Aligning Documentation and Q&A Forum through Constrained Decoding with Weak Supervision

Rohith Pudari
University of Toronto
r.pudari@mail.utoronto.ca

Shiyuan Zhou
University of Toronto
shiyuan.zhou@mail.utoronto.ca

Iftekhar Ahmed
University of California, Irvine
iftekha@uci.edu

Zhuyun Dai
Google
zhuyundai@google.com

Shurui Zhou
University of Toronto
shurui.zhou@utoronto.ca

***Abstract*—Stack Overflow (*SO*) is a widely used question-and-answer (Q&A) forum dedicated to software development. It plays a supplementary role to official documentation (*DOC* for short) by offering practical examples and resolving uncertainties. However, the process of simultaneously consulting both the documentation and *SO* posts can be challenging and time-consuming due to their disconnected nature. In this study, we propose *DOSA*, a novel approach to automatically align *SO* and *DOC*, which inject domain-specific knowledge about the *DOC* structure into large language models (LLMs) through weak supervision and constrained decoding, thereby enhancing knowledge retrieval and streamlining task completion during the software development procedure. Our preliminary experiments find that *DOSA* outperforms various widely-used baselines, showing the promise of using generative retrieval models to perform low-resource software engineering tasks.**

***Index Terms*—Stack Overflow, Natural language processing, Constrained Decoding, Weak Supervision**

I. Introduction & Background

Stack Overflow (*SO*) is a popular Q&A forum that has garnered over 23M software engineering (SE) related questions posed by 18M users as of June 2023 [1]. Earlier research indicated that *SO* is confronted with the issue of excessive information and lacks an effective way to manage the contents [2], [3]. Therefore, solutions are proposed to improve *SO*, such as recommending the appropriate tags for each question [2], [4], [5], help individuals use *SO* more effectively [6], [7], detect duplicate *SO* questions using titles and descriptions [8], and link two related *SO* posts [9] for better information dispersion.

Prior work also demonstrated that *SO* users sometimes manually link official documentation ('*DOC*' for short) as references during the discussion since *DOC* could provide a complimentary role for *SO* by providing official explanations [10] (see Fig. 1 as an example). According to our analysis (explained later), 54 out of 200 sampled *SO* questions with the tag '*Python*', have at least one link to the *DOC* in their answers. Nevertheless, the task of concurrently referring to both *DOC* and *SO* posts can be difficult and time-consuming because they are not seamlessly integrated [10], [11]. Therefore, in this work, we aim for designing an approach to automatically align *DOC* and *SO* questions.

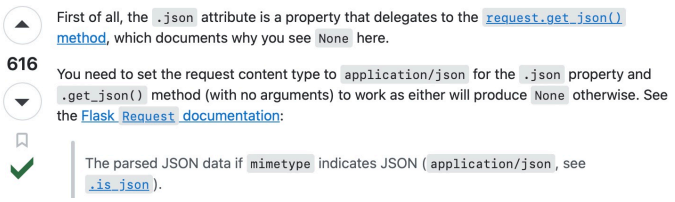

Fig. 1. An example from *SO* where official documents are linked to support a better understanding of the context [12].

However, accomplishing this task poses a difficulty given the scarcity of training data. As far as we know, there is currently no public dataset available that includes the alignment between *SO* questions and the relevant sections in the *DOC*. Creating a large-size custom dataset by using labor-intensive manual labeling is not scalable, making it not feasible to directly and efficiently employ conventional machine learning (ML) models that are data-hungry.

Recent large language models (LLMs) have been proven to be potentially applicable to this problem, which have undergone pre-training on vast text and code collections, demonstrating impressive capabilities in *zero-shot learning* [13] and *few-shot learning* [14][1] scenarios across various Natural Language Processing (NLP) assignments [15] such as generative retrieval [16]. They have also shown promising performance on a variety of SE tasks, such as generating code [17] and test cases [18], and even answering *SO* questions [19]. Nevertheless, it remains an open research problem for using LLMs to align *SO* and *DOC*. Importantly, as LLMs generate free-form text, it is unclear how to use them to retrieve an existing piece of official documentation for a question instead of generating a new, hallucinated text [20].

