CIP at TREC Deep Learning Track 2023

Xiaoyang Chen[1,2], Ben He[1,2], Le Sun[2], and Yingfei Sun[1]

[1] University of Chinese Academy of Sciences, Beijing, China
[2] Chinese Information Processing Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China
chenxiaoyang19@mails.ucas.ac.cn, {benhe, yfsun}@ucas.ac.cn
sunle@iscas.ac.cn

Abstract. This study presents the strategies and experimental results employed by the CIP team in the Passage Ranking task of the 2023 TREC Deep Learning Track. In the full-ranking task, we incorporated sparse retrieval methods such as Unicoil [4] and DocT5Query [6], cross-attention mechanism (MonoT5 [8]), and the recent advancements in large language models (LLM) to achieve improved sorting effectiveness. Additionally, we utilized a multi-round iterative optimization strategy for deep ranking of selected candidate documents. The experimental data suggests that by harnessing the power of existing resources, our approach has yielded favorable results in this task, without necessitating any additional training.

1 Introduction

In the field of information retrieval, passage ranking is a crucial task that helps extract relevant information from a large number of text segments. This paper provides a comprehensive overview of the CIP team's strategies and experimental outcomes in the TREC DL 2023 Passage Ranking task.

Currently, the sparse retrieval method [4, 6], which recalculates term weights via pre-trained language models, has gained widespread use due to its combination of context-informed effects and the efficiency of inverted indexing. Meanwhile, although the ranker based on the cross-attention mechanism [3, 7, 8] is less efficient than sparse retrieval, it captures the relevance between queries and texts more meticulously. Hence, it is commonly used to rank a certain number of top-ranked documents after initial retrieval.

Recently, the emergence of ChatGPT [10] and GPT4 [9] has offered new possibilities for optimizing ranking effects. Existing methods [5, 12] demonstrate that zero-shot learning, which inputs a query and a certain number of texts ranked by BM25 into a large language model, and outputs text numbers in descending order of relevance to the query, can effectively improve the ranking effect of BM25. Experiments show that the results ranked by ChatGPT are still inferior to the commonly used ranker tuned for specific tasks based on the cross-attention mechanism, while GPT4 can directly achieve SOTA results.

For most IR researchers, the high price of GPT4 hinders relevant research. Conversely, the price of ChatGPT falls within an acceptable range for many.

Therefore, in this Passage Ranking task, the primary objectives of the CIP team are to investigate the following questions: Firstly, could superior initial ranking outcomes enhance the performance of ChatGPT beyond the results currently reported in the field of study? Secondly, through iterative processes, is it possible for ChatGPT to further refine its ranking performance, building upon its own initial results?

In line with the full-ranking task setup, our strategy mainly involves three key parts:

- In the retrieval stage, we utilized Unicoil [4] and DocT5query [6], both based on sparse indexing, to recall 1000 passages for each query.
- In the ranking stage, we employed MonoT5-3B [8] to score the documents recalled by Unicoil and DocT5query, attaining the top 100 passages.
- In the re-ranking stage, we utilized large language models to perform zero-shot group ranking on the top passages. We integrated the Pseudo Relevance Feedback (PRF) mechanism into the prompts while conducting multiple rounds of iteration under various group sizes and coverage settings to further optimize the ranking effect. To verify the best effect that this method could achieve, we used GPT4 results as a reference for comparison.

2 Methodology

In this section, we introduce the detailed methodology employed by the CIP team to accomplish the Passage Ranking task. Our approach involves uniform processing at the retrieval and ranking stages. Initially, we make use of Unicoil [4] and DocT5Query [6] for query retrieval across the entire dataset. Following this, the top 1000 documents recalled by each method are rescored using the MonoT5-3B model [8]. Furthermore, during the reranking stage, we adopt the methods proposed by [5] and [12], to group and sort the top-ranked documents. We also incorporate the Pseudo Relevance Feedback (PRF), fusion, and multi-round iteration concepts under different parameters to achieve our final results.

2.1 Retrieval Stage

At this stage, we utilize two modified sparse retrieval methods to recall the top 1000 texts for the queries across the entire dataset. The first method, **Unicoil** [4], is a sparse retrieval model that employs a pre-trained language model to assign weights to the text and query tokens. Unicoil works on a dual-encoder structure and uses exact vocabulary matching to compute the similarity between the query and the passage, as opposed to matching in the latent semantic space. Hence, the vector representation generated by Unicoil is fundamentally based on the vocabulary space.

