Curating Online Forum Knowledge as Troubleshooting Dataset for Generative AI Using Fusion Retrieval

Youyang Kim[†], Yaoping Ruan[‡*], Byungchul Tak[†*]

[†]Kyungpook National University, Daegu, Republic of Korea
[‡]Zhejiang A&F University, Zhejiang, China
youyangkim@knu.ac.kr, ruanyaoping@gmail.com, bctak@knu.ac.kr

Abstract

Problem-solution datasets are commonly used for generative AI training. Especially when building a domain-specific conversational system, both fine-tuning and retrieval-augmented generation rely heavily on the availability of high-quality dataset. In system troubleshooting domain, few troubleshooting datasets are available for model tuning. We find annotating troubleshooting dataset a nontrivial task mainly due to the multi-modality nature of the data, which contain a mix of disparate data artifacts such as codes, log messages, console outputs, commands, and some descriptions in natural language. In this paper, we present a comprehensive approach to acquire the most relevant online forum data as answer for the input problem description. Our main goal is to retrieve the most relevant online forum post for a given problem description, either as a semi-supervised curation method for training dataset, or as a retrieval mechanism for augmented generation. Our key idea is to effectively separate data artifacts in the documents and assess the relevance across pairs of heterogeneous artifact types. To this end, we utilize a bag of language models, and then use the weighted accumulative score to find the most relevant answer. Compared with several baseline techniques, our method demonstrates significant improvements by at least 42.89% against the best competitor regarding search ranking quality. Also, it successfully ranks the ground-truth forum posts within the top 10 in 96.1% of the cases, significantly reducing human annotation effort.

Introduction

Online technical forums such as Stack Overflow (Stack Overflow 2023), Quora (Quora, Inc 2023), Microsoft Community (Microsoft 2023), and Apple Support Community (Apple 2023) have become a popular resource for technical staff seeking solutions to problems arising in computer systems. These forums offer a vast repository of knowledge and experiences, allowing users to publish, discuss, and retrieve questions related to their specific fields. With the growing complexity of modern software stacks, they have become indispensable for technical staff for quick and effective problem-solving.

To the best of our knowledge, however, few have focused on harvesting online forums comprehensively as troubleshooting dataset for generative AI. The troubleshooting dataset falls into the question-answer category, but problem descriptions for troubleshooting can contain richer information than a question in human language. According to the Information Technology Infrastructure Library (ITIL), which prescribes best practices for IT service and asset management, a problem description should consist of almost a dozen of key elements including problem summary, problem manifests such as error messages, system crashes or performance degradation, and environment details such as operating system, hardware, network configuration and other factors (iti 2024). Seeking the most relevant post which can server as the answer to the given problem description is at the core. In addition to traditional keyword-based search engines, semantic search using language models based on word embedding or sentence embedding has gained large attention in recent years (Antonio Luca, Cimino, and Vaglini 2021; Esteva et al. 2021; Peinelt, Nguyen, and Liakata 2020; Deshmukh and Sethi 2020; Khattab and Zaharia 2020). Both the keyword-based and semantic search perform poorly on forum data mainly due to the complexity and multi-modality nature of the posts.

To facilitate such data curation process, we propose a fusion retrieval framework that delivers significantly improved matching capability of online forum troubleshooting data *via a fine-granular multi-modal data retrieval approach*. The results can be used for model training or serve as a context for retrieval augmented generation (RAG) (Siriwardhana et al. 2023; Salemi and Zamani 2024) using LLMs. Our scheme is an overarching approach of separating data artifacts in each post and computing relevance scores *across pairs of the artifacts* and aggregating these multiple search results for better accuracy. Typical data artifacts include code snippets, log messages, console output, command line with arguments, and failure description texts. To our knowledge, this is the most *comprehensive and general approach* to explore online forum data with multi-modality as a key design factor.