The closest work to our idea is from Treude et al. [11], which focuses on augmenting Java API *DOC* by extracting manually annotated meaningful sentences from *SO* answers, which is similar to our baseline approach *Pyserini*, which

[1]**Zero-shot**: The model can generalize and make predictions on classes it has never seen during training; **Few-shot**: The model learns to extract relevant features and the underlying patterns in the few available examples, enabling it to make predictions on unseen classes.

performs poorly in our problem setting (as we will explain later). Different from this work, we leverage *generative retrieval models* with domain-specific knowledge to solve the problem.

Specifically, we propose *DOSA* (short for *DOC & SO* Alignment), a novel approach to turn LLMs into a domain-specific retriever capable of determining the specific section within the *DOC* to which a given *SO* question belongs. We first design a *weak supervision* technique to adapt LLM to the specific *DOC*. Then, we employ *constrained decoding* [21], [22] to cast LLM's generation into retrieval by constraining that the LLM must generate an existing category from *DOC* followed by an existing sub-category under that category. With this process, we can generate predictions on the most relevant existing section of *DOC* for the input *SO* question without hallucination. To the best of our knowledge, we are the first to *use generative retrieval for aligning multi-modality SE resources*.

We evaluated *DOSA* on two documentation sites, Python and Flask, which are popular libraries used in web application development. As the starting point, we focus on evaluating the feasibility of the proposed method. The preliminary results show that our approach exhibits significant improvements over the state-of-the-art (SOTA) LLMs, such as GPT-2 [23] and LLaMA-7B [24], with an average precision and recall increase of (22%, 18%) and (43%, 49%) respectively. Our ablations also find that *DOSA* can achieve reasonable performance by just relying on pre-trained LLMs without any additional training.

In sum, our contributions include (1) designing a novel approach to augment LLMs with domain-specific knowledge through weak supervision and constrained decoding for the knowledge retrieval task of connecting *SO* questions with corresponding *DOC*; (2) constructing two datasets, Python and Flask, by aligning *SO* and *DOC* with human-annotated labels for training and evaluation, to encourage future research. All of the source code and the constructed datasets can be accessed at https://doi.org/10.5281/zenodo.8036663.

II. Our Approach – *DOSA*

In this section, we first explain the construction of the training and evaluation dataset, then we justify the feasibility of LLM for the defined task. Lastly, we describe the two main components/steps of *DOSA*, including (1) weakly-supervised LLM adaptation on *DOC*, and (2) using LLM to retrieve a category of *DOC* for a given *SO* question via constrained decoding. We present the overview of *DOSA* in Fig. 3.

A. Training and Evaluation Datasets

As a starting point, we chose two popular topics, Python [25] and Flask [26], to study the feasibility of our method. We organize the content and structures of official documentation to facilitate model training. We also randomly sampled a subset of *SO* questions and manually labeled them as ground truth for evaluation purposes.

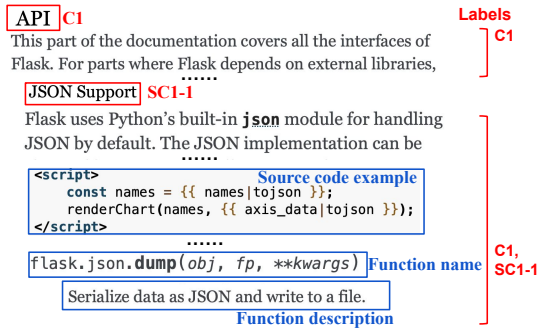

Fig. 2. Example from Flask documentation used for constructing training data, which contains two labels – one category (C) and one sub-category (SC). C1 is the heading of this section parsed from the HTML page, and SC1-1 is the subheading. The texts under each heading are labeled by the corresponding C&SC.

