TECHSUMBOT: A Stack Overflow Answer Summarization Tool for Technical Query

Chengran Yang, Bowen Xu*, Jiakun Liu, David Lo
School of Computing and Information Systems, Singapore Management University
{cryang, bowenxu, jkliu, davidlo}@smu.edu.sg

***Abstract*—Stack Overflow is a popular platform for developers to seek solutions to programming-related problems. However, prior studies identified that developers may suffer from the redundant, useless, and incomplete information retrieved by the Stack Overflow search engine. To help developers better utilize the Stack Overflow knowledge, researchers proposed tools to summarize answers to a Stack Overflow question. However, existing tools use hand-craft features to assess the usefulness of each answer sentence and fail to remove semantically redundant information in the result. Besides, existing tools only focus on a certain programming language and cannot retrieve up-to-date new posted knowledge from Stack Overflow. In this paper, we propose TECHSUMBOT, an automatic answer summary generation tool for a technical problem. Given a question, TECHSUMBOT first retrieves answers using the Stack Overflow search engine, then TECHSUMBOT 1) ranks each answers sentence based on the sentence's usefulness, 2) estimates the centrality of each sentence to all candidates, and 3) removes the semantic redundant information. Finally, TECHSUMBOT returns the top 5 ranked answer sentences as the answer summary. We implement TECHSUMBOT in the form of a search engine website. To evaluate TECHSUMBOT in both automatic and manual manners, we construct the first Stack Overflow multi-answer summarization benchmark and design a manual evaluation study to assess the effectiveness of TECHSUMBOT and state-of-the-art baselines from the NLP and SE domain. Both results indicate that the summaries generated by TECHSUMBOT are more diverse, useful, and similar to the ground truth summaries.**

Tool Link: www.techsumbot.com
Video Link: https://youtube.com/watch?v=ozuJOp_vILM
Replication Package: https://github.com/TechSumBot/TechSumBot

***Index Terms*—Summarization, Question Retrieval**

I. INTRODUCTION

Nowadays, Stack Overflow hosts a valuable resource for developers to solve technical problems. As of Nov 2022, Stack Overflow has around 23 million questions and 34 million corresponding answers, making Stack Overflow a huge base for developers to find programming-related knowledge. While developers rely on Stack Overflow to search for solutions to their technical problems, prior studies find that developers suffer from ineffective and time-consuming answer-searching processes as the Stack Overflow search engines may provide redundant, useless, and incomplete information [1]. Through a survey of developers, developers expect an answer summarization tool to provide an answer summary for their technical problems [1].

To help developers better capture the knowledge to answer technical questions, prior studies proposed several tools to retrieve information from Stack Overflow [1], [2]. More specifically, Xu et al. consider the task as a query-focused extractive answer summarization process and propose an approach to perform Stack Overflow answer summarization for a technical problem [1]. Following their work, Cai et al. implement an answer summarization tool AnswerBot that can generate an answer summary from multiple relevant Stack Overflow answers for a given technical problem [2]. However, existing approaches [1], [2] use hand-craft features to model the importance of answer sentences to the query. The handcrafted features require much human effort to create and update and are not yet enough to capture the semantics. Moreover, the generated summaries from existing approaches still can contain apparent information redundancy. They are weak at distinguishing and eliminating semantically similar yet syntactically different sentences. Apart from the deficiencies of the summarization algorithm, we observe that AnswerBot lacks timeliness, extensibility, and customizability. Considering that Stack Overflow produces 4.55 answers per minute[1], it is hard to ensure the information timeliness of the local dataset. Besides, AnswerBot only focused on Java-related problems, and it might not perform well in other programming-related problems [2]. Moreover, AnswerBot can not meet the customization needs of developers as it does not allow users to modify the parameters.

