2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC) | 979-8-3503-3750-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICPC58990.2023.00038

Properly Offer Options to Improve the Practicality of Software Document Completion Tools

Zhipeng Cai*†, Songqiang Chen*†, Xiaoyuan Xie*‡
*School of Computer Science, Wuhan University, China
zhipengcai@whu.edu.cn, i9schen@gmail.com, xxie@whu.edu.cn

***Abstract*—With the great progress in deep learning and natural language processing, many completion tools are proposed to help practitioners efficiently fill in various fields in software document. However, most of these tools offer their users only one option and this option generally requires much revision to meet a satisfactory quality, which hurts much practicality of the completion tools. By finding that the beam search model of such tools often generates a much better output at relatively high confidence and considering the interactive use of such tools, we advise such tools to offer multiple high-confidence model outputs for more chances of offering a good option. And we further suggest these tools offer dissimilar outputs to expand the chance of including a better output in a few options. To evaluate our whole idea, we design a clustering-based initial method to help these tools properly offer some dissimilar model outputs as options. We adopt this method to improve nine completion tools for three software document fields. Results show it can help all the nine tools offer an option that needs less revision from users and thus effectively improve the practicality of tools.**

***Index Terms*—software document completion, tool practicality, output selection, clustering, natural language processing**

I. Introduction

Software documents, such as the bug report and Stack Overflow post, help practitioners comprehend and deal with various works in developing activities. With the great advances in deep learning and natural language processing (NLP) techniques, researchers have proposed various neural models to generate content for many fields in the software documents, such as the descriptions of pull requests [1], titles of Stack Overflow posts [2], [3], titles of issue reports [4], etc [5]–[8]. One outstanding usage of these models is to help practitioners efficiently draft good documents. To fulfill this usage, recent studies implement completion tools based on such models [9]–[11]. When writing documents, tool users can use the result offered by these tools to quickly fill in the target fields. If these tools can offer a good result, they will save much time for document authors, improve document quality, and thus benefit relevant development tasks.

Most of these tools are powered by a Sequence-to-Sequence (Seq2Seq) neural model with a beam search decoder. And they **only offer users the result getting the Top-1 confidence** from the model **as the only option** [9]–[11]. However, there is often **a large discrepancy between this only option and its ground truth**. For example, the Top-1 pull request description and bug report title offered by their corresponding generating models have only about 30% similarity to the ground truth on average w.r.t. widely-used ROUGE-L F1 score [1], [4], [9]. In this case, users still have to **do much revision to this poor option** before they use it. This impairs much practicality of completion tools.

One popular solution to the above problem is to redesign the model in completion tools to generate a better Top-1 result and thus reduce users' revision need [3], [9], [12]. But it is usually tool-specific and time-consuming. Instead, by considering the interactive usage scenario of these tools, we recognize a more straightforward and generic yet neglected new solution: **letting these tools offer users multiple potential options to choose from**. In fact, this idea has been taken by some interactive tools for other development tasks, such as fault localization [13] and API recommendation [14]. When a tool offers a proper number of options, it will not put too much reading burden on users but **provide more chances to offer users a better result**. As such, in this work, we first encourage software document completion tools to offer multiple options to improve their practicality.

In the context of offering multiple options, the problem goes to which options to offer. In particular, to avoid putting much reading burden on users, we can offer only a limited number of options. Thus, it is crucial to select the most helpful options to offer. Given the nature of the beam search Seq2Seq model in current completion tools, these tools can easily get a batch of high-confidence outputs from their model to offer. By studying the high-confidence outputs generated by the models, we found there is usually an output much better than the poor Top-1, as shown in Case A and B in Table I, but there are also many poor outputs similar to the poor Top-1. On these bases, we propose a preliminary idea to **offer multiple dissimilar high-confidence outputs from the model as options**. Since many outputs from the model can be similarly poor, this idea helps a tool try more distinct outputs so that the **chance to offer a better option can be largely expanded**. As a reminder, this is a preliminary idea to reveal the benefit of offering multiple options and the need of output selection. We will further enhance it in the future.