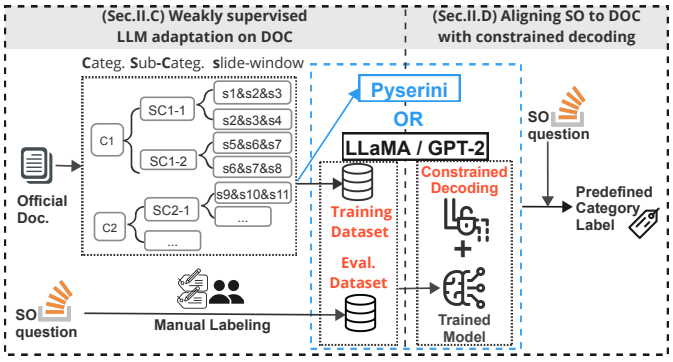

Fig. 3. Research method overview. C – Category, SC – Sub-category, s – sentence. Pyserini, GPT-2, and LLaMA are three baselines of our method. The texts highlighted in red are our contribution.

1) Constructing training datasets from DOC*:* We parsed the HTML page of the corresponding *DOC* [25], [26] using a web scraper [27] and extracted section headings, function descriptions, code snippets, and other textual content. The heading of a section is regarded as the *category* for the text contained in that section. Likewise, the sub-section heading is seen as the *sub-category* for the corresponding content. For example, the content presented in Fig. 2 is a screenshot from the Flask documentation, which includes the *API* section and *JSON Support* sub-section. In our study, we defined "*API*" as the first-level category (C) and "*JSON Support*" as the sub-category (SC).

Note that our current method design solely captures two tiers of headings from the official documentation as a proof-of-concept. Subsequent efforts could involve incorporating additional levels of categories to attain more accurate labeling outcomes. Similarly, we treat both code and text as natural language and do not consider the syntactic meaning of source code. We also treat the whole code block as one sentence for ease of implementation at the current stage. Future work could include analyzing the syntax and semantics of the source code, allowing for a more comprehensive understanding of source code functionality and intent. In total, the Python dataset contains 35 categories and 1644 sub-categories; the

Flask dataset contains 28 categories and 381 sub-categories.

2) Constructing evaluation datasets from SO *questions:* This step aims at finding the mapping between the *SO* questions and the corresponding categories (equivalent to the headings in the *DOC*). We first utilized the SOTorrent dataset [28] and the Stack Exchange Data Dump [1] to collect all the questions with the tags '*Python*' or '*Flask*'. In total, we collected 1.8M questions with the tag '*Python*' and 52K with the tag '*Flask*'. According to prior work and also our observations, we found that the tags of each *SO* question are not unified and could be duplicated and messy [29], which is not reliable to be used as the ground truth labels. Therefore, we decided to manually construct our own ground truth dataset.

However, due to the substantial volume of data and heavy workload of manual labeling, we utilized a sampling size calculator with a confidence level and margin of error set at 85% and 5% respectively and randomly sampled 200 questions from the *SO* dataset for each of the '*Python*' and '*Flask*' tags. Two of the authors independently labeled the sampled questions based on their relevance to specific topics within Python or Flask documentation. Each question was assigned a single label that best represented its primary category and sub-category of documentation. This labeling process involved examining keywords within the questions, utilizing the documentation search feature, and manually cross-referencing with the top search results to ensure alignment with the corresponding documentation sections. After the initial labeling, the two authors held an alignment session to compare and discuss their labeled results. During this session, any discrepancies in labeling were identified and discussed to reach a consensus on the correct labels for each question. The Cohen's Kappa [30] score of 0.83 indicated a near-perfect level of agreement between the two author's labeling decisions.