We evaluated our approach on several popular distributed applications (OpenStack (OpenInfra Foundation 2023), Spark (Spark 2023), Elasticsearch (B.V. 2023), Kafka (kafka 2023) and Mongodb (Inc. 2023)) which are popular topics in Stack Overflow. We reproduced 77 failure cases by injecting failures into the applications replicated in our local environment. The metrics we use are the

*Corresponding authors

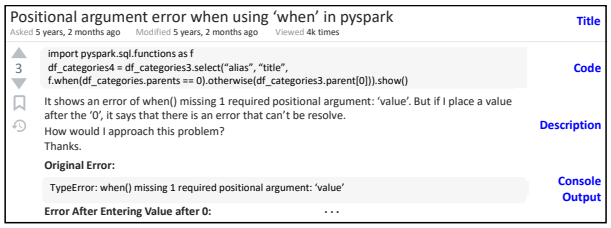

Figure 1: Stack Overflow question with various artifacts.

Mean Reciprocal Rank (MRR) and the top k SuccessRates (R@k). The results demonstrated much higher accuracy in finding the relevant troubleshooting forum posts compared to 4 other baselines including `Google` search, ensemble of 3 models, `BM25`, and `SentenceBERT`. Overall, the MRR of our technique was 42.89% better than the best competitor. Furthermore, our technique ranked the ground-truth Stack Overflow documents within the top-10 of the returned list for 96.1% of the cases, whereas the runner-up competitor obtained 72.7%.

The main contributions are: First, we propose a troubleshooting dataset curation approach based on the individual and aggregated relevance scores *across heterogeneous artifacts pairs*. To enable this, we use a bag of language models and tune them using domain-specific documents, which yield significantly higher accuracy than existing models publicly available in finding correct troubleshooting information. Second, we release the troubleshooting dataset and data processing models to promote research in this field. It consists of all the code snippets, commands, logs, and console outputs from the reproduced failure cases which can be used for troubleshooting model training.

Data Retrieval and Curation Pipeline

Forum Data Artifacts

Figure 1 showcases the data artifacts found in forum posts. Besides the title (**`tnd`**) and description (**`des`**) of the problem, each post may contain various data such as code snippets (**`cod`**), commands (**`cmd`**), log messages (**`log`**), and console output (**`cns`**).

One of the challenges in conventional approaches of finding the most relevant post is to utilize all the artifacts. Based on this observation, we divide the task into two steps: *i)* calculating the relevance score of each artifact pair and *ii)* aggregating the calculated scores for that pair. Figure 2 shows the overall workflow and pipelines.

The data artifact type falls into three categories: (1) programming language type such as code snippet (`cod`), (2) (semi) natural language types such as log (`log`), console output (`cns`), and description (`des` and `tnd`), and (3) short phrase or token type such as command (`cmd`). For a dataset consists of all artifacts of {`cod`, `cmd`, `log`, `cns`, `des`, `tnd`}, theoretically an arbitrary pair of two data artifacts (u_i, $v_{j,k}$) can be compared for relevance.

In practice, within the stackoverflow posts we used for this study, we found that only 4.91% contain both `cmd` and `cns` and 2.88% contain both `cmd` and `log`. Due to the insufficiency of training data, we exclude `cmd:log`, `log:cmd`, `cmd:cns`, and `cns:cmd`.

In IT operation, code and command, log and console output are always interchangeably referred, so that we also exclude the `cod` and `cmd` pair and `log` and `cns` pair.

We adopt five popular models based on the nature of each artifact: `CodeBERT` (Feng et al. 2020), `GraphCodeBERT` (Guo et al. 2021) and `CodeT5+` (Wang et al. 2023) for programming language type since all these models are trained with code, `SentenceBERT` (Reimers and Gurevych 2019) for natural language types, and `BM25` for short phrases, given that they are essentially keyword tokens. The full list of models per pair and the choice of modeling techniques are in Figure 3.

Assessing Relevance Between Different Artifacts

Relevance score of code and natural language pairs `CodeBERT`, `GraphCodeBERT` and `CodeT5+` are all pretrained transformer-based models for code and natural language pairs. We created a subset of artifact pairs and select the best performed model for each pair. As a result, `CodeBERT` is used for `cod:tnd`, `GraphCodeBERT` for `cod:log`, `cod:cns`, `log:cod`, and `cns:cod`, and `CodeT5+` for `cod:cod` and `des:cod`. When quantifying relevance score, we directly use the prediction probability of `CodeBERT` output as the result, and the inner product of each of the input and target vector of `GraphCodeBERT` and `CodeT5+embedding` as the result. This unification is based on the fact that `CodeBERT` uses cross-encoder and the output ranks the most relevant candidate already while the other two models are bi-encoder-based which merely convert the input text and each of the target text into vectors.