The other method, **DocT5Query** [6], predicts queries potentially related to the text by training the T5 model. The generated queries are added to the

original text as supplementary information. DocT5Query builds a sparse index on the expanded dataset and retrieves the query using the BM25 algorithm. Although Unicoil also leverages the query prediction feature of DocT5Query to expand the document, we believe the different weight calculation methods make the retrieval results from both methods complementary.

Note that at this stage, we use the implementation and index structure of pyserini [3] to obtain the top 1000 candidate results from both methods.

2.2 Ranking Stage

Following the retrieval stage, we obtain the top 1000 passages via the two aforementioned methods. We then proceed to deduplicate the text ids that were recalled by both methods. The subsequent step involves employing MonoT5-3B [8], which is based on the cross-attention mechanism, to re-score the query-text pairs for relevance.

MonoT5 is designed on a sequence-to-sequence task. Its input is formatted as follows:

$$Query : [Q]\ Document : [D]\ Relevant : \tag{1}$$

The model generates outputs by applying the softmax function to the logits of 'true' and 'false' tokens. This allows the model to calculate the probability $Pr(relevant = 1|q, d)$, which is the probability assigned to the 'true' token, and is interpreted as the relevance score of each query-document pair.

MonoT5-3B has been widely utilized as a baseline model in previous research due to its commendable performance. This step provides a refined ranking result that measures the relationship between queries and texts with higher precision.

Note that the checkpoint file [4] we used for MonoT5-3B was trained on the MSMARCO v1 version, and we directly applied it to the ranking task on MS-MARCO v2 version without any further adaptation.

2.3 Reranking Stage

In the reranking stage, we employ the zero-shot learning method based on large language models to rank the top 50 passages obtained from MonoT5-3B in the previous stage. At this stage, we primarily draw reference from the Instructional Permutation Generation methods mentioned in [5] and [12]. Simultaneously, we conduct multiple rounds of iteration and fusion under different window sizes and step sizes, further optimizing the ranking effect.

Further, we also introduce the Pseudo Relevance Feedback (PRF) concept, providing more information for the model to execute the ranking task. The large language model we use is ChatGPT [10], which is within our acceptable price range and also provides relatively good ranking results. To verify the best effect that this method can achieve, we use GPT4 [9] results for comparison. The

[3] https://github.com/castorini/pyserini
[4] https://huggingface.co/castorini/monot5-3b-msmarco-10k

Instructional Permutation Generation method is an innovative approach that leverages the robust text understanding and reasoning capabilities of large language models such as ChatGPT and GPT4. This method aims to generate a permutation of passages based on their relevance to a given query. Each passage is assigned a unique identifier and no intermediate relevance scores are produced. These passages are arranged in descending order of relevance to the query using their identifiers, thus producing a ranking list such as [2] > [3] > [1] > *etc.*

In our implementation, we construct the following instruction input into the model to obtain the ranking list:

```
Rank the w passages based on their relevance to the search query.
The passages will be listed in descending order using identifiers,
and the most relevant passages should be listed first. The output
format should be [] > [] > etc, just output the identifiers of the
candidates, not the text.

### Query:
How does the process of digestion and metabolism of carbohydrates
start

### Candidates:
[1] ...
[2] ...
...
[w] ...

### Response:
```

Due to the limited number of input tokens allowed by these models, the Instructional Permutation Generation method can only rank a finite number of n passages. To solve this problem, sliding window strategies have been used in [5, 11, 12] to enable the model to rank any number of passages. Here, two hyperparameters, namely window size (w) and step size (s), are defined. If the goal is to rank the top M texts, the model arranges passages from the (M-w)-th to the M-th position and slides the window using s as the step size. Assuming the reranker can always correctly rank the passages within the window, then after using the sliding window process K times, the top K×(w-s) passages in the final ranking are equivalent to directly ranking all the passages.

