Two-Stage Ranking Using HyDE and SimCSE for Paper Retrieval

Meisaku Suzuki
NTT DOCOMO, INC.
Tokyo, Japan
meisaku.suzuki.fw@nttdocomo.com

Sho Maeoki
NTT DOCOMO, INC.
Tokyo, Japan
syou.maeoki.rz@nttdocomo.com

Kenichirou Miyaki
NTT DOCOMO, INC.
Tokyo, Japan
kenichirou.miyaki.dk@nttdocomo.com

Abstract

The overall goal of academic data mining is to increase the understanding of development, nature, and trends in science. Academic data mining has the potential to extract enormous scientific, technical, and educational value. The organizers of KDD Cup 2024 published the Open Academic Graph (OAG) Benchmark for Academic Graph Mining, and within it, the OAG-academic question answering (AQA) competition focused on paper retrieval. In this paper, we present a solution that achieved 6th place as DOCOMO-LABS in the public leaderboard in the OAG-AQA competition. This solution proposes a two-stage prediction model. In stage 1, we use query augmentation and contrastive learning to create candidates for paper retrieval. In stage 2, the encoder-based language model is used as a re-ranker to train binary classification and eventually ensembling the predictions of several models.

CCS Concepts

• **Information systems → Digital libraries and archives; Data mining**.

Keywords

KDD Cup, academic knowledge graph; benchmark; academic graph mining

ACM Reference Format:
Meisaku Suzuki, Sho Maeoki, and Kenichirou Miyaki. 2024. Two-Stage Ranking Using HyDE and SimCSE for Paper Retrieval. In *KDDCUP '24: ACM SIGKDD Conference on Knowledge Discovery and Data Mining,August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The volume of published research papers has increased dramatically, and effective data mining from this vast amount of literature holds significant value. For researchers and engineers, gaining specialized knowledge in their respective fields is crucial. However, manually searching for and identifying relevant papers can be a daunting task. Thus, having a system that retrieves the most pertinent papers in response to specific queries would be highly beneficial. To address this need, the organizers of KDD Cup 2024 proposed a task referred to as academic question answering (AQA) [10]. This task involves using a dataset constructed from the Open Academic Graph (OAG) dataset [7] to input questions and output a set of the most relevant paper candidates. In this paper, we introduce a method that achieved 6th place on the public leaderboard for this task. The core approach involves employing a two-stage model that combines a retriever and a re-ranking mechanism. In stage 1, a retrieval model is trained using contrastive learning along with augmented sentences generated using large language models (LLMs) as is performed in Hypothetical Document Embeddings (HyDE) [2]. Then, the candidate items are obtained using the trained model. In stage 2, the obtained candidates are re-ranked with respect to the probability scores ensembled using several different models.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
KDDCUP '24, August 25–29, 2024, Barcelona, Spain
© 2024 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

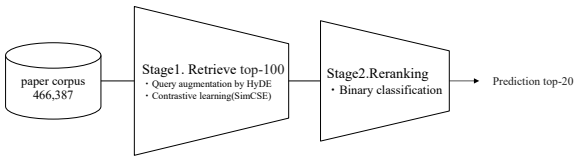

Figure 1: Two-stage architecture including HyDE and SimCSE. In stage 1, 100 candidates are obtained through vector search from a 466,387 paper corpus. In stage 2, the 100 candidates are re-ranked and the top-20 candidates are selected as the final prediction.

2 Related Work

2.1 Question and Answer Retrieval

2.1.1 Classical vs. Neural Retrieval Methods. Traditional retrieval methods such as BM25 [6] have long been used for information retrieval tasks. These methods rely on term frequency and document frequency to score and rank documents. However, recent advances in machine learning have introduced neural dense retrieval methods, which have shown significant improvements in performance.

Neural retrieval typically employs contrastive learning for training. This involves learning to differentiate between relevant and non-relevant document pairs, thus improving the model's ability to retrieve pertinent information. We adopt neural retrieval models, specifically SimCSE [3], thanks to its promising performance.

2.1.2 Retrieval with Large Language Models. In the context of retrieval with LLMs, retrieval-augmented generation (RAG) is one

field that is actively studied in the natural language processing community. In RAG, to increase the performance of retrieval, approaches like HyDE [2] have been widely accepted. HyDE involves generating hypothetical answers using LLMs and leveraging these generated outputs to perform the retrieval task. This method enhances the ability to find relevant documents by utilizing the generative capabilities of LLMs. By incorporating the idea of HyDE, e attempt to mitigate the difference of text styles between input query and answer candidates, and improve the retrieval performance.