B. The feasibility of leveraging the LLM

Formally, we define the task of assigning labels to a *SO* question (denoted as q) using the extracted categories and sub-categories (denoted as C and SC) from a provided SE official documentation (denoted as DOC):

$$q \rightarrow \{C, SC\} \in DOC \tag{1}$$

Since there is currently no public dataset available that includes the mapping between *SO* questions and the relevant category in the *DOC*, it is infeasible to apply conventional ML approaches that need lots of human-labeled training data. Therefore, we propose a novel approach, *DOSA*, that leverages LLMs (e.g., GPT-2 or LLaMA) to perform this task in a zero-shot manner. Specifically, we speculate that the language comprehension capabilities of LLMs (e.g., LLaMA [24] or GPT-4 [15]) fine-tuned on *DOC* could establish the connection between the inherent meaning and context of the *SO* question and the content of the *DOC* eliminating the need for explicit training data. For example, in Fig. 1, the word "property" in the first sentence holds distinct meanings in everyday English usage compared to its significance within a programming context making LLMs a good choice to perform this task in a zero-shot manner.

C. Weakly-supervised LLM adaptation on DOC

The majority of publicly available LLMs are pre-trained on a general-domain corpus, so their knowledge of a specific SE domain's official documentation can be limited or outdated, as it requires re-collecting and processing large amounts of new data, which may not always be feasible or practical to do frequently. The first step in *DOSA* injects knowledge about the specific *DOC* into the LLM by training it with the contents of *DOC*, which acts as a form of weak supervision. Weak supervision in this context refers to the use of imprecise or noisy information, such as the content of *DOC*, to guide the alignment of *SO* questions to categories and sub-categories extracted from *DOC*.

$$t \rightarrow (C_t, SC_t) \tag{2}$$

where t is a chunk of text from DOC, C_t and SC_t are its corresponding category and sub-category.

We use a *sliding-window approach* to generate chunks of text t from the documentation DOC, meaning a fixed-size window that is moved across the collected text, capturing subsets of sequential words or phrases. This way, the model can be fine-tuned to analyze multiple sentences together, thus providing a broader context for learning to predict the chunk's category/sub-category. After conducting several pilot experiments with different window sizes on the dataset, we found that grouping three (3) consecutive sentences together from the *DOC* yields the best performance. On the other hand, larger window sizes may introduce noise or unnecessary complexity, hindering the model's ability to generalize effectively. This window size serves as the "*sweet spot*" for capturing the context within the text.

D. Constrained Decoding for Producing Targeted Output

One of the challenges of directly applying LLMs to the problem that *DOSA* is focusing on is that a standard LLM picks one token from the entire vocabulary as the next token at each decoding step. But in our task, we expect the model's output space should follow the structure of *DOC* – only consisting of existing categories and the corresponding sub-categories defined in the *DOC*. Thus, we leverage *constrained decoding* [21], [22] technique, which allows us to incorporate domain terminology during the token generation process, enabling the models to produce more targeted outputs.

Given the example showed in Fig. 1, a standard LLM (i.e., GPT-2 model in this case) will generate an output sequence – [*post, postman, text, print, build, ...*], which is sorted by the order of probability. However, none of the first five tokens is a category defined in the Flask *DOC*, which is not useful to fulfill our task as it does not allow the ability to align it with *DOC*. After adding the step of *constrained decoding*, for the same example above, the top-5 produced tokens are transformed into [*JSON, API, application, handling, quickstart*], with a descending order of probability and are all within the set of

existing categories and sub-categories collected from the *DOC*. This enables us to align these tokens to *DOC* as they follow the same categorical structure as defined in the *DOC*.

This process continues till the final output sequence matches one of the predefined category or sub-category labels. Take the multi-token category of "*JSON Support*" (see *SC1-1* in Fig. 2) as an example, our approach produces the word "*JSON*" in the first decoding step and then generates "*support*" in the next decoding step, and eventually stops by finding "*JSON support*" matches a sub-category label from the Flask documentation.

Note that constrained decoding only operates during the generation process at *test* time; it does not change the LLM's parameters and the knowledge encoded in those parameters ("parametric knowledge") [21], [22]. Hence, as we will show in the experiments, its effectiveness heavily relies on how much the LLM already knows about the domain.

E. Implementation

Since *DOSA* is designed to be built on top of the SOTA LLMs, we select two LLMs for a feasibility study: GPT-2 [23] and LLaMA [24] (as shown in Fig. 3).