To address the aforementioned limitations of the previous tools, we propose our tool TECHSUMBOT to generate answer summaries for a technical problem, in which the summarization algorithm is described in detail in our prior work [3]. We also implement a web-based search engine that facilitates using TECHSUMBOT. We present both out-of-the-box summarization and advanced summarization modes for using TECHSUMBOT. The out-of-the-box summarization mode allows users to input their technical problems in the search box and offer an answer summary extracted from multiple most relevant answers in Stack Overflow just in a few seconds. More specifically, TECHSUMBOT adopts the Stack Overflow built-in search engine to identify relevant answers while a query is given, which ensures the summarization source is in sync with Stack Overflow. Then TECHSUMBOT leverages a three-stage summarization framework to produce an answer

∗ Corresponding author

[1]https://api.stackexchange.com/docs/info#&site=stackoverflow&run=true

summary. In each summarization module, TECHSUMBOT performs summarization while taking into consideration 1) the answer sentence's usefulness to the query, 2) the centrality of each sentence to all candidates, and 3) semantic information redundancy, in turn. In the advanced summarization mode, TECHSUMBOT allows users to define the summary length and summarization scope when producing the answer summary.

We perform both automatic and manual evaluation to explore the effectiveness of TECHSUMBOT. TECHSUMBOT outperforms state-of-the-art baselines significantly in both automatic and manual evaluation manner. The manual evaluation further demonstrates that TECHSUMBOT can produce diverse and useful answer summaries for a technical problem.

II. APPROACH

We demonstrate the overall workflow of TECHSUMBOT in Figure 1. TECHSUMBOT takes a technical query as input and outputs an answer summary from multiple relevant Stack Overflow answers. It contains three core modules: Usefulness Ranking, Centrality Estimation, and Redundancy Removal. Firstly, given a technical query, TECHSUMBOT adopts the Stack Overflow built-in searching engine to search for its most relevant answers and decompose them into a list of sentences. Then Usefulness Ranking module outputs a ranked list of sentences, which is ordered by the usefulness of each answer sentence to a specific query. Next, the second Centrality Estimation module re-rank the sentence list by measuring the centrality of each sentence among candidates by TextRank [4]. The third Redundancy Removal module removes semantically redundant sentences in the ranked list through a greedy algorithm-based selection mechanism, where an in-domain sentence representation model is applied. In the end, the top-5 answer sentences in the final list are used to form an answer summary to the target technical query.

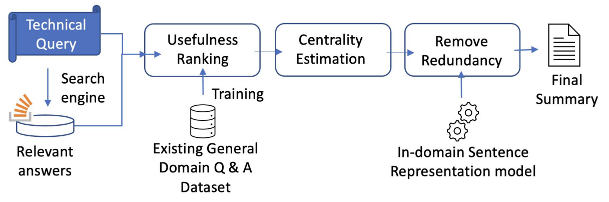


Fig. 1. Overview of TECHSUMBOT

A. Module I: Usefulness Ranking

In this Usefulness Ranking module, we model the usefulness of each answer sentence with respect to a technical problem. We train a BERT-based classifier to predict the usefulness of each answer sentence to the query. BERT is a powerful transformer-based language model [5].

We apply a large-scale general-domain QA dataset ASNQ [6] to fine-tune the BERT-based classifier. Each data in ASNQ consists of a query and corresponding answer sentence. For each of the pairs $\langle query, sentence \rangle$ in the ASNQ dataset, we input them into the BERT model. Then we extract the representation vectors from the BERT output and feed into the final classification layer. We apply the sigmoid function to produce the likelihood of the input class (i.e., positive and negative). we select cross entropy as the loss function.

For the inference stage, we pick the classifier's predicted score of positive class as the representation of sentence usefulness. Finally, Usefulness Ranking module returns a sentence list ranked with the usefulness score. A sub-list including the top k answer sentences with the highest usefulness scores serves as the input of the next module.

B. Module II: Centrality Estimation

To select summative sentences, we further consider the centrality of each sentence among all candidates. The most central sentences in a sentence cluster give all the necessary and sufficient amount of information for the cluster's main content. We apply extractive summarization approach TextRank [4] to extract each sentence's centrality score. By constructing a sentence graph, where each node is an answer sentence, and the weight of each edge is the word similarity between two nodes, TextRank quantifies the centrality of each sentence based on the graph's information. Finally, we re-rank the input sentence list with the centrality score, which is obtained from TextRank. We feed the re-ranked list into the final module.