2.2 Two-Stage Approach in Ranking Problems

In ranking problems such as recommender systems, a common and effective strategy is to use a two-stage approach [1]. In the first stage, a retrieval model is employed to perform a broad search, identifying a preliminary set of relevant documents. Following this, a re-ranking model is used to refine the rankings and improve precision.

Retrieval models are proficient at conducting an initial coarse search, but their performance can be further enhanced by employing a re-ranking step. This subsequent re-ranking step evaluates the initially retrieved documents in more detail, leading to improved accuracy and relevance in the final results. Consequently, we can further enhance ranking performance of retrieval models.

3 Methodology

3.1 Task Formulation

The OAG-AQA task is to develop a model that retrieves the most relevant papers to answer questions in a variety of fields. In the AQA competition, a set of specialized questions and a corpus of papers in a variety of fields are given, and the papers that best match each specialized question must be found. In the OAG-AQA competition, the paper corpus and the question and answer pairs for test evaluation are released at the end of the competition. We refer to the periods before the final test release as phase 1, and the periods after the test data release as phase 2.

3.2 Method Overview

Here, we describe the proposed solution method. The solution is outlined as a two-stage prediction using LLMs and contrast learning. Figure 1 is a schematic diagram of the solution. Stage 1 is a vector search that extracts text candidates from the paper corpus. that correspond to the input query. This stage comprises two components.

- In the vector search, fine tuning of the embedding model is performed using contrastive learning, *i.e.*, SimCSE [3]. Supervised SimCSE is performed with queries, which represent questions in the training data, and passages, which correspond to answers to those queries, as positive example pairs. Contrastive learning acquires an embedding model that takes into account the relational context of the question and answer. Figure 2 shows an overview of fine tuning with SimCSE.
- The answer to the query is generated using the LLM in a zero shot manner and the generated result is added to the query. This is referred to as HyDE [2], which is known as a method to improve the search accuracy in RAG. HyDE alleviates one of the search challenges, *i.e.*, the low search accuracy issue caused by differences in query and passage formats. Figure 3 shows an overview of HyDE.

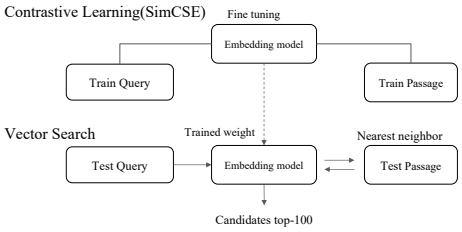

Figure 2: Fine tuning of the embedding model using SimCSE. The embedding model is trained using question and answer positive pairs in the training dataset.

In stage 2, the encoder-based language model performs binary classification and re-ranks candidates using predicted probability values. The final score is obtained by ensembling the three predictions. Figure 4 is a schematic of the ensembles.

3.3 Implementation Details

Please refer to our code[1] for more details. Here, we present a brief explanation of the implementation points.

In stage 1, we first perform two preprocessing methods for both queries and passages: the head tail method and prefix addition. In the head -tail method, since both query and passage may have important information at the beginning and end, only the first and last words are used for text where the word count exceeds the threshold(512 words). In prefix addition, prefixes indicating query, passage, and LLM generation are added to each text data. This provides a marker for the embedding model and each text. These preprocessing steps improve the public leaderboard score by a small amount.We process HyDE with two different prompts to enhance the query information:.

We input the preprocessed specialized questions into the LLM, which generates answers and summaries to the specialized questions. The prompt to the LLM instructs the LLM to output not only the answer and summary for the question, but also technical keywords, so that the technical terms of the paper can be easily found during later retrieval. We use "mlabonne/Marcoro14-7B-slerp" [2] for the LLM. This LLM was generated using "Model merge" [8], [9], and is adopted because it achieves high performance on the Open LLM Leaderboard [3].

In SimCSE fine tuning, the embedding model is trained using positive examples. The leaderboard score was worse when adding a hard negatives so we did not use any hard negatives. Finally, based

[1]https://github.com/NTT-DOCOMO-RD/kddcup2024-oag-challenge-ind-7th-aqa-7th-solution-nttdocomolabs/tree/main/AQA)
[2]https://huggingface.co/mlabonne/Marcoro14-7B-slerp
[3]https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard

on the above, we search for papers with high cosine-similarity to the query and select the top 100 candidates. For the embedding model, we use "gte-small" [5].

For stage 2 training, supervised candidate data are labeled with positive and negative examples for each query, and a passage pair is generated using the training data.Here, in creating candidate data for training, supervised candidate data are generated through cross-validation. In the training dataset, we include the paper corpus and the training data in the passage, so the number of positive examples in the supervised candidate data can be increased. On the other hand, when generating inference candidate data, only the paper corpus is used in the passage.