- ***GPT-2*** is pre-trained on text written in English using a causal language modeling objective and has been used in tasks like text generation and summarization. It is suitable for our task because it could understand the context and detect the complex patterns in the *DOC*. Note that we could not use the newer versions of this model like GPT-4 because they are closed-source, but we need to make changes to the model architecture, such as implementing constrained decoding and adding a classification head, to suit our task as a classification problem. We used the default medium-size variation [31], which has 355 million parameters, denoted as *DOSA*(GPT-2).
- ***LLaMA*** is an auto-regressive language model, whose training dataset consisted of stack exchange data among other sources. LLaMA is one of the recent models (launched in Feb 2023) and has been proven to outperform GPT-3 [24] in most benchmarks [24]. This was the largest open-source model available while testing. We used the default 7 billion parameter variation from HuggingFace [32], denoted as *DOSA*(LLaMA-7B).

III. Evaluation and Discussion

Since *DOSA* uses LLM as a knowledge retriever, it is important to compare it against conventional retrieval systems. Thus, we include a SOTA retrieval-based baseline, *Pyserini [33]*, which is a widely-used reproducible Information Retrieval (IR) toolkit in the field of NLP for indexing a collection of documents and querying the index to retrieve relevant documents. Pyserini supports efficient indexing and retrieval of large-scale collections, making it applicable for a wide array of applications, including IR research, question-answering systems, and document ranking. It provides a Pythonic API for indexing and searching large document collections using various retrieval models. We used the BM25 retrieval model [34], a popular ranking function with a strong performance on zero-shot retrieval tasks [35]. It assigns a relevance score to each document in a collection based on the query terms and their frequencies within the document. The BM25 algorithm is widely used in search engines and other text retrieval applications. As shown in Fig. 3, *Pyserini* does not require training, and we follow the default hyperparameters for inference. This is an appropriate baseline for our task since it can index *DOC* pages and retrieve pertinent categories and sub-categories of *DOC* by matching common keywords found in *SO* questions.

TABLE I
PRECISION (P) AND RECALL (R) SCORES OF DIFFERENT MODELS. WK – WEAK SUPERVISION, CD – CONSTRAINED DECODING. LLaMA-7B + WK DENOTES THAT THE MODEL WAS IMPLEMENTED WITH LLaMA-7B AND ONLY HAS *weak supervision* BUT NOT *constrained decoding*.

Model \ Topic	Flask		Python	
	Cat.	Sub-Cat.	Cat.	Sub-Cat.
Pyserini	P: 0.26 R: 0.19	P: 0.03 R: 0.10	P: 0.17 R: 0.13	P: 0.08 R: 0.06
***DOSA*(GPT-2)**	**P: 0.98** **R: 0.99**	**P: 0.99** **R: 1.0**	P: 0.98 R: 0.94	P: 0.33 R: 0.41
***DOSA*(LLaMA-7B)**	P: 0.93 R: 0.94	P: 0.97 R: 0.91	**P: 0.94** **R: 0.96**	**P: 0.62** **R: 0.57**
GPT-2 + WK	P: 0.61 R: 0.60	P: 0.66 R: 0.61	P: 0.52 R: 0.55	P: 0.19 R: 0.20
LLaMA-7B + WK	P: 0.48 R: 0.51	P: 0.32 R: 0.39	P: 0.69 R: 0.61	P: 0.19 R: 0.11
GPT-2 + CD	P: 0.30 R: 0.22	P: 0.14 R: 0.15	P: 0.11 R: 0.08	P: 0.09 R: 0.08
LLaMA-7B + CD	P: 0.80 R: 0.81	P: 0.76 R: 0.78	P: 0.81 R: 0.82	P: 0.41 R: 0.40
LLaMA-13B + CD	P: 0.87 R: 0.88	P: 0.82 R: 0.84	P: 0.80 R: 0.83	P: 0.44 R: 0.51

In this section, we compare the performance of *DOSA* with baselines and also dive deeper into understanding *DOSA* through ablation studies to evaluate the importance of (1) *Weak supervision*, (2) *constrained decoding*, and (3) the benefit of model scaling. We report all the experiment results in Tab. I.