Relevance score of natural language pairs We adopt `SentenceBERT` to compute the relevance between natural language pairs, namely logs, console outputs and descriptions. We first preprocess the input data to be sentence-like as much as possible to comply with the expected model input. Then, we compute the cosine similarity between a user-side data artifact (u_i) and every question part data artifact ($v_{j,k}$). At post level, the relevance score is the maximum similarity among the sentence pairs.

Relevance score of tokens or short phrase pairs `BM25` is adopted to compare two tokens or short phrases, given its popularity. The core of it is a bag-of-words retrieval function, which works well on scenarios that can be ranked using keywords. Unlike the codes, system commands (`cmd`) are quite distinctive and primarily used in fixed forms with predefined arguments. We use the return value for each specific input query directly as the relevance score.

Aggregating Relevance Using Weighted Geometric Mean

In order to aggregate relevance at the document level, we use the *weighted geometric mean* of ranks from all available artifact pairs for a given input and candidate pair. The use of weighted geometric mean reflects our intention to reward highly matched artifacts more through multiplication rather than adding the arithmetic means. Since not all posts contain

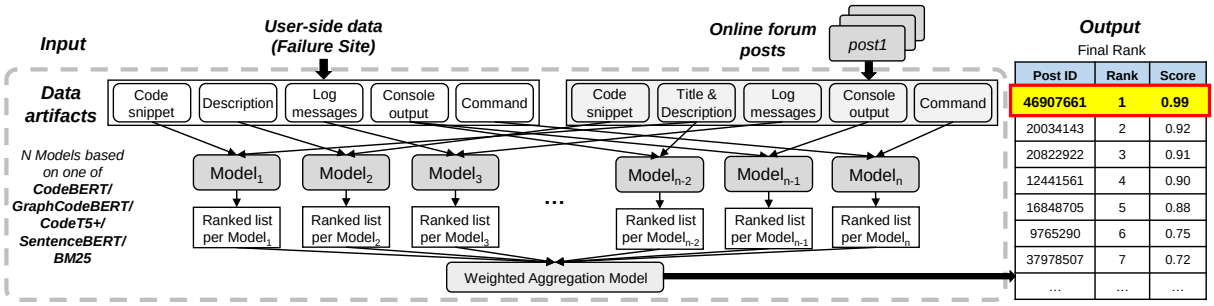

Figure 2: Overview of our technique's workflow.

Model number	Data artifact pair (user-side:forum-side)	Associated model	Model number	Data artifact pair (user-side:forum-side)	Associated model
M_1	cod:cod	CodeT5+	M_{10}	des:tnd	SentenceBERT
M_2	cod:log	GraphCodeBERT	M_{11}	des:cns	SentenceBERT
M_3	cod:tnd	CodeBERT	M_{12}	des:cmd	BM25
M_4	cod:cns	GraphCodeBERT	M_{13}	cns:cod	GraphCodeBERT
M_5	log:cod	GraphCodeBERT	M_{14}	cns:tnd	SentenceBERT
M_6	log:log	SentenceBERT	M_{15}	cns:cns	SentenceBERT
M_7	log:tnd	SentenceBERT	M_{16}	cmd:tnd	BM25
M_8	des:cod	CodeT5+	M_{17}	cmd:cmd	BM25
M_9	des:log	SentenceBERT			

Figure 3: Data artifact pairs and associated models (m_i) to compute the relevance scores.

the full set of artifacts, by applying the weight sum ratio we also give preference to those with more information by penalizing the ones with missing artifacts.

Evaluation of Retrieval Accuracy

Since the foundation of our data curation is to retrieve the most relevant forum post as the solution for the given problem description, we evaluate the effectiveness of our fusion algorithm by quantifying the ranked position of the ground truth post. Among the 23 million posts, we selected a subset based on the tag name: MongoDB (Inc. 2023), Spark (Spark 2023), OpenStack (OpenInfra Foundation 2023), Kafka (kafka 2023), and Elasticsearch (B.V. 2023). In choosing target applications, we opt to software packages rather than programming languages or specific technique given the limited availability of the application-specific training datasets. These applications are rather popular, with the most number of posts in the forum and covering modern technologies such as NoSQL database, big data platform, cloud management, distributed event processor and search engine. Despite of their popularity, the scarcity of corresponding datasets in these domains presumably is because of the complexity and difficulty in data curation.