To study the potential of our whole idea, we build an initial method based on clustering to help each completion tool offer multiple dissimilar options. It packs high-confidence outputs of the model into groups and picks one from each group to offer. We use this method to improve nine tools for 3 popular fields, i.e., Stack Overflow post (SOP) titles [3], bug report (BR) titles [4], and pull request (PR) descriptions [1]. Results show that offering 10 dissimilar options with this method will contribute an increase of about 16 to the ROUGE-L F1 that these tools obtain. When offering 20 options with this method, the growth can go up to 20, which nearly doubles the original performance

†Equal contribution and co-first authors.
‡Corresponding author.

2643-7171/23/$31.00 ©2023 IEEE
DOI 10.1109/ICPC58990.2023.00038

of some tools offering only one option. Such outcomes suggest our idea can effectively reduce the revision need for users and thereby improve the practicality of various completion tools.

In summary, most existing works focus on improving model design for better software document completion results. In this paper, we reveal that offering multiple proper model outputs as options and optimizing selection of outputs can be a new way orthogonal to the update of model design to improve the practicality of completion tools. Following this way, we propose to offer dissimilar beam search outputs as a preliminary solution. To evaluate our whole idea, we design an initial method based on clustering to help nine completion tools offer a few dissimilar options. With the help of this method, all these tools can offer a much better option, which confirms the good potential of our idea in reducing the revision need for users and improving the practicality of various software document completion tools.

II. Motivating Examples and Our Preliminary Idea

Offering multiple options is widely taken as an efficient idea by interactive tools like the fault-localization tool [13] and API recommender [14] to make up for their imprecision and boost their practicality. With the tradeoff to have the users read a bit more options, the chance to offer a more satisfying option can be greatly enhanced. This improves the practicality of a tool in general. Additionally, the Seq2Seq model with a beam search decoder keeps multiple prefixes with top confidence at every decoding step (the number of kept prefixes is known as "beam size") and is widely adopted in software document completion tools [1]–[11]; thus, a completion tool can easily offer multiple options by offering some outputs kept at the last decoding step by its model without the need to modify the model. These facts show offering multiple options is **potentially helpful and easy to deploy** on software document completion tools. Next, let us elaborate our idea and its expected effect with examples.

Aiming to improve the tools, we first study the cases where the only option from a tool (i.e., its Top-1 model output) needs much revision and thus impairs much practicality of that tool. We perform this study on the nine completion tools considered in our evaluation, which will be introduced in Sect. IV-A. We then found two major cases where tools work poorly. The first case is illustrated by Case A in Table I. In this case, the model wrongly gives the top confidence to a totally wrong result. The limited model performance is blamed for this case. The other situation is illustrated by Case B in Table I. The Top-1 result in this case describes the issue symptom; while the ground truth title focuses more on the latent cause of the issue. That is, the only option given by the tool can be acceptable but not exactly match the desire of the user. We reckon the reason may be that the distinct contexts (e.g., projects) or users may have distinct yet hard requirements for the document style (e.g., describing the issue symptoms or causes), but the input barely shows this custom yet hard desire. In this case, it can be challenging for a tool to hit the precise answer by giving only one option.

Fortunately, based on our observation, although the model in current tools cannot provide a precise Top-1 output in the poor cases, **it often generates a much better output item at fairly high confidence** and this better item can be found in the items kept by beam search. For instance, there is a nearly correct output ranking at 11 in Case A and an output in the desired style ranking at 13 in CaseB. Thus, by offering multiple outputs kept by beam search as options, the tool **may give the users access to a better result at their acceptable cost of glancing over a few options**. This should help the tools become more practical. Thus, in this paper, we first advise the completion tools to offer multiple model outputs as options to improve their practicality.