Notably, we use a corpus comprising only 68,673 papers of the differential update between phase 1 and phase 2 for the paper corpus used for the passage in generating the inference candidate data. Since there is a 395,812 paper corpus in phase 1 and a 466,387 paper corpus in phase 2, the 70,575 paper corpus of differential updates between phase 1 and phase 2 is only approximately 15% of the phase 2 paper corpus. However, this data selection is performed to contribute to the improvement of the public leaderboard ranking. In practical applications, however, reducing the corpus of paper should be avoided, as it may prevent the proper retrieval of papers. In stage 2, the input information to the encoder is the combined text of query and passage. Query and passage are combined after limiting them to the same number of words per query and passage, so that each amount of information from the query and passage can be fed into the encoder. The encoder model is "deberta-v3-large" [4]. Because it takes a long time for the encoder model to train using 100 candidates, the encoder is trained after downsampling the negative items. In this process, negative items are reduced at ratios of 1/25, 1/33, or 1/100. Inference is performed using the trained model with these three models trained with different amounts of negative samples, and the final prediction result is obtained by ensemble of the three prediction probabilities by simple arithmetic mean.

4 Evaluation

4.1 Experimental Setup

4.1.1 Dataset. The OAG-AQA competition dataset is generated by retrieving question submissions from the StackExchange and Zhihu websites, extracting the URLs of the papers mentioned in the answers, and matching them to the papers in the OAG. [7] The paper corpus contains 466,387 records, with paper id, paper title, and paper abstract. There are 8,757 records as training data, with the paper id as the technical question and correct answer data. There are 3,000 records for the test split that contain only technical questions.

4.1.2 Metrics. The OAG-AQA evaluation metric uses MAP@k, a metric based on mean average precision. It is an index that counts the mean average precision against the top-k candidates. In this case, k is set to 20, and the final score is calculated using MAP@20.

4.2 Results and Discussion

The public leaderboard scores for each solution method are given in Table 1. The results are evaluated online using MAP@20 on the public leaderboard. Here, we describe the experimental results for each of the methods given in Table 1. First, we perform a vector search of 20 candidates using the embedding model as a baseline for stage 1. We obtain a score of 0.120 for this baseline. We then boost the score to 0.132 and 0.133 by adding contrastive learning and HyDE to the baseline, respectively. Then, in stage 2, the score is re-ranked so that the papers which are correct among the candidates should be ranked higher, resulting in a score of 0.173. The number of candidates is increased to 50 and 100 so that more positive candidate items are included, and the score is increased to 0.175, and 0.183, respectively. Finally, by simply averaging the three predictions generated using the training model with different negative sampling ratios, we obtain 0.186 on the leaderboard, which achieved 6th place.

Table 1: Results of methods in leaderboard

Solution	LB score
Vector search (gte-small)	0.1205
Vector search with SimCSE	0.1326
Vector search with HyDE	0.1335
Vector search with SimCSE & HyDE	0.1502
Re-ranking top 20	0.1739
Re-ranking top 50	0.1754
Re-ranking top 100	0.1830
Ensemble (simple average)	0.1860

4.3 Future work

In the future, we will explore different ways to improve our proposal. For example, in stage 1, an embedding model with a parameter size of a few million orders is used, but it could be improved by using LLM with a few billion parameter size. Also, the LLM used in HyDE is a 7 billion model, but performance could be improved by using larger models such as 13 billion, 70 billion, or more. And the prompts used in HyDE could be improved by adding prompt engineering experiments. In stage 2, it may be effective to increase the number of candidate types, *e.g.*, graph-based candidate creation and rule-based keyword searches. Table 1 also shows that the public leaderboard score improves as the number of candidates becomes larger, so there is room to improve the public leaderboard score by further increasing the number of candidates. Nevertheless, we need to choose carefully the top-k value considering the trade-off between performance and computational cost. Also, Table 1 shows that the leaderbord scores vary widely depending on the downsampling rate. Thus, there may be a optimal downsampling rate.

5 Conclusion

In this paper, we introduced a two-stage model for the OAG-AQA competition that employs HyDE and contrastive learning. In stage 1, we performed query augmentation using HyDE. Subsequently, candidates were generated using an embedding model employing SimCSE. In stage 2, re-ranking was performed using the encoder model. As a result, the model won 6th place on the public leaderboard in the OAG-AQA competition.