In order to collect the artifacts as complete as possible, as well as to verify the correctness of the solution, we set up the applications in our local environment and inject errors based on the description of the post. To this end, we selected a total of 77 problem scenarios, reproduced the problem and verified the solutions manually. We admit that while this approach may not be optimal for large scale production, it covers the richest range of artifacts and provides the least ambiguity to demonstrate the validity of the approach.

We adopted Mean Reciprocal Rank (MRR) and the SuccessRate@k (Gu, Zhang, and Kim 2018; Jiang et al. 2020) as the metrics for our evaluation. MRR is concerned with the rank of a single best item in the list so that it indicates the quality of each retrieval approach, and the SuccessRate@k (also denoted as $R@k$) metric measures the average proportion of ground truth questions ranked within the top k. MRR is defined as:

$$MRR = \frac{1}{|C|}\sum_{c=1}^{|C|}\frac{1}{Rank_c} \quad (1)$$

where C is the number of reproduced cases. The maximum value of MRR is 1, and the higher the better.

Results Analysis

We compare our approach with `BM25`, `SentenceBERT`, `Google` search, and `Ensemble` which combines `BM25`, `SentenceBERT` and `CodeBERT`. For all experiments, we use the same problem description as the input, and identify the position of the ground truth returned by each method. The rank position is based on the dataset we filtered from Stack Overflow using tags. Specifically, there are 3451 posts for OpenStack, 172209 for MongoDB, 100640 for Spark, 57523 for Elasticsearch and 30896 for Kafka.

All five applications' results show that our technique produces significantly better ranking results than all competitors. Furthermore, it can rank the ground truth documents within the top 10 results for 96% of all failure cases. Figure 4 shows the overall performance in ranking the ground truth based on the MRR and SuccessRate@k metrics. Based on MRR, our technique obtained a score of 0.713, the highest value, beating all other alternative approaches. While the absolute value of MRR is a little hard to interpret, SuccessRate@k results are quite straightforward: We ranked 57.1% of the cases as the first choice suggesting that the matching results can be used as the answer for training dataset directly. If we annotate based on the top 5 results, then 92.2% of the cases fall in that category. If our base is the top 10 results, then it can retrieve 96.1% of the ground truth successfully. It suggests that our approach can effectively match the ground truth solutions to generate a troubleshooting dataset.

Within all the approaches, `BM25` performs well compared to other baseline techniques, considering that it is, in principle, based on the classic term and document frequency arithmetically similar to TF-IDF. This is particularly true for applications such as OpenStack where commands are the main artifacts. For other applications, the most prevalent artifact is code snippets. Among the cases, 64% of MongoDB, all of Spark, Elasticsearch, and Kafka contain codes.

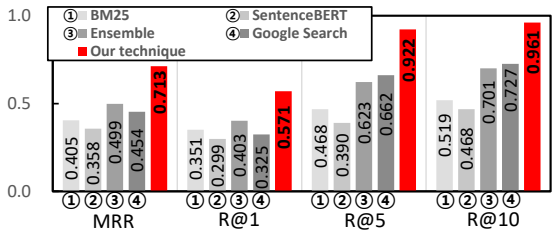

Figure 4: Ranking quality comparison.

We investigate individual model's contribution to accuracy to understand which data artifact pairs are the most critical. Based on MRR, we find that `cns:cns` model produces the highest MRR of 0.547, followed by `cmd:cmd`. This shows that it is important to utilize the console output `(cns)` in the search for the most relevant online forum documents whenever available. However, when we looked at the ranking output produced by only using homogeneous data pair models such as the `cns:cns` or `cmd:cmd`, we found that the rank variance was high. This observation supports our technique of using multiple data types and comparing heterogeneous data type models together complementarily to achieve high search performance. Furthermore, even if a data artifact pair may show low MRR when used alone, they are indispensable to our technique since their synergy is needed to produce high overall search accuracy. Although not presented here in detail, we also conducted an ablation analysis by removing certain pairs from our technique to observe their impacts. The models that affected the accuracy the most came out to be `cns:cns`, followed by `des:tnd`, `log:log`, `des:cod` and `cod:tnd`. Their removal resulted in the MRR being dropped to 0.376, 0.618, 0.647, 0.677 and 0.678, respectively.

