Application of Multi-way Recall Fusion Reranking Based on Tensor and ColBERT in RAG

Wenmin Wang[a], Junpeng Ma[b], Peilin Zhang[c], Zhuolun Hu[d], Qi Jiang[e], Yafei Liu
China Mobile Online Service Co., Ltd., Zhengzhou, 450000, Henan, China
[a]xiaofengcanyuexp@163.com, [b]majunpeng@cmos.chinamobile.com, [c]zhangpeilin@cmos.chinamobile.com,
[d]huzhuaolun@cmos.chinamobile.com, [e]jiangqi@cmos.chinamobile.com

2024 IEEE 7th International Conference on Information Systems and Computer Aided Education (ICISCAE) | 979-8-3503-5076-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICISCAE62304.2024.10761558

***Abstract*—In recent years, Retrieval-Augmented Generation (RAG), as a framework to assist large model (LLM) retrieval enhancement and generation capabilities, has been widely used in the field of natural language processing. However, the recall algorithm, recall rate, precision and recommendation algorithm (Reciprocal Ranker) of different RAG frameworks largely determine the effectiveness of the final retrieval results of large models. This paper proposes a multi-way recall ranking method based on Tensor + ColBERT (Contextualized Late Interaction over BERT) to improve the retrieval part of the RAG system. This method uses the delayed interaction mechanism of ColBERT to improve the diversity and relevance of retrieval results through a multi-path retrieval strategy on the one hand, and uses Tensor to improve the multi-vector MaxSim similarity calculation of ColBERT on the other hand, achieving the efficiency of dual encoders and accurate fusion reranking. Experimental results show that the improved RAG system can significantly improve the recall and precision of retrieval in complex query and long document scenarios, while taking into account the balance of computing resources and processing speed.**

Keywords—Tensor, ColBERT, RAG, Multi-way recall, Fusion ReRanker

I. Introduction

With the rapid development of the field of natural language processing (NLP), the Retrieval-Augmented Generation (RAG) [1] framework has been widely used in tasks such as large language model (LLM) knowledge question answering and dialogue systems. RAG improves the accuracy of answers while increasing the density and real-time nature of answer information by retrieving external real-time knowledge bases and generating large language models [2]. However, the overall performance of RAG depends largely on the recall, efficiency, and accuracy of its retrieval module, which is particularly prominent and challenging when processing complex queries and long documents [3].

Most of the common RAG retrieval modules rely on vector retrieval. In order to improve retrieval efficiency, the complete vector (Dense Vector) is usually compressed into a sparse vector (Sparse Vector). Although sparse vectors remove a large number of redundant words through pre-trained models and introduce words that help query expansion, which shows advantages in general query tasks [4], in actual applications, there are still a large number of keywords in user questions that are not in the pre-trained model for generating sparse vectors, such as specific machine models, manuals, or professional terms. Regardless of whether all keywords are expressed in massive dimensions or in multiple languages, the problem of information loss still exists. In addition, phrase query requirements that are indispensable in many business scenarios must be implemented using full-text search. IBM's recent research [5] compared different recall combinations, including BM25 [6], dense vectors, BM25 + dense vectors, dense vectors + sparse vectors, and a combination of the three, and finally concluded that three-way recall is the most suitable strategy for RAG. The advantage of three-way recall is that it combines the semantic representation ability of dense vectors, the precise recall advantage of sparse vectors in specific scenarios, and the robustness of full-text search in diverse scenarios.

After implementing three-way recall, how to effectively rerank the retrieval results (ReRanker) while ensuring efficiency and accuracy has become the key to improving RAG performance [7].

II. Design and Implementation

A. Reciprocal Rank Fusion (RRF) Algorithm

The Reciprocal Rank Fusion (RRF) algorithm [8] provides a simple and effective ranking fusion method that can integrate the ranking results of multiple independent recall strategies into a unified result set. The advantage of this algorithm is that it does not require adjusting the correlation between different relevance indicators. RRF assigns a reciprocal ranking score to each document in the result list of each recall path. This score is calculated based on the position of the document in the list, as shown in formula (1):

$$RRF\left(d \in D\right) = \sum_{s \in S} \frac{1}{k + s\left(d\right)} \tag{1}$$

Among them, s(d) is the rank of document d in recall path si, and k is a hyperparameter.

Although the RRF algorithm is highly robust, it assigns scores based entirely on the ranking of recall results, which may ignore the similarity information between the document and the query in the original recall. When faced with specific problems, such as "What should I do if the device model XX-300 does not work", it is necessary to emphasize keywords to increase their score weight, but this method may not be sufficient to meet the needs.