A. Conventional IR vs. LLMs

Our results show that LLMs are better suited for our task compared to the conventional IR approach (i.e., *Pyserini*), as LLMs excel at tasks requiring natural language understanding, inference, and even generating human-like responses [23], [24]. *Pyserini*, on the other hand, is primarily designed for IR and indexing tasks, which rely on traditional techniques such as bag-of-words, which may struggle to capture the nuanced and contextual nature of *SO* questions and *DOC*. These findings underscore the suitability of LLMs for our specific task.

B. How important is constrained decoding?

The results show that including both *weak supervision* and *constrained decoding* (i.e., *DOSA*(GPT-2) and *DOSA*(LLaMA-7B)) exhibits significant improvements over the versions without *constrained decoding* (i.e., GPT-2 + WK and LLaMA-7B + WK), with an average precision and recall increase of (22%, 18%) and (43%, 49%) respectively.

A potential explanation is that LLMs heavily rely on pre-training and fine-tuning on large corpora [24]. This makes

TABLE II
HALLUCINATION RATES WHEN MODEL ONLY HAS WEAK SUPERVISION.

Model \ Topic	Flask		Python	
	Category	Sub-Cat.	Category	Sub-Cat.
GPT-2 + WK	5%	11%	6%	9%
LLaMA-7B + WK	17%	25%	20%	28%

them susceptible to biases and noise present in the training data. Without specifying the scope of the output labels, the fine-tuned models may generate plausible labels but not in direct alignment with *DOC*.

Constrained decoding helps address this issue by incorporating specific constraints (i.e., only output labels within the defined scope) during the decoding process. It also reduces hallucinations by guiding the model to generate labels that are from the list of categories and sub-categories from *DOC*. In our experiments, the introduction of constrained decoding led to a notable reduction in hallucination rates. Specifically, the hallucination rate diminished by approximately 8% for GPT-2 and 22% for LLAMA (see Tab. II). This reduction highlights the effectiveness of constrained decoding in enhancing the alignment accuracy of LLMs in the context of our task. The effectiveness relies on the model's capacity to handle and respect these constraints. Larger models like *LLaMA* have higher parameter counts and more expressive power, allowing them to better accommodate and leverage these constraints, resulting in more precise alignments and bigger performance gains than the fine-tuned-only version.

C. Is the weakly-supervised adaptation necessary?

The weakly-supervised adaptation step aims to inject knowledge about *DOC* to the LLM, but it can be expensive as it requires training the LLM. Since LLaMA is trained on vast amounts of publicly available text corpora [24], it might have already seen and memorized the documentation sites during pre-training and, therefore might not need the weak supervision.

To understand the importance, we remove the weak supervision component from our method (a.k.a zero-shot) and only keep the *constrained decoding* component, denoted as GPT-2 + CD and LLaMA-7B + CD. The results show that the introduction of *weak supervision* has yielded an approximate enhancement of 10% at the category level and 20% at the sub-category level for both Python and Flask. This improvement, however, comes with a trade-off between convenience and accuracy. The zero-shot approach, which does not require explicit training, offers convenience but may result in less precise alignments. On the other hand, models fine-tuned with weak supervision tend to achieve higher accuracy at the cost of additional training efforts.

With constrained decoding only, LLaMA-7B + CD is capable of aligning *SO* questions to relevant pieces of documentation with an average precision and recall of (0.69, 0.70), however, GPT-2 + CD could only achieve an average precision and recall of (0.16, 0.13). This indicates that LLaMA's pre-training phase has already provided a good amount of programming-related knowledge as compared to GPT-2. Performance of LLaMA-7B + CD is a significant advantage as it allows users to immediately benefit from our approach without the time and resources required for training their own models. It also sheds light on next-generation knowledge retrievers for SE tasks, as one can easily achieve a retriever with reasonable performance by converting an LLM using constrained generation, without any LLM weak supervision.