Tables 1, 2, 3, 4, and 5 in Appendix show ranking results on all baselines and ours of each failure case.

Related Work

The availability of troubleshooting dataset is quite limited. There are a handful of datasets in HuggingFace that consist of posts from Stack Overflow. Some of them do contain questions along with their selected answer or question and answers grouped by specific tags. However, this simple question and answer dataset are often including low-quality or off-topic answers. LinkSo (Liu et al. 2018) proposes a dataset of similar questions in Stack Overflow. Their work identifies similar questions by manual annotation by community users. Moreover, their study shows that the learning-based approach outperforms retrieving linked question pairs rather than non-learning-based approaches.

Significant attention has been paid to data management of online forum knowledge from information retrieval perspective. Researchers have focused on different aspects of online forum knowledge, such as recommending relevant questions, retrieving APIs or code snippets, clustering and mining topics, and summarizing answers in online forums. Ponzanelli et al. (Ponzanelli et al. 2014) proposed `PROMPTER` that retrieved and recommended useful `Stack Overflow` discussions by computing various relevance scores of code and questions. Rahman et al. (Rahman, Yeasmin, and Roy 2014) and Li et al. (Li et al. 2015) explored IDE-based search solution with working context to capture the relevance between programming problems and online posts. `AnswerBot` summarized relevant question posts for developers to quickly catch the key points (Xu et al. 2017), but it used only the title of the posts in retrieving the relevant posts. Liu et al. (Liu et al. 2021) observed a gap between the knowledge users are interested in and the knowledge they can retrieve using search engines. Developers also advocated a better data organization of Q&A forums so that massive code snippets available on Stack Overflow could be easily reused (Wu et al. 2019). Hoogeveen et al. (Hoogeveen et al. 2018) proposed various natural language processing and ML approaches to web forum retrieval. Furthermore, most of the existing literature has focused on understanding user intents in the forums (Cao et al. 2021), the structural characteristics of the forum (Meldrum, Licorish, and Savarimuthu 2017; Kim, Wang, and Baldwin 2010), and coding aspects (Yang, Hussain, and Lopes 2016; Baltes et al. 2018).

Different from the above studies that focused on utilizing the code artifact for searching, we consider multiple types of data artifacts on the user-side and online forum side to retrieve relevant questions specifically to the troubleshooting problem. Gao et al. (Gao et al. 2023) proposed a query-driven code recommendation tool for developers to find the most relevant code examples. Their work is complementary to our work in the sense that it assists the developers in figuring out the best code snippets to satisfy their needs.

Conclusion & Future Work

In this work, we presented a novel data curation approach by retrieving the most relevant document from the online forum posts. The key principle of our technique was to recognize data artifact types separately in the online forum documents and to apply the most suitable NLP models on multiple pairs of them to differentiate accumulative relevance across disparate types. Performing beyond simple string search or keyword search, our technique was able to leverage more relevant information between disparate data types in the posts, and it was able to improve the accuracy of matching significantly. Based on the 77 actual failure cases from 5 applications, our results showed 96.1% matching accuracy and an improvement of 42.89% over the best competitor.

While our fault injection experiments provide the most confident verification against the ground truth, it is unrealistic to reproduce all cases in the dataset. However, we can perform the same artifact-level evaluation method by reconstructing problem descriptions for any case in the forum to form a more complete dataset. We also believe our general idea is applicable to curate multi-modal dataset. As another future work, we are exploring topics other than IT troubleshooting forums.

`Data & code availability:` Data curation scripts, models, and dataset will be available online.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2021-NR060080).

References

2024. Information Technology Infrastructure Library. https://www.ibm.com/topics/it-infrastructure-library.

Antonio Luca, A.; Cimino, M. G. C.; and Vaglini, G. 2021. Technological troubleshooting based on sentence embedding with deep transformers. *Journal of Intelligent Manufacturing*, 32.

Apple. 2023. Apple Support Community. https://discussions.apple.com/.

Baltes, S.; Dumani, L.; Treude, C.; and Diehl, S. 2018. Sotorrent: Reconstructing and analyzing the evolution of stack overflow posts. In *Proceedings of the 15th international conference on mining software repositories*, 319–330.