But to avoid putting too much extra burden on users, we can still offer **only a few options**. Therefore, it is essential to select the most helpful outputs to offer. As shown in Cases A and B, simply offering the top outputs returned by beam search can be sub-optimal since it often gives many very similar outputs [15]. Since most current poor Top-1 are very dissimilar to its ground truth, the similar outputs tend to be poor as well and solely take up the chances for the other potentially helpful outputs. Taking Cases A and B as examples, many of their top outputs are quite similar but poor. When only 10 options are allowed to offer, the tool would still miss the chance to offer the better output. Thus, it deserves further study to select the helpful outputs to offer.

To address this problem, we further advise completion tools to **pack similar outputs and offer several dissimilar options**. This idea can expand the chance to offer better options in both poor situations. More specifically, in the cases where the tools have fairly limited performance (e.g., Case A), this idea avoids the helpless cost of the tool to offer similar incorrect options and leaves the chance to the other dissimilar potentially correct outputs. In the cases where a tool cannot sense a precise style requirement (e.g., Case B), this idea introduces options in more styles for the users to choose from. Taking Cases A and B as examples, the top 15 outputs can be packed into 3 and 5 groups (the outputs in the same color belong to a group), respectively. Next, we select a representative (e.g., the underlined top item) from every group to offer. As a result, many less helpful items (i.e., wrong outputs in Case A and outputs in a similar style in Case B) will be skipped; while other potentially helpful items can be offered early. In this case, even when only 5 options are allowed, the better output item will still be offered to improve practicality of tools. Besides, even if a better item is incorrectly discarded, one item quite similar to it will be offered. Taking Case C in Table I as an example, its Top-1 output is fairly good and a better item is similar to the Top-1. Even if we offer the Top-1 rather than the better item, the users still need only a little effort to further revise the offered result. In general, *this idea makes a tradeoff to average the revision need of all cases into a small amount*. Though it may miss some more accurate options, as most current Top-1 results require much revision, it should reduce the revision need for users most time and make software document completion tools more practical.

III. Initial Clustering-based Output Selection

In this work, we design an initial method based on clustering to realize our whole preliminary idea. Its details can be further adjusted, but it should basically reflect the potential of our idea in improving the practicality of software document completion

TABLE I: Three Software Document Completion Examples of Different Characteristics

Case A: Poor Top-1 with Totally Wrong Content.	**Case B: Poor Top-1 Missing Style Requirement.**	**Case C: Acceptable Tradeoff to Offer Dissimilar Options.**
GT Label: GridBagLayout gridwidth doesn't work as expected	**GT Label:** service does not support epsv mode	**GT Label:** text truncation doesn't work in touchablehighlight in a flatlist
Top 15 Model Outputs: *(with descending confidence)*	**Top 15 Model Outputs:** *(with descending confidence)*	**Top 15 Model Outputs:** *(with descending confidence)*
GridBagLayout taking up same number of columns *(Top-1)*	ftp uses passive mode,access is slow *(Top-1)*	text truncation doesn't work with numberoflines *(Top-1)*
GridBagLayout taking up the same number of columns	ftp uses passive mode,access is slow by clusterip	text truncation doesn't work with numberoflines prop
GridBagLayout takes up the same number of columns	ftp uses passive mode,access is very slow	text truncation doesn't work for numberoflines
GridBagLayout takes up same number of columns	ftp uses passive mode,access takes too long to get results	text truncation doesn't work in touchablehighlight in flatlist **[BETTER]**
Why does GridBagLayout take up so much space?	ftp uses passive mode,access is slow to get results	text truncation doesn't work for numberoflines prop
GridBagLayout taking up same number of columns?	ftp is slow by clusterip	text truncation doesn't work for numberoflines in flatlist
GridBagLayout columns take up the same number of columns	ftp uses passive mode,access is too slow	text truncation with numberoflines in touchablehighlight in flatlist
GridBagLayout taking up the same number of columns?	ftp uses passive mode,access to clusterip is slow	text truncation in touchablehighlight in flatlist doesn't work
Why does GridBagLayout take up same number of columns?	ftp uses passive mode,access takes 3 minutes to get results	text truncation in touchablehighlight in flatlist
GridBagLayout columns take up same number of columns	ftp uses passive mode by clusterip is slow	text truncation not working with numberoflines
GridBagLayout gridwidth not working as expected **[BETTER]**	why ftp takes about 3 minutes to get results?	text truncation doesn't work with numberoflines in flatlist
GridBagLayout columns taking up same number of columns	why ftp uses passive mode by clusterip is slow?	text truncation not working with numberoflines prop
Why does Java GridBagLayout take up so much space?	ftp does not support epsv mode **[BETTER]**	text truncation doesn't work for touchablehighlight in flatlist
Why does the GridBagLayout take up so much space?	ftp uses passive mode by clusterip is very slow	text truncation doesn't seem to work with numberoflines
GridBagLayout columns taking up the same number of columns	ftp uses passive mode,access takes about 3 minutes	text truncation not working in touchablehighlight in flatlist
SOP Title Generation for https://stackoverflow.com/questions/32120258	*BR Title Gen. for https://github.com/kubernetes/kubernetes/issues/40137*	*BR Title Generation for https://github.com/facebook/react-native/issues/25284*