C. Module III: Redundancy Removal

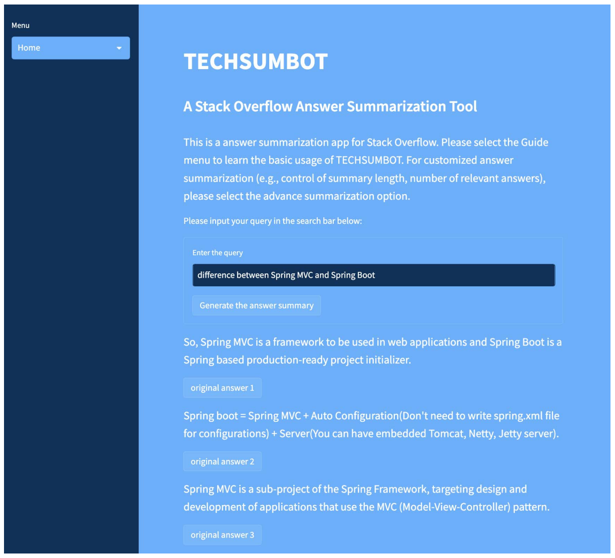
Given a re-ranked sentence list, Redundancy Removal module eliminates redundant information by applying the greedy selection to remove redundant sentences.

*1) **Greedy Algorithm for Redundancy Removal:*** Given the ranked sentence list from the Centrality Estimation module, we pick each sentence into the final summary one by one. Within each selection, when the cosine similarity between the current sentence's representation and which has been included in the final summary is above a threshold T, we regard the current sentence as redundant to the final summary and discard it. At the end of the greedy selection, we pick the top five sentences in the sentence list and form them as the final summary.

*2) **Contrastive Learning for In-domain Sentence Embedding:*** Note that the essential step of the greedy selection is to transform each sentence into a vector representation through a sentence representation model, which enables the calculation of the cosine similarity between two sentences. Here we leverage the SE domain language knowledge inferred from a large-scale SO sentence relationship dataset and propose an in-domain sentence representation model via contrastive learning.

Contrastive learning-based sentence representation aims to learn a sentence embedding space, where similar sentence pairs stay close to each other while dissimilar ones are far apart in this space [7]. Following SIMCSE [7], the state-of-the-art sentence representation approach based on contrastive learning, we use a transformer-based pre-trained model RoBERTa [8] as the base model and add a multilayer perceptron layer on the top of it. At the training stage, for each input sentence triplet, the loss function is defined as:

$$\mathcal{L} = -log \frac{e^{sim(r_i, r_i^+)/\tau}}{\sum_{j=1}^{N} (e^{sim(r_i, r_j^+)/\tau} + e^{sim(r_i, r_j^-)/\tau})} \quad (1)$$



TECHSUMBOT

A Stack Overflow Answer Summarization Tool

This is a answer summarization app for Stack Overflow. Please select the Guide menu to learn the basic usage of TECHSUMBOT. For customized answer summarization (e.g., control of summary length, number of relevant answers), please select the advance summarization option.

Please input your query in the search bar below:

Enter the query

difference between Spring MVC and Spring Boot

Generate the answer summary

So, Spring MVC is a framework to be used in web applications and Spring Boot is a Spring based production-ready project initializer.

original answer 1

Spring boot = Spring MVC + Auto Configuration(Don't need to write spring.xml file for configurations) + Server(You can have embedded Tomcat, Netty, Jetty server).

original answer 2

Spring MVC is a sub-project of the Spring Framework, targeting design and development of applications that use the MVC (Model-View-Controller) pattern.

original answer 3

Fig. 2. Homepage of TechSumBot Website

while r_i denotes the representation of each sentence. N is the length of a mini-batch, τ is a temperature hyperparameter, and $sim(r_1, r_2)$ denotes the cosine similarity. We train RoBERTa model by leveraging an in-domain sentence representation dataset that is automatically crafted by our prior work [3].

III. Implementation Details

We implement TechSumBot in the form of a search engine website. Figure 2 shows the homepage of TechSumBot. Developers input the technical queries in the search box of TechSumBot, then TechSumBot would directly return the answer summary as well as the links to the corresponding Stack Overflow answers. Meanwhile, considering the importance of hyperlinks, TechSumBot allows users to trace the hyperlinks in answers [9], [10]. For each hyperlink embedded in the answer sentence, TechSumBot set a flag (i.e., '[hyperlink]') to remind users to trace the external resources.