B.V., E. 2023. Elasticsearch. https://www.elastic.co/kr/elasticsearch/. Accessed: 2023-07-20.

Cao, K.; Chen, C.; Baltes, S.; Treude, C.; and Chen, X. 2021. Automated query reformulation for efficient search based on query logs from stack overflow. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 1273–1285. IEEE.

Deshmukh, A. A.; and Sethi, U. 2020. IR-BERT: Leveraging BERT for Semantic Search in Background Linking for News Articles. arXiv:2007.12603.

Esteva, A.; Kale, A.; Paulus, R.; Hashimoto, K.; Yin, W.; Radev, D.; and Socher, R. 2021. COVID-19 Information Retrieval with Deep-Learning based Semantic Search, Question Answering, and Abstractive Summarization. *NPJ Digital Medicine*, 4(1).

Feng, Z.; Guo, D.; Tang, D.; Duan, N.; Feng, X.; Gong, M.; Shou, L.; Qin, B.; Liu, T.; Jiang, D.; and Zhou, M. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1536–1547. Online: Association for Computational Linguistics.

Gao, Z.; Xia, X.; Lo, D.; Grundy, J.; Zhang, X.; and Xing, Z. 2023. I Know What You are Searching For: Code Snippet Recommendation from Stack Overflow Posts. *ACM Transactions on Software Engineering and Methodology*, 32(3): 1–42.

Gu, X.; Zhang, H.; and Kim, S. 2018. Deep code search. In *Proceedings of the 40th International Conference on Software Engineering*, 933–944.

Guo, D.; Ren, S.; Lu, S.; Feng, Z.; Tang, D.; Liu, S.; Zhou, L.; Duan, N.; Svyatkovskiy, A.; Fu, S.; Tufano, M.; Deng, S. K.; Clement, C. B.; Drain, D.; Sundaresan, N.; Yin, J.; Jiang, D.; and Zhou, M. 2021. GraphCodeBERT: Pre-training Code Representations with Data Flow. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Hoogeveen, D.; Wang, L.; Baldwin, T.; Verspoor, K. M.; et al. 2018. Web forum retrieval and text analytics: A survey. *Foundations and Trends® in Information Retrieval*, 12(1): 1–163.

Inc., M. 2023. MongoDB. https://www.mongodb.com/. Accessed: 2023-07-20.

Jiang, J.; Lu, W.; Chen, J.; Lin, Q.; Zhao, P.; Kang, Y.; Zhang, H.; Xiong, Y.; Gao, F.; Xu, Z.; et al. 2020. How to mitigate the incident? an effective troubleshooting guide recommendation technique for online service systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 1410–1420.

kafka. 2023. apache kafka. https://kafka.apache.org/. Accessed: 2023-07-20.

Khattab, O.; and Zaharia, M. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, 39–48. New York, NY, USA: Association for Computing Machinery. ISBN 9781450380164.

Kim, S. N.; Wang, L.; and Baldwin, T. 2010. Tagging and linking web forum posts. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, 192–202.

Li, H.; Zhao, X.; Xing, Z.; Bao, L.; Peng, X.; Gao, D.; and Zhao, W. 2015. amAssist: In-IDE ambient search of online programming resources. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 390–398. IEEE.

Liu, J.; Baltes, S.; Treude, C.; Lo, D.; Zhang, Y.; and Xia, X. 2021. Characterizing search activities on stack overflow. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 919–931.

Liu, X.; Wang, C.; Leng, Y.; and Zhai, C. 2018. Linkso: a dataset for learning to retrieve similar question answer pairs on software development forums. In *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering*, 2–5.

Meldrum, S.; Licorish, S. A.; and Savarimuthu, B. T. R. 2017. Crowdsourced knowledge on stack overflow: A systematic mapping study. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 180–185.

Microsoft. 2023. Microsoft Support Community. https://answers.microsoft.com/.

OpenInfra Foundation. 2023. OpenStack Webpage. https://openstack.org/. Viewed on 16 March 2023.

Peinelt, N.; Nguyen, D.; and Liakata, M. 2020. tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7047–7055. Online: Association for Computational Linguistics.

Ponzanelli, L.; Bavota, G.; Di Penta, M.; Oliveto, R.; and Lanza, M. 2014. Mining stackoverflow to turn the ide into a self-confident programming prompter. In *Proceedings of the 11th working conference on mining software repositories*, 102–111.