Pseudo Relevance Feedback (PRF) is a classic method used in Information Retrieval (IR) to enhance the performance of retrieval systems. The basic assumption of PRF is that documents retrieved at the top of the initial retrieval are likely to be relevant and can provide more context or information to refine the query and improve retrieval results. In our reranking stage, we introduce the PRF concept to provide more information for the model to execute the ranking task. We modify the instruction to include the top-p passages as pseudo-relevant feedback information:

```
Rank the w passages based on their relevance to the search query.
```

```
The passages will be listed in descending order using identifiers,
and the most relevant passages should be listed first. The output
format should be [] > [] > etc, just output the identifiers of the
candidates not the text.

Query:
How does the process of digestion and metabolism of carbohydrates
start

Candidates:
[1] ...
[2] ...
...
[w] ...

Pseudo-Relevant Feedback:
[prf 1] ...
[prf 2] ...
...
[prf p] ...

Response:
```

Although the Instructional Permutation Generation method provides a way to rank a certain number of texts under specific parameters, we believe that the length of different input texts and different step sizes will also affect the ranking effect. Intuitively, the smaller the window size, the higher the accuracy of the ranking, and the pairwise-based research of [11] indirectly confirms this point. Therefore, we propose to rerank the top-ranked passages in **multiple rounds and fuse** the results of multiple rounds.

In our submitted results, we iterated with ChatGPT for 4 rounds: in the first round, M=50, w=20, s=10, p=3, K=1 (cip_run_6); in the second round, M=40, w=20, s=10, p=0, K=1 (cip_run_4); in the third round, M=30, w=10, s=5, p=0, K=2 (cip_run_7); in the fourth round, M=30, w=6, s=3, p=0, K=4 (cip_run_5). We used the queries from DL21 [1] and DL22 [2] as our validation set, calculating the optimal weights for integrating the four outcomes to obtain the best results (cip_run_3). The optimal weights for the two query sets were averaged, and then applied to the sum of the four results from this year, in accordance with the calculated weights, to yield our submission.

To verify whether GPT-4 can further improve performance on top of the multi-turn iteration of ChatGPT, we conducted an experiment with the following settings: M=20, w=20, s=10, p=0, K=1 (cip_run_1). We employed GPT-4 to re-rank the fused results obtained earlier and submitted it as one of the experimental outcomes. As GPT-4 does not directly generate relevance scores, we also reprocessed the scores from MonoT5, filtering out duplicates, to obtain the final relevance scores (cip_run_2).

3 Results

	Initial	DL21		DL22		DL23	
		NDCG@10	RR	NDCG@10	RR	NDCG@10	RR
[*a*] UniCOIL [4]	-	0.6160	0.7311	0.4609	0.5831	-	-
[*b*] DocT5Query [6]	-	0.4816	0.6848	0.3599	0.4221	-	-
[*c*] MonoT5-3B [8]	[*a*], [*b*]	0.7041	0.8092	0.6183	0.7581	-	-
[*d*] Ours (ChatGPT)	[*c*]	0.7595	0.8761	0.6861	0.8294	0.6185	0.7724
[*e*] Ours (GPT4)	[*d*]	**0.7716**	**0.8921**	**0.7072**	**0.8646**	**0.6558**	**0.8320**

Table 1: Results on DL21, DL22, and DL23.

	DL21		DL22		DL23	
	NDCG@10	MAP	NDCG@10	MAP	NDCG@10	MAP
MonoT5-3B [8]	0.7041	0.3470	0.6183	0.2250	-	-
M=50, w=20, ChatGPT	0.7274	0.3652	0.6807	0.2412	0.6078	0.2677
M=40, w=20, ChatGPT	0.7445	0.3882	0.6786	0.2429	0.6137	0.2721
M=30, w=10, ChatGPT	0.7588	0.3950	0.6796	0.2419	0.6075	0.2676
M=30, w=6, ChatGPT	0.7667	0.4008	0.6650	0.2391	0.6074	0.2656
F_4	0.7595	0.3974	0.6861	0.2455	0.6185	0.2714
M=20, w=20, GPT4	0.7716	**0.4044**	0.7072	0.2508	**0.6558**	**0.2862**
(score reassigned)	**0.7722**	0.4043	**0.7083**	**0.2512**	**0.6558**	**0.2862**

Table 2: Results of different iterations. Apart from F_4, which represents the fused result of the previous four iterations, all other iterations use the result of the preceding row as the initial ranking.

The results at each stage of this study are presented in Table 1, where "Ours (ChatGPT)" represents the weighted results. Despite MonoT5-3B not being trained on the current dataset, it still shows improved performance based on UniCOIL and DocT5Query. By using ChatGPT to re-rank the results obtained from MonoT5, significant improvements were achieved on DL21 and DL22, with a considerable increase of more than 5 points in NDCG@10. This suggests that ChatGPT is capable of further optimizing the results when the initial ranking is strong. However, since the MonoT5-3B model used for experiments was not trained in a supervised manner, there is still room for improvement in the initial ranking performance, and the impact of ChatGPT's optimization on better initial rankings requires further investigation. GPT4 was still able to further improve results after multiple iterations of ChatGPT, which is particularly evident

in the DL23 test data, with an increase of over 6% in NDCG@10, demonstrating GPT4's superior ranking ability.