Note: 1) The output with the highest ROUGE-L F1 similarity to the ground truth in the top 15 outputs is marked with **[BETTER]**. 2) The color tells the similarity among outputs. The outputs of the same color have a ROUGE-L similarity above 65 and can be grouped. 3) Due to the space limitation, we do not include the inputs in the table. But they can be accessed via the link that we give in the bottom of each case.

tools. Specifically, it helps a tool select dissimilar ones in the tool's high-confidence model outputs as the proper options to offer. Algorithm 1 is the overall process of this method. It first collects all outputs from the inner model of a tool at a specific beam size (L1). To prepare adequate high-confidence outputs to select from, we adopt a beam size of 200 following [12] as a preliminary exploration. Afterward, it computes the pair-wise ROUGE similarity matrix for the collected outputs (L2). Next, it performs agglomerative clustering to group similar outputs following our idea. To avoid packing too dissimilar results into a group, it first conducts clustering at a minimum intra-group similarity threshold θ (L3). We tentatively set θ as 40 based on our trial on the validation sets from [1], [3], [4]. If the returned groups are fewer than the allowed options, it runs clustering again to obtain sufficient groups (L4-L6). After that, it collects the output with top confidence from each group as the potential options to offer (L7). Finally, it further picks the k outputs with the highest confidence from the previously collected outputs to offer (L8). In the last two steps, now it identifies better outputs following the confidence of the model as our initial solution.

Algorithm 1 an initial output selection method with clustering.

Input: the Seq2Seq model M, input text i, beam size $beamSize$, number of the options to offer k, intra-group similarity threshold θ.
Output: a list of k selected proper options for the tool to offer.

1: $initRes \leftarrow M\,(i, beamSize)$
2: $simMat \leftarrow$ GetPairWiseROUGELF1SimilarityMatrix $(initRes)$
3: $groups \leftarrow$ AgglomerativeClusterT $(initRes, simMat, \theta)$
4: **if** sizeof $(groups) < k$ **then**
5: $\quad groups \leftarrow$ AgglomerativeClusterN $(initRes, simMat, k)$
6: **end if**
7: $topRes_0 \leftarrow$ PickResultxTopConfidenceFromGroups $(groups)$
8: $topRes \leftarrow$ PickKResultsxTopConfidenceFromList $(topRes_0, k)$
9: **return** $topRes$

IV. Evaluation Setup

A. Target Tasks and Tools

We conduct evaluation on the completion tools for 3 popular software document fields, i.e., the Stack Overflow Post (SOP) title [3], the bug report (BR) title [4], and the pull request (PR) description [1]. For every field, we first consider the tool built with the **model** proposed in the corresponding **literature**. To study the generalizability of our idea across models, we further build tools using two pre-trained models, **BART** [16] and **T5** [17], which also show good ability on the target fields [3], [9]. We carefully follow all the setups in the relevant literatures to train the models and build corresponding tools with them. As a result, we have nine completion tools to improve. We apply a specific option offering method to offer the final options.