Quora, Inc. 2023. Quora Homepage. https://quora.com/.

Rahman, M. M.; Yeasmin, S.; and Roy, C. K. 2014. Towards a context-aware IDE-based meta search engine for recommendation about programming errors and exceptions. In *2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, 194–203. IEEE.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Salemi, A.; and Zamani, H. 2024. Evaluating Retrieval Quality in Retrieval-Augmented Generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, 2395–2400. New York, NY, USA: Association for Computing Machinery. ISBN 9798400704314.

Siriwardhana, S.; Weerasekera, R.; Wen, E.; Kaluarachchi, T.; Rana, R.; and Nanayakkara, S. 2023. Improving the Domain Adaptation of Retrieval Augmented Generation (RAG) Models for Open Domain Question Answering. *Transactions of the Association for Computational Linguistics*, 11: 1–17.

Spark. 2023. Apache Spark. https://spark.apache.org/. Accessed: 2023-07-20.

Stack Overflow. 2023. Stack Overflow Webpage. https://stackoverflow.com/. Viewed on 16 March 2023.

Wang, Y.; Le, H.; Gotmare, A. D.; Bui, N. D.; Li, J.; and Hoi, S. C. H. 2023. CodeT5+: Open Code Large Language Models for Code Understanding and Generation. *arXiv preprint*.

Wu, Y.; Wang, S.; Bezemer, C.-P.; and Inoue, K. 2019. How do developers utilize source code from stack overflow? *Empirical Software Engineering*, 24: 637–673.

Xu, B.; Xing, Z.; Xia, X.; and Lo, D. 2017. AnswerBot: Automated generation of answer summary to developers' technical questions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 706–716.

Yang, D.; Hussain, A.; and Lopes, C. V. 2016. From query to usable code: an analysis of stack overflow code snippets. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, 391–401. IEEE.

Table 1: Accuracy comparison on 24 OpenStack cases. The numbers are ranks of the target Stack Overflow document (the ground truth). The number of total ranked Stack Overflow is 3,451 for OpenStack. NF indicates 'Not Found'. Green cells indicate that our technique obtained the best.

Case No.	Type of Case	BM25	Sentence BERT	Ensemble	Google Search	**Ours**
O1	Launch Server (Case1)	5	39	6	5	**1**
O2	Launch Server (Case2)	**1**	7	**1**	8	4
O3	List Server	29	267	31	**2**	4
O4	List Compute Services	**1**	**1**	**1**	**1**	**1**
O5	Create Image	**1**	223	4	2	3
O6	List Network Services	**1**	3	**1**	**1**	**1**
O7	Create Network	**1**	**1**	**1**	**1**	**1**
O8	Create Cloud	**1**	2	**1**	**1**	**1**
O9	List Servers (Case1)	**1**	6	**1**	**1**	**1**
O10	List Projects	43	1,559	147	2	**1**
O11	Create Keypair	**1**	**1**	**1**	**1**	**1**
O12	Add Custom Filter	**1**	**1**	**1**	**1**	**1**
O13	Request Token	**1**	2	**1**	**1**	**1**
O14	Pull Image with Ansible	**1**	**1**	**1**	10	**1**
O15	List Servers (Case2)	21	12	4	**2**	**2**
O16	Launch Server (Case3)	**1**	**1**	**1**	NF	2
O17	Launch Server (Case4)	4	80	7	3	**1**
O18	Attach Volume to Server	4	154	13	**1**	**1**
O19	Create Domain	**1**	**1**	**1**	2	**1**
O20	Launch Server (Case5)	85	54	40	5	**3**
O21	Restart Compute Service	**1**	26	2	4	2
O22	Launch Server (Case6)	94	3	5	**1**	**1**
O23	Launch Server (Case7)	**1**	**1**	**1**	NF	**1**
O24	Launch Server (Case8)	3	13	5	NF	**2**

Table 2: Accuracy comparison on 22 MongoDB cases. The number of total Stack Overflow documents is 172,209.