Table 2 shows the results of iterative implementation of the Instructional Permutation Generation method with different hyperparameters. On DL21 data, as the iteration process continues, the ranking effect gradually improves, but this trend is not reflected on DL22 and DL23. The fuse method can maintain stable good results in ranking, resisting performance fluctuations in different rounds.

4 Conclusion

This paper introduces the three-stage method employed by the CIP team in the TREC Deep Learning Track 2023 Passage Ranking task. In the first stage, we use UniCOIL and DocT5Query to perform retrieval queries across the entire corpus. In the second stage, we use MonoT-3B to re-score the documents retrieved in the previous step. In the third stage, we make use of ChatGPT and GPT4 to re-rank the top documents under various settings. Experimental results indicate that under conditions of strong initial ranking, ChatGPT is able to further enhance ranking performance. However, for results that have already been ranked by ChatGPT, additional iterations may have limited impact.

References

1. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Lin, J.: Overview of the TREC 2021 deep learning track. In: Soboroff, I., Ellis, A. (eds.) Proceedings of the Thirtieth Text REtrieval Conference, TREC 2021, online, November 15-19, 2021. NIST Special Publication, vol. 500-335. National Institute of Standards and Technology (NIST) (2021), https://trec.nist.gov/pubs/trec30/papers/Overview-DL.pdf
2. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Lin, J., Voorhees, E.M., Soboroff, I.: Overview of the TREC 2022 deep learning track. In: Soboroff, I., Ellis, A. (eds.) Proceedings of the Thirty-First Text REtrieval Conference, TREC 2022, online, November 15-19, 2022. NIST Special Publication, vol. 500-338. National Institute of Standards and Technology (NIST) (2022), https://trec.nist.gov/pubs/trec31/papers/Overview_deep.pdf
3. Li, C., Yates, A., MacAvaney, S., He, B., Sun, Y.: Parade: Passage representation aggregation for document reranking. arXiv preprint arXiv:2008.09093 (2020)
4. Ma, X., Pradeep, R., Nogueira, R., Lin, J.: Document expansion baselines and learned sparse lexical representations for ms marco v1 and v2. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 3187–3197. SIGIR '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3477495.3531749, https://doi.org/10.1145/3477495.3531749
5. Ma, X., Zhang, X., Pradeep, R., Lin, J.: Zero-shot listwise document reranking with a large language model. CoRR **abs/2305.02156** (2023). https://doi.org/10.48550/arXiv.2305.02156, https://doi.org/10.48550/arXiv.2305.02156
6. Nogueira, R., Lin, J., Epistemic, A.: From doc2query to doctttttquery. Online preprint **6**, 2 (2019)

7. Nogueira, R.F., Cho, K.: Passage re-ranking with BERT. CoRR **abs/1901.04085** (2019), http://arxiv.org/abs/1901.04085
8. Nogueira, R.F., Jiang, Z., Pradeep, R., Lin, J.: Document ranking with a pretrained sequence-to-sequence model. In: Cohn, T., He, Y., Liu, Y. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020. Findings of ACL, vol. EMNLP 2020, pp. 708–718. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.findings-emnlp.63, https://doi.org/10.18653/v1/2020.findings-emnlp.63
9. OpenAI: GPT-4 technical report. CoRR **abs/2303.08774** (2023). https://doi.org/10.48550/arXiv.2303.08774, https://doi.org/10.48550/arXiv.2303.08774
10. OpenAI: Introducing chatgpt (2023), https://openai.com/blog/chatgpt
11. Qin, Z., Jagerman, R., Hui, K., Zhuang, H., Wu, J., Shen, J., Liu, T., Liu, J., Metzler, D., Wang, X., Bendersky, M.: Large language models are effective text rankers with pairwise ranking prompting. CoRR **abs/2306.17563** (2023). https://doi.org/10.48550/arXiv.2306.17563, https://doi.org/10.48550/arXiv.2306.17563
12. Sun, W., Yan, L., Ma, X., Ren, P., Yin, D., Ren, Z.: Is chatgpt good at search? investigating large language models as re-ranking agent. CoRR **abs/2304.09542** (2023). https://doi.org/10.48550/arXiv.2304.09542, https://doi.org/10.48550/arXiv.2304.09542