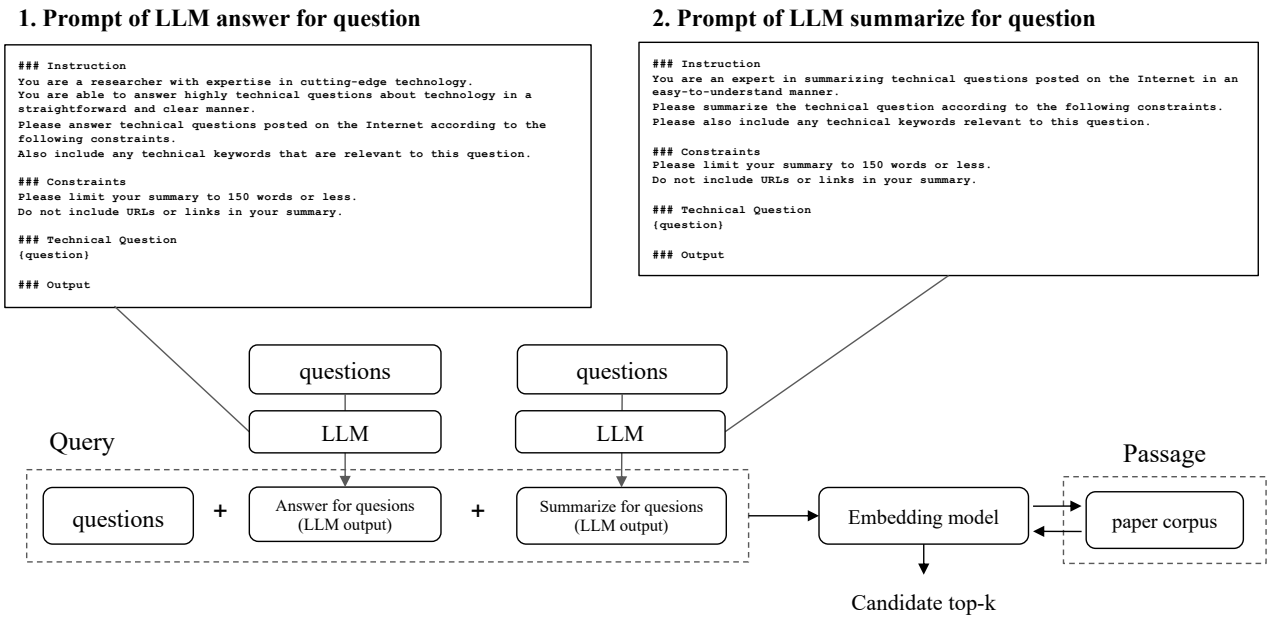

Figure 3: Query augmentation using HyDE. Two LLM-generated output texts are added to the query. These texts include the answer and summary of the query.

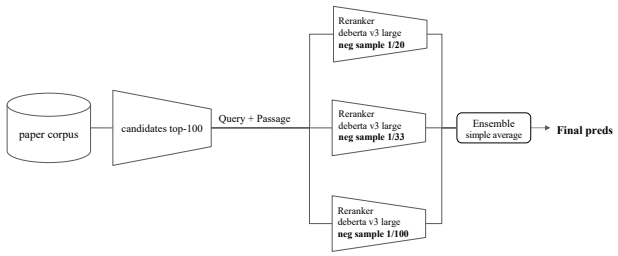

Figure 4: Ensemble of three re-ranker predictions. The three predictions are output from the three models trained with different negative downsampling rates.

References

[1] Chris Deotte, Kazuki onodera, Jean-Francois Puget, Benedikt Schifferer, and Gilberto Titericz. 2023. Winning Amazon KDD Cup'23. In *Amazon KDD Cup 2023 Workshop*. https://openreview.net/forum?id=J3wj55kK5t

[2] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *ACL*.

[3] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *EMNLP*.

[4] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION. In *ICLR*. https://openreview.net/forum?id=XPZIaotutsD

[5] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards General Text Embeddings with Multi-stage Contrastive Learning. *arXiv preprint arXiv:2308.03281* (2023).

[6] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[7] Weng Tam, Xiao Liu, Kaixuan Ji, Lilong Xue, Jiahua Liu, Tao Li, Yuxiao Dong, and Jie Tang. 2023. Parameter-Efficient Prompt Tuning Makes Generalized and Calibrated Neural Text Retrievers. In *EMNLP Findings*.

[8] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-Merging: Resolving Interference When Merging Models. In *NeurIPS*. https://openreview.net/forum?id=xtaX3WyCj1

[9] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *ICML*.

[10] Fanjin Zhang, Shijie Shi, Yifan Zhu, Bo Chen, Yukuo Cen, Jifan Yu, Yelin Chen, Lulu Wang, Qingfei Zhao, Yuqing Cheng, Tianyi Han, Yuwei An, Dan Zhang, Weng Lam Tam, Kun Cao, Yunhe Pang, Xinyu Guan, Huihui Yuan, Jian Song, Xiaoyan Li, Yuxiao Dong, and Jie Tang. 2024. OAG-Bench: A Human-Curated Benchmark for Academic Graph Mining. In *KDD*.