B. Baseline Option Offering Methods

We introduce the following 5 baseline option offering methods to compare with:

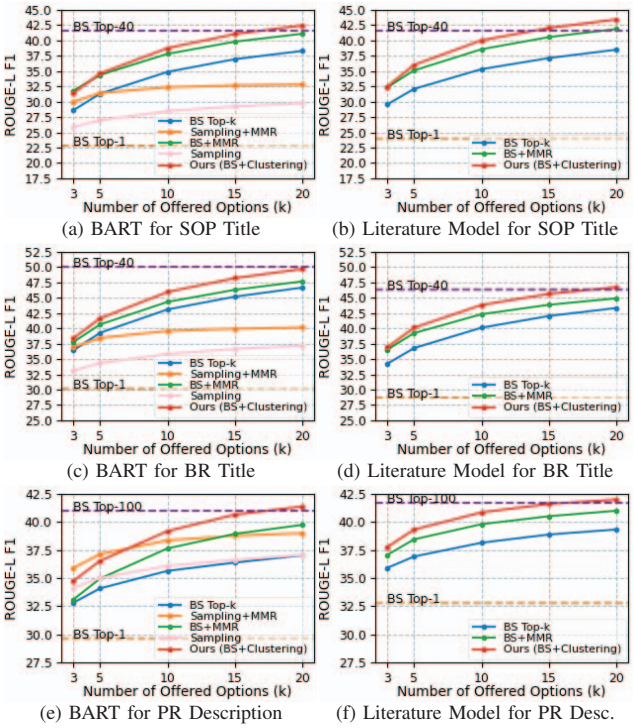
• *BS Top-1:* The original option strategy of current tools, i.e., to offer the Top-1 output of beam search as the *only* option.

• *BS Top-k:* One straightforward idea to offer *multiple* options with a beam search Seq2Seq model mentioned in Sect. II, i.e., to offer k outputs with the Top-k confidence from the model.

• *Sampling + MMR:* To obtain *multiple* possible Stack Overflow questions for each code snippet, Zhang et al. [12] recently propose a model with sampling decoding [18] instead of beam search decoding and design maximal marginal ranking (MMR) to generate k diverse outputs. We have different goals to them, i.e., they design a solution to a problem that explicitly requires diverse outputs; while we try dissimilar outputs to make up for model imprecision on various software document fields. But as their method can also produce multiple diverse outputs, we use it as a baseline. But this method changes the decoding method of model and thus it is not easy to quickly deploy it on existing models. As such, in this preliminary evaluation, *we only deploy it on BART and T5 given the friendly libraries to build them.*

• *BS + MMR:* As both components of the method from Zhang et al. contribute to the selection of diverse outputs, we build a baseline by combining beam search with their MMR selection, i.e., to use MMR instead of our clustering to pick the k options. Since this baseline does not change the beam search decoding used by most current models, we can apply it to all models.

• *Sampling:* Sampling [18] is a state-of-the-art decoding strategy in the NLP community to generate diverse model outputs. We also follow [12] to build a baseline that directly adopts the Sampling strategy to generate k outputs as the options. Since this baseline also changes the decoding method, *we only apply it to the tools based on the BART and T5 models as well.*



(a) BART for SOP Title (b) Literature Model for SOP Title

(c) BART for BR Title (d) Literature Model for BR Title

(e) BART for PR Description (f) Literature Model for PR Desc.

Fig. 1: Performance Comparison of Using Different Methods to Select and Offer Outputs on Different Models and Tasks

C. Performance Metric

We follow existing works to adopt the ROUGE [19] metric to measure the similarity between a result and the ground truth [1], [3], [4]. It varies in [0,100]. When multiple (k) options are offered, we follow [12] to report the highest similarity of these options. Due to the space limit, we only report the most general ROUGE-L F1 score. The other ROUGE scores can be found in our artifact [20]. We follow [12] to set k as 3, 5, 10, 15, and 20. Offering more than 20 options is not considered as it may add too much burden on the tool users to read so many options.