TechSumBot provides a guide page to introduce how to use the two summarization modes as well as several usage examples. We provide two modes for users: out-of-the-box summarization, which allows users to experience our tool without additional configuration. We set the length of the answer summary as five sentences by default and treat the top-10 relevant Stack Overflow answers returned by the Stack Overflow search engine as the summarization source. The other mode is advanced summarization to meet developers' customized summarization needs. Compared with the out-of-the-box summarization mode, TechSumBot offers configurable parameters to enable users to modify the length of generated summaries and the number of relevant sentences. To narrow down the answer search scope and provide more useful summaries, TechSumBot also allows users to specify the relevant programming language of their technical questions.

TechSumBot is based on the Browser/Server architecture. For each user query, we firstly use the official Stack Exchange API[2] (i.e., Stack Overflow built-in search engine) to search for the most relevant Stack Overflow answers for that query. We register the stack exchange API key for more API request quotas. The Stack Overflow answers as well as the customized parameters are then fed into the backend service, where we apply our summarization algorithm to generate an answer summary. After that, TechSumBot visualize the answer summary in the web browser. We use the streamlit[3] framework to implement TechSumBot web interface. Streamlit is an open-source web-based app framework.

IV. Evaluation

A. Evaluation Setup

Dataset We adopt the Stack Overflow answer summarization dataset TechSumBench proposed by our prior work [3] for the automatic evaluation. This dataset consists of 111 manually created summaries for 37 technical questions.

Baselines we compare TechSumBot against two extractive summarization baselines: 1) the state-of-the-art Stack Overflow answer summarization approach, AnswerBot [1]; and 2) the state-of-the-art query-focused multi-doc summarization approaches in the NLP domain, QuerySum [11].

Automatic Evaluation Metric Following the existing extractive summarization approaches [11], we use ROUGE-N for automatic evaluation. ROUGE-N is a standard automatic evaluation metric for summarization systems. It evaluates the n-gram overlapping between a generated summary and the ground-truth summaries. ROUGE-N is defined as:

$$\text{ROUGE-N} = \frac{\sum\limits_{S_i \in S} \sum\limits_{gram_n \in Si} Count_{match}(gram_n)}{\sum\limits_{S_i \in S} \sum\limits_{gram_n \in Si} Count(gram_n)} \quad (2)$$

where S denotes the ground-truth summary, n denotes the number of grams in a summary, $Count_{match}(gram_n)$ and $gram_n$ denote the number of grams that are coexistence in the ground-truth summaries and generated summaries. We apply ROUGE-N to evaluate the quality of generated summaries against the ground-truth summaries. The higher value of ROUGE-N, the better performance of an approach is.

Manual Evaluation Setting By following the prior work [1], we conduct a user study to evaluate all approaches regarding the usefulness and diversity of their generated summaries. Five participants are involved in this user study. One is a postdoctoral fellow, and the others are PhD students. All the participants have at least three years of developing experience in Java or Python. We randomly sample ten technical queries from TechSumBench as the experiment data. We collect the answer summaries for each query generated from TechSumBot, AnswerBot, and QuerySum. We ask the participants to give 5-point Likert scores to all summaries in terms of the diversity and usefulness of the query. "Extremely useless" and "extremely redundant" denotes as 1. "Extremely useful" and "extremely diverse" denotes as 5.

[2]https://api.stackexchange.com/docs/search
[3]https://streamlit.io/

TABLE I
PERFORMANCE OF TECHSUMBOT AND BASELINES

System	ROUGE-1 (%)	ROUGE-2 (%)	ROUGE-L (%)
AnswerBot	0.490 (14.90)	0.276 (36.59)	0.456 (17.54)
QuerySum	0.508 (10.83)	0.284 (32.75)	0.476 (12.61)
TECHSUMBOT	**0.563**	**0.377**	**0.536**

B. Evaluation Result

1) Automatic Evaluation Result: Table I is the automatic evaluation result for TECHSUMBOT and baselines. TECHSUMBOT consistently outperforms AnswerBot and QuerySum in terms of all evaluation metrics. Compared with AnswerBot, TECHSUMBOT achieves better performance in terms of ROUGE-1, ROUGE-2, and ROUGE-L by 14.90%, 36.59%, and 17.54%, respectively. Compared with QuerySum, TECHSUMBOT outperforms it by 10.83%, 32.75%, and 12.61%, in terms of ROUGE-1, ROUGE-2, and ROUGE-L, respectively.