D. Can DOSA benefit from model scaling?

We further investigate the effects of scaling in terms of the size of the model, to find if larger language models always yield better performance for our task. To this end, we scale the model size by performing experiments on LLaMA from 7 billion (denoted as LLaMA-7B + CD) to 13 billion (denoted as LLaMA-13B + CD). Our results demonstrate that larger language models perform better, particularly when combined with our zero-shot constrained decoding approach. The superiority of larger models becomes especially evident when no task-specific fine-tuning is applied, showcasing the power of their pre-trained knowledge and capacity for generalization. The observed scaling effects indicate that advances made in LLM pre-training directly correspond to enhancements in *DOSA*'s performance. As researchers continue to develop more advanced and larger LLMs, the quality of *DOSA* is expected to consistently improve.

IV. LIMITATIONS AND FUTURE WORK

While our approach, *DOSA*, has showcased promising outcomes in aligning *SO* questions with relevant categories and sub-categories in *DOC*, there exist certain limitations that require further attention. Primarily, the current testing of our approach has been confined to only two specific topics. For its application to diverse programming contexts, additional training data and potential adjustments to the classification model might be necessary. Another limitation arises from the manual labeling process involved in constructing the evaluation dataset. Although undertaken meticulously, the human judgment factor could introduce inconsistencies. Despite achieving a high Cohen's Kappa score, the inherent subjectivity of labeling might introduce a degree of error or discordance in the ground truth labels.

Despite many advancements in newer iterations of LLMs, these models still encounter difficulties in direct retrieval tasks [15], [24]. As a result, our approach offers a practical solution to the persisting problem of aligning *SO* to *DOC*, providing a crucial bridging mechanism between advanced LLMs and the realm of precise information retrieval. We established a foundation for aligning various documentation resources within the same domain. Future work can capitalize on these findings to not only amplify the generalizability, performance, and utility of the proposed approach but also to extend its application to broader domains beyond programming. Exploring innovative methods for categorizing diverse types of technical documents is also a promising avenue for future exploration.

V. Conclusion

In this paper, we present an effective approach, *DOSA*, for aligning *SO* questions to the relevant sections of *DOC*. By combining *constrained decoding* and *weak supervision*, along with leveraging LLMs, we achieved significant improvements in alignment performance compared to other baseline techniques. Our results highlight the importance of incorporating domain-specific knowledge, utilizing constraints, and leveraging additional sources of weak supervision to enhance alignment performance. Furthermore, we demonstrated that LLMs, with their broader knowledge base and capacity for generalization, outperformed smaller models, especially in the zero-shot setting. Although generative retrieval has been explored in other NLP tasks such as entity retrieval, to the best of our knowledge, this result is the first to use generative retrieval on SE tasks. Overall, this research lays a foundation for aligning multiple resources of documentation on the same topic.

References

[1] "Stack exchange data dump: Stack exchange, inc." [Online]. Available: https://archive.org/details/stackexchange

[2] T. Saini and S. Tripathi, "Predicting tags for stack overflow questions using different classifiers," in *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 2018, pp. 1–5.

[3] R. Rubei, C. Di Sipio, P. T. Nguyen, J. Di Rocco, and D. Di Ruscio, "Postfinder: Mining stack overflow posts to support software developers," *Information and Software Technology*, vol. 127, p. 106367, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584920301361

[4] S. K. Maity, A. Panigrahi, S. Ghosh, A. Banerjee, P. Goyal, and A. Mukherjee, "DeepTagRec: A content-cum-user based tag recommendation framework for stack overflow," in *European conference on information retrieval (ECIR)*, pp. 125–131.

[5] J. He, B. Xu, Z. Yang, D. Han, C. Yang, and D. Lo, "PTM4tag," in *Proceedings of the 30th IEEE/ACM international conference on program comprehension (ICPC)*. ACM.

[6] S. Beyer and M. Pinzger, "A manual categorization of android app development issues on stack overflow," in *2014 IEEE international conference on software maintenance and evolution (ICSME)*, pp. 531–535.

[7] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?: NIER track," in *2011 33rd international conference on software engineering (ICSE)*, pp. 804–807.