V. Preliminary Evaluation Results and Analysis

Due to limited space, we only visualize the results on BART and the literature models in Fig. 1 in this paper. The results on T5 imply a similar conclusion. *The complete numerical results and the replication package can be found in our artifact [20].*

As presented in Fig. 1, we first found that our initial method (red lines) can offer a much better option with a much higher ROUGE-L F1 in comparison with the original setup of offering only Top-1 option (brown reference lines marked "*BS Top-1*") on all nine tools for different fields or based on unique models. For example, when offering 10 options using our method, the ROUGE-L F1 scores of all tools increase by 16, 16, and 9 on three tasks, respectively. When offering 20 options using our method, the scores on SOP title generation even nearly double. The evident gain demonstrates the potential and generalizability of our idea to offer several dissimilar options as an effective solution to lessen users' revision need and improve tools' practicality. In fact, though not an apple-to-apple comparison, this gain is much greater than a growth of 3 to 6 contributed by the works that focus on improving the Top-1 output [3], [9], [21].

We next compare our method to the straightforward baseline *BS Top-k* (blue lines). It is not surprising that our method beats this baseline. But we found this simple method also surpasses the original *BS Top-1* dramatically. For example, with top 10 results offered, the ROUGE-L F1 of all tools go up by 12, 12, and 6 on the SOP title, BR title, and PR description generation, respectively. This confirms offering multiple options can easily improve the practicality of completion tools. Meanwhile, our method (red lines) beats this method by large margins: offering 20 options using our method contributes an improvement at the level when offering around 40 top results with *BS Top-k* on the SOP and BR title generation and even 100 top results with *BS Top-k* on the PR description generation (violet reference lines). This emphasizes the need to carefully select the more helpful outputs, so as to obtain a higher gain with fewer options.

We also compare our method to the three baselines inspired by Zhang et al. [12] that consciously select diverse outputs. As shown in Fig. 1, *Sampling + MMR* (orange lines) from Zhang et al. works slightly better than our method (red lines) on the BART-based PR description generation when k is small, but our method surpasses it a lot on all other k values and models (incl. all T5-based tools [20]). And our method also beats the other two baselines, i.e., *Sampling* (pink lines) and our adapted *BS + MMR* (green lines). Such results demonstrate our method is generally more effective than these baselines in selecting the proper outputs for software document completion tool to offer. They also tell a better selection of helpful beam search outputs, which is also easier to deploy as we discuss in Sect. II, can be more useful than an update on decoding method in our task.

VI. Conclusion, Limitations, and Future Work

Recently, various software document completion tools have been proposed to help people efficiently draft good documents. But current such tools only offer their user the Top-1 result and this only option usually needs much revision from the user to use, which hurts the practicality of such tools. In this work, we propose to let such tools offer some dissimilar high-confidence model outputs as a preliminary solution and build a clustering-based initial method to realize it. Evaluation on nine tools for three popular software document fields shows a great potential of our idea to reduce the revision need for users and improve the practicality of various software document completion tools.

This preliminary work still has some limitations. We plan to enrich it and improve our method in these aspects in the future:

- *Analysis of Parameter Settings:* There are several parameters in our method, e.g., the beam size and the clustering thresholds. In this work, we adopt the setup suggested in other studies or set them based on our initial observation on the validation set. We will tune them to better know and improve our method.
- *Evaluation on More Tools:* The preliminary evaluation only assesses our method on 9 tools for 3 tasks. In future, we plan to evaluate our method on more tasks and tools to discuss how the task features (e.g., length of output) and tool features (e.g., whether using the retrieval technique [22] or not) would affect

our method. We are also interested to explore how our method and idea can effectively serve for a wider range of tools.

• *Larger-Scale Manual Inspection:* In this work, we only perform a small-scale manual inspection. We will perform a more extensive case study later to better understand the strengths and weaknesses of our method and improve it. Besides, we plan to collect real-life feedback from the developers to understand the helpfulness of our method in practical developing activities.