TABLE II
USER STUDY RESULT

System	Usefulness	Diversity
AnswerBot	3.68	3.62
QuerySum	3.80	3.64
TECHSUMBOT	**4.02**	**4.26**

2) Manual Evaluation Result: User study result is shown in Table II. We observe that TECHSUMBOT achieves the best performance consistently and the improvement is significant (p¡0.05) in terms of usefulness and diversity. Participants consider TECHSUMBOT to be useful (4.02 out of 5) and diverse (4.26 out of 5). TECHSUMBOT outperforms AnswerBot and QuerySum by 9.23% and 5.79% in terms of average usefulness score and 17.68% and 17.03% in terms of diversity score.

V. RELATED WORK

Many text summarization approaches have been applied to support developers in better programming and learning. Uddin et al. [12] proposed Opiner to support developers in making informative API selection and usage decisions by providing opinion-based API summaries. Besides, Nadi and Treude [13] propose an approach to identify the essential sentences in Stack Overflow. The essential sentences are helpful to navigate the developers reading Stack Overflow answers efficiently. Also, Naghshzan et al. [14] proposed a tool to summarize the discussion about method-level Android API in Stack Overflow. They apply TextRank to extractive the most informative Stack Overflow answer sentences to the Android API.

On the other hand, there are many query-focused and multi-answer summarization extractive summarization approaches in the NLP community. Traditional summarization approaches, e.g., LexRank [15], apply the PageRank algorithm to extract the most representative sentences from a sentence graph. Xu et al. [11] proposed a coarse-to-fine framework to filter out the irrelevant information during the summarization.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present TECHSUMBOT, an answer summarization generation tool, which generates answer summaries from Stack Overflow for technical queries. TECHSUMBOT enables users to obtain the answer summaries out-of-the-box. Users can further modify the summarization conditions via advanced summarization mode. TECHSUMBOT outperforms state-of-the-art summarization baselines from NLP and SE domains in both automatic evaluation and manual evaluation. Furthermore, participants of the manual evaluation agree that TECHSUMBOT generates useful and diverse answer summaries.

ACKNOWLEDGMENT

This research / project is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award No.: MOE2019-T2-1-193). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

REFERENCES

[1] B. Xu, Z. Xing, X. Xia, and D. Lo, "Answerbot: Automated generation of answer summary to developers' technical questions," in *ASE 2017*. IEEE, 2017, pp. 706–716.

[2] L. Cai, H. Wang, B. Xu, Q. Huang, X. Xia, D. Lo, and Z. Xing, "Answerbot: an answer summary generation tool based on stack overflow," in *FSE 2019*, 2019, pp. 1134–1138.

[3] Y. Chengran, B. Xu, F. Thung, Y. Shi, T. Zhang, Z. Yang, X. Zhou, J. Shi, J. He, D. Han *et al.*, "Answer summarization for technical queries: Benchmark and new approach," in *2022 37nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2022.

[4] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004, pp. 404–411.

[5] D. Jacob, C. Ming-Wei, L. Kenton, and T. Kristina, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[6] S. Garg, T. Vu, and A. Moschitti, "Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7780–7788.

[7] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 6894–6910.

[8] 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.

[9] J. Liu, X. Xia, D. Lo, H. Zhang, Y. Zou, A. E. Hassan, and S. Li, "Broken external links on stack overflow," *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3242–3267, 2022.

[10] J. Liu, H. Zhang, X. Xia, D. Lo, Y. Zou, A. E. Hassan, and S. Li, "An exploratory study on the repeatedly shared external links on stack overflow," *Empirical Software Engineering*, vol. 27, no. 1, pp. 1–32, 2022.

[11] Y. Xu and M. Lapata, "Coarse-to-fine query focused multi-document summarization," in *Proceedings of the 2020 Conference on empirical methods in natural language processing (EMNLP)*, 2020, pp. 3632–3645.

[12] G. Uddin and F. Khomh, "Automatic summarization of api reviews," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 159–170.

[13] S. Nadi and C. Treude, "Essential sentences for navigating stack overflow answers," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2020, pp. 229–239.

[14] A. Naghshzan, L. Guerrouj, and O. Baysal, "Leveraging unsupervised learning to summarize apis discussed in stack overflow," in *2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2021, pp. 142–152.

[15] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *Journal of artificial intelligence research*, vol. 22, pp. 457–479, 2004.