B. Reranking Based on ColBERT

ColBERT (Contextualized Late Interaction over BERT) [9] introduces a delayed interaction similarity function to

calculate the similarity (MaxSim) between queries and documents. Compared with RRF, ColBERT can more accurately capture the complex interactive relationship between queries and documents, thereby providing more accurate search ranking results. MaxSim is calculated as follows: the vector of each query token is similar to the vectors of all document tokens, and the maximum score of each query token, that is, the maximum cosine similarity, is tracked. The total score of the query and document is the sum of these maximum cosine scores. For example, for a query with 32 token vectors (maximum query length is 32) and a document with 128 tokens, 32*128 similarity operations need to be performed, as shown in Figure 1.

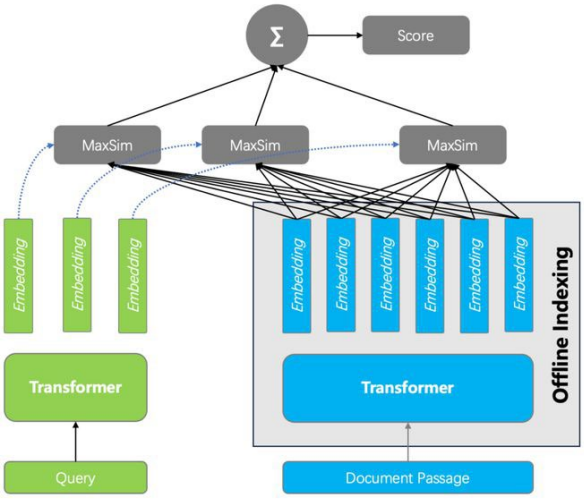

Fig. 1. ColBERT model principle.

Although the ColBERT model surpasses the Cross Encoder in efficiency by adopting a delayed interaction similarity function, its computational cost is still higher than that of traditional vector search. The computational cost of MaxSim is proportional to the number of tokens in the query and document, that is, M times N (M represents the number of tokens in the query and N represents the number of tokens in the document), which means that the computational cost will increase significantly when processing large-scale data. To solve this problem, the developers of ColBERT launched ColBERT v2 in 2021 [10]. ColBERT v2 improves the quality of generated embeddings by combining Cross Encoder and model distillation technology. In addition, by using compression technology to quantize document vectors, ColBERT v2 significantly improves the computational performance of MaxSim. The RAGatouille project based on ColBERT v2 has become a high-quality RAG question-answering solution, but integrating ColBERT v2 as an algorithm library into an enterprise-level RAG system still faces certain challenges.

In addition, as a pre-trained model, ColBERT's training data mainly comes from search engine queries and return results. The scale of these text data is relatively small, and the number of tokens in queries is usually limited to 32, and the number of tokens in documents is limited to 128. When ColBERT is applied to real data, the part exceeding these length limits will be truncated, which may have an adverse effect on the retrieval effect of long documents.

C. Fusion Reranking Based on Tensor and ColBERT

In view of the respective advantages and disadvantages of RRF and ColBERT, this paper proposes a fusion re-ranking strategy based on tensors and ColBERT to achieve better retrieval performance. We provides a built-in Tensor data type and an embedded end-to-end ColBERT solution, which can directly store multiple vectors output by the ColBERT model as Tensors, thereby achieving MaxSim scoring through similarity calculations between Tensors. In addition, for documents that exceed the Token limit, they will be divided into multiple paragraphs, encoded to generate Tensors, and stored in the Tensor Array, which will be saved in the same row as the original document. When calculating MaxSim, the query is calculated separately for these paragraphs, and the maximum value is taken as the score of the entire document, as shown in Figure 2.

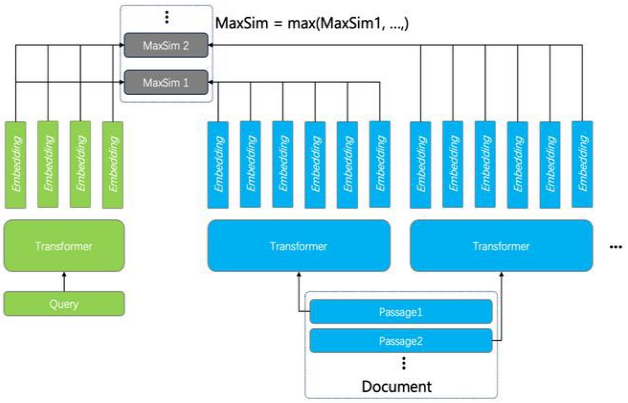

Fig. 2. Segmented calculation and MaxSim integration process.