[8] R. F. G. Silva, K. Paixao, and M. de Almeida Maia, "Duplicate question detection in stack overflow: A reproducibility study," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE.

[9] B. Xu, D. Ye, Z. Xing, X. Xia, G. Chen, and S. Li, "Predicting semantically linkable knowledge in developer online forums via convolutional neural network," in *2016 31st IEEE/ACM international conference on automated software engineering (ASE)*, pp. 51–62.

[10] S. Baltes, C. Treude, and M. P. Robillard, "Contextual documentation referencing on stack overflow," vol. 48, no. 1, pp. 135–149, publisher: Institute of Electrical and Electronics Engineers (IEEE).

[11] C. Treude and M. P. Robillard, "Augmenting API documentation with insights from stack overflow," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pp. 392–403, ISSN: 1558-1225.

[12] How to get POSTed JSON in flask? [Online]. Available: https://stackoverflow.com/q/20001229/13285998

[13] M. Palatucci, D. Pomerleau, G. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, ser. NIPS'09. Curran Associates Inc., pp. 1410–1418.

[14] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," vol. 53, no. 3, pp. 63:1–63:34.

[15] OpenAI, "GPT-4 technical report." [Online]. Available: http://arxiv.org/abs/2303.08774

[16] N. D. Cao, G. Izacard, S. Riedel, and F. Petroni, "Autoregressive entity retrieval," in *International Conference on Learning Representations (ICLR) 2020*.

[17] A. Karmakar and R. Robbes, "What do pre-trained code models know about code?" in *IEEE/ACM international conference on automated software engineering (ASE)*, pp. 1332–1336.

[18] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "Adaptive test generation using a large language model," tex.copyright: arXiv.org perpetual, non-exclusive license. [Online]. Available: https://arxiv.org/abs/2302.06527

[19] S. Kabir, D. N. Udo-Imeh, B. Kou, and T. Zhang, "Who answers it better? an in-depth analysis of ChatGPT and stack overflow answers to software engineering questions," version: 3. [Online]. Available: http://arxiv.org/abs/2308.02312

[20] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," vol. 55, no. 12, pp. 1–38.

[21] C. Hokamp and Q. Liu, "Lexically constrained decoding for sequence generation using grid beam search," in *Proceedings of the 55th annual meeting of the association for computational linguistics (ACL) (volume 1: Long papers)*. Association for Computational Linguistics.

[22] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Guided open vocabulary image captioning with constrained beam search," in *Proceedings of the 2017 conference on empirical methods in natural language processing (EMNLP)*. Association for Computational Linguistics.

[23] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners."

[24] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and efficient foundation language models," tex.copyright: Creative Commons Attribution 4.0 International. [Online]. Available: https://arxiv.org/abs/2302.13971

[25] The python standard library. python documentation. [Online]. Available: https://docs.python.org/3/library/index.html

[26] Welcome to flask — flask documentation (2.3.x). [Online]. Available: https://flask.palletsprojects.com/en/2.3.x/

[27] Beautiful soup. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/

[28] S. Baltes, L. Dumani, C. Treude, and S. Diehl, "SOTorrent: Reconstructing and analyzing the evolution of stack overflow posts," in *2018 IEEE/ACM 15th international conference on mining software repositories (MSR)*. IEEE Computer Society, pp. 319–330.

[29] 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, Nov. 2021.

[30] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," vol. 33, no. 1, p. 159, publisher: JSTOR.

[31] gpt2-medium · hugging face. [Online]. Available: https://huggingface.co/gpt2-medium

[32] decapoda-research/llama-7b-hf · hugging face. [Online]. Available: https://huggingface.co/decapoda-research/llama-7b-hf

[33] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. Nogueira, "Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations," in *Proceedings of the 44th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2021)*, pp. 2356–2362.

[34] D. Harman, "NIST special publication 500-236: The fourth text retrieval conference (TREC-4)."

[35] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models," in *Proceedings of the 35th conference on neural information processing systems datasets and benchmarks track (round 2)*.