and useful in generating good pull request titles. In the future, we would like to improve AUTOPRTITLE by leveraging more types of information, such as code comments in the code changes.

Acknowledgment

This research / project is supported by the National Research Foundation, Singapore, under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu, "Wait for it: Determinants of pull request evaluation latency on github," in *2015 IEEE/ACM 12th working conference on mining software repositories*. IEEE, 2015, pp. 367–371.

[2] Z. Liu, X. Xia, C. Treude, D. Lo, and S. Li, "Automatic generation of pull request descriptions," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 176–188.

[3] S. Fang, T. Zhang, Y.-S. Tan, Z. Xu, Z.-X. Yuan, and L.-Z. Meng, "PRHAN: automated pull request description generation based on hybrid attention network," *Journal of Systems and Software*, vol. 185, p. 111160, 2022.

[4] T. Zhang, I. C. Irsan, F. Thung, D. Han, D. Lo, and L. Jiang, "Automatic pull request title generation," 2022 IEEE 38th International Conference on Software Maintenance and Evolution (ICSME), Research Track.

[5] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Association for Computational Linguistics, 2020, pp. 7871–7880.

[6] S. Chen, X. Xie, B. Yin, Y. Ji, L. Chen, and B. Xu, "Stay professional and efficient: Automatically generate titles for your bug reports," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2020, pp. 385–397.

[7] T. Zhang, D. P. Chandrasekaran, F. Thung, and D. Lo, "Benchmarking library recognition in tweets," in *2022 IEEE/ACM 30th International Conference on Program Comprehension (ICPC)*, 2022, pp. 343–353.

[8] T. Zhang, B. Xu, F. Thung, S. A. Haryono, D. Lo, and L. Jiang, "Sentiment analysis for software engineering: How far can pre-trained transformer models go?" in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 70–80.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[10] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[11] E. Dinan, V. Logacheva, V. Malykh, A. Miller, K. Shuster, J. Urbanek, D. Kiela, A. Szlam, I. Serban, R. Lowe *et al.*, "The second conversational intelligence challenge (convai2)," in *The NeurIPS'18 Competition*. Springer, 2020, pp. 187–208.

[12] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, R. Barzilay and M. Kan, Eds. Association for Computational Linguistics, 2017, pp. 1073–1083.

[13] J. Gu, Z. Lu, H. Li, and V. O. K. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[14] "Streamlit," http://streamlit.io, (Accessed on 06/23/2022).

[15] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[16] E. Wong, J. Yang, and L. Tan, "Autocomment: Mining question and answer sites for automatic comment generation," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2013, pp. 562–567.

[17] Z. Gao, X. Xia, D. Lo, J. Grundy, and Y.-F. Li, "Code2Que: a tool for improving question titles from mined code snippets in stack overflow," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 1525–1529.

[18] T. Zhang, I. C. Irsan, F. Thung, D. Han, D. Lo, and L. Jiang, "iTiger: an automatic issue title generation tool," 2022. [Online]. Available: https://arxiv.org/abs/2206.10811