ACKNOWLEDGMENT

This work was partially supported by the National Natural Science Foundation of China under the grant numbers 62250610224, 61972289, and 61832009.

REFERENCES

[1] Z. Liu, X. Xia, C. Treude, D. Lo, and S. Li, "Automatic generation of pull request descriptions," in *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*. IEEE, 2019, pp. 176–188.

[2] Z. Gao, X. Xia, J. Grundy, D. Lo, and Y. Li, "Generating question titles for stack overflow from mined code snippets," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 4, pp. 26:1–26:37, 2020.

[3] K. Liu, G. Yang, X. Chen, and C. Yu, "Sotitle: A transformer-based post title generation approach for stack overflow," in *IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022, Honolulu, HI, USA, March 15-18, 2022*. IEEE, 2022, pp. 577–588.

[4] 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 *35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21-25, 2020*. IEEE, 2020, pp. 385–397.

[5] T. Zhang, I. C. Irsan, F. Thung, D. Han, D. Lo, and L. Jiang, "Automatic pull request title generation," pp. 71–81, 2022.

[6] S. Jiang, A. Armaly, and C. McMillan, "Automatically generating commit messages from diffs using neural machine translation," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017*, G. Rosu, M. D. Penta, and T. N. Nguyen, Eds. IEEE Computer Society, 2017, pp. 135–146.

[7] C. Gao, J. Zeng, X. Xia, D. Lo, M. R. Lyu, and I. King, "Automating app review response generation," in *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*. IEEE, 2019, pp. 163–175.

[8] C. Gao, W. Zhou, X. Xia, D. Lo, Q. Xie, and M. R. Lyu, "Automating app review response generation based on contextual knowledge," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 1, pp. 11:1–11:36, 2022.

[9] T. Zhang, I. C. Irsan, F. Thung, D. Han, D. Lo, and L. Jiang, "itiger: an automatic issue title generation tool," pp. 1637–1641, 2022.

[10] Z. Gao, X. Xia, D. Lo, J. C. Grundy, and Y. Li, "Code2que: a tool for improving question titles from mined code snippets in stack overflow," in *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, D. Spinellis, G. Gousios, M. Chechik, and M. D. Penta, Eds. ACM, 2021, pp. 1525–1529.

[11] I. C. Irsan, T. Zhang, F. Thung, D. Lo, and L. Jiang, "Autoprtitle: A tool for automatic pull request title generation," pp. 454–458, 2022.

[12] F. Zhang, J. Liu, Y. Wan, X. Yu, X. Liu, and J. W. Keung, "Diverse title generation for stack overflow posts with multiple sampling enhanced transformer," *CoRR*, vol. abs/2208.11523, 2022.

[13] A. Koyuncu, K. Liu, T. F. Bissyandé, D. Kim, M. Monperrus, J. Klein, and Y. L. Traon, "ifixr: bug report driven program repair," in *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*, M. Dumas, D. Pfahl, S. Apel, and A. Russo, Eds. ACM, 2019, pp. 314–325.

[14] X. He, L. Xu, X. Zhang, R. Hao, Y. Feng, and B. Xu, "Pyart: Python API recommendation in real-time," in *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 2021, pp. 1634–1645.

[15] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[16] 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*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 7871–7880.

[17] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 140:1–140:67, 2020.

[18] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *2014 Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014, pp. 3104–3112.

[19] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Association for Computational Linguistics, 2004, pp. 74–81.

[20] "The complete evaluation results and replication package for this paper." [Online]. Available: https://github.com/zhipeng-cai/ICPC23ERA-MultiOpts

[21] H. Lin, X. Chen, X. Chen, Z. Cui, Y. Miao, S. Zhou, J. Wang, and Z. Su, "Titlegen-fl: Quality prediction-based filter for automated issue title generation," *J. Syst. Softw.*, vol. 195, p. 111513, 2023.

[22] B. Wei, Y. Li, G. Li, X. Xia, and Z. Jin, "Retrieve and refine: Exemplar-based neural comment generation," in *35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21-25, 2020*. IEEE, 2020, pp. 349–360.