Case No.	Type of Case	BM25	Sentence BERT	Ensemble	Google Search	**Ours**
M1	Connect MongoDB (Case1)	99	16	4	2	**1**
M2	Connect MongoDB (Case2)	22	1,663	115	**1**	2
M3	Search with id Entries	51	**1**	**1**	NF	**1**
M4	Mongoimport with Atlas	**1**	16	**1**	**1**	**1**
M5	Restore Collection	**1**	**1**	**1**	**1**	**1**
M6	Connect MongoDB from Node.js	57	85	18	**1**	**1**
M7	Mongodump from remote server	**1**	17	**1**	2	4
M8	Start and Connect MongoDB	76	7	13	**3**	**3**
M9	Delete Document with Query	110,025	**1**	15	NF	4
M10	Load and Restore Data	**2**	15	**2**	NF	3
M11	Create Backup with MongoDump	2	**1**	**1**	9	**1**
M12	Collection with insert_one	3	9	**1**	**1**	1
M13	Insert Data into Collection	17	**1**	2	4	**1**
M14	Retrieve Sorted Data	5	**1**	**1**	4	2
M15	Insert Data with Pymongo	72	23	3	NF	3
M16	Sort and Get Record	314	43	15	NF	**1**
M17	Access Document with find	3	13	2	3	**1**
M18	Update Collection	**1**	58	1	6	5
M19	Connect MongoDB and Insert	50	75	26	**1**	2
M20	Connect MongoDB with Pymongo	3,314	54	217	5	3
M21	Configure Mongdb Replicaset	2,220	80	171	NF	4
M22	Retrieve a distinct list of value	13	**1**	**1**	4	2

Table 3: Search accuracy comparison on 18 Spark cases. The number of total Stack Overflow documents is 100,640.

Case No.	Type of Case	BM25	Sentence BERT	Ensemble	Google Search	**Ours**
S1	KMeans with Python	24	10	4	7	**1**
S2	Connect Mongodb from spark	**1**	24	3	**1**	3
S3	Calling Java/Scala Function	35	**1**	3	69	**1**
S4	Create DataFrame with Condition	24	15,566	139	NF	**1**
S5	Use Multiple Conditions	863	**1**	**1**	26	**1**
S6	Pyspark Mongodb Connector	60	19	21	13	**2**
S7	Check Dataframe with Condition	389	27,254	1,731	23	**5**
S8	Create Spark Session	10	20	2	**1**	**1**
S9	Create User Defined Function	**1**	204	**1**	4	3
S10	Read and Write Table	162	3	7	**1**	**1**
S11	Set Spark Configuration	34	279	19	**1**	**1**
S12	Delete and Recreate Context	**1**	50	4	2	**1**
S13	Create DataFrame	6	22	4	NF	**2**
S14	Run Logistic Regression	90	36	6	**1**	**1**
S15	Run Mllib in Pyspark	9,032	338	816	NF	**1**
S16	Initializing Spark Context	**1**	2	**1**	3	**1**
S17	Run Spark on Remote Cluster	1,722	340	423	NF	60
S18	Run Spark in Cluster Mode	1,120	10,423	2,128	**12**	57

Table 4: Search accuracy comparison on 8 Elasticsearch cases. The number of Stack Overflow documents is 57,523.

Case No.	Type of Case	BM25	Sentence BERT	Ensemble	Google Search	**Ours**
E1	Add Range Filter to Search Query	**1**	**1**	**1**	4	**1**
E2	Send Request with Apache HttpClient	**1**	**1**	**1**	**1**	**1**
E3	Re-Index	**1**	**1**	**1**	3	2
E4	Create Index and Mapping	**1**	**1**	**1**	2	3
E5	Create Transport Client and Search	177	1,000	203	70	**34**
E6	Index a New Document	16	8	6	5	**1**
E7	Read Document with Index	9	**1**	**1**	**1**	**1**
E8	Take a Snapshot of Index	104	**1**	2	**1**	7

Table 5: Search accuracy comparison on 5 Kafka cases. The number of total Stack Overflow documents is 30,896.

Case No.	Type of Case	BM25	Sentence BERT	Ensemble	Google Search	**Ours**
K1	Send Large Message	1,085	75	91	11	**1**
K2	Create Kafka Cluster and Send Message	8,615	5	114	5	**1**
K3	Consume Message with Zookeeper	6	41	**1**	4	22
K4	Create Producer	502	50	72	**1**	**1**
K5	Connect Kafka with Kafka-Python	23	74	7	89	**1**