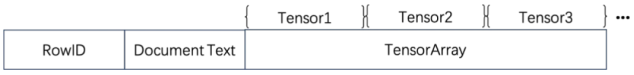

Fig. 3. Dealing with token limitations in long documents: piecewise tensor representation and storage.

The storage structure of a long document after segmentation is shown in Figure 3.

In addition, to address the challenge of large MaxSim computational complexity, We provides two optimization solutions: one is binary quantization, which can reduce the space of the original Tensor to only 1/32 of the original size, but does not change the relative ranking results of the MaxSim calculation. This solution is mainly used for Reranker, because it is necessary to retrieve the corresponding Tensor based on the sorting result of the previous stage. The other is Tensor Index. We uses EMVB [11] technology to implement Tensor Index. EMVB can be regarded as an improved version of ColBERT v2. It mainly accelerates the implementation through quantization and pre-filtering technology, as well as the introduction of SIMD instructions on key operations. Tensor index technology mainly serves Retriever, not Reranker.

III. Experiment and Result Analysis

A. Experimental Setup

1) Dataset: This paper selects the Multi Long Document Retrieval (MLDR) dataset for evaluation. This dataset is one of the benchmark sets used by MTEB[12] to evaluate the quality of the Embedding model and contains 200,000 long text data.

2) Evaluation indicators: This paper uses normalized discounted cumulative gain (nDCG@k) as the main evaluation indicator [13]. Other parameters: When using RRF Reranker, the top N returned by the coarse screening is 1000, the total number of queries is 800, and the average length of each query is about 10 tokens.

3) Data processing: 200,000 MLDR data are used to generate dense vectors and sparse vectors using BGE-M3[14], and Jina-ColBERT is used to generate Tensors, which are then inserted into the database. The database contains original text, vectors, sparse vectors, and Tensors, and the corresponding full-text index, vector index, and sparse vector index are constructed respectively.

4) Evaluation method: The evaluation includes all recall combinations, including single-way recall, dual-way recall, and triple-way recall, as shown in Figure 4. In these recall combinations, the effectiveness of ColBERT as a reranker is further evaluated, and its role in improving the ranking quality of Top-k results under different recall combinations is analyzed.

Fig. 4. Evaluation recall combination.

B. Experimental Results

1) Comparison of retrieval effects under different recall strategies: Figure 5 shows the evaluation results of different recall strategies on the MLDR dataset.

As shown in Figure 5, the combination of BM25 full-text search and vector search significantly improves the retrieval effect compared to a single vector search. When a three-way combination of full-text search, vector search, and sparse vector search is further adopted, the quality of the retrieval results is further improved. On this basis, the introduction of ColBERT as a Reranker further improves the quality of the retrieval results. Compared with the method that relies on an external encoder for sorting optimization, this multi-way recall reranking strategy that combines tensors (Tensor) + ColBERT as Reranker can efficiently complete the sorting within the database. The advantage of internal sorting is that it allows us to expand the range of retrieval results (for example, the top 1,000 results) and finely sort them without significantly increasing the computational overhead.

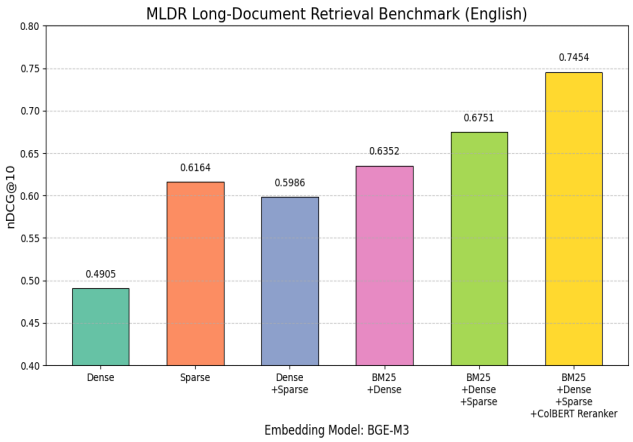

Fig. 5. nDCG@10 performance of different retrieval models in the MLDR long document retrieval benchmark.

2) Analysis of the effect of ColBERT as a Reranker: Figure 6 shows the improvement of different recall methods for Top 100 after adding ColBERT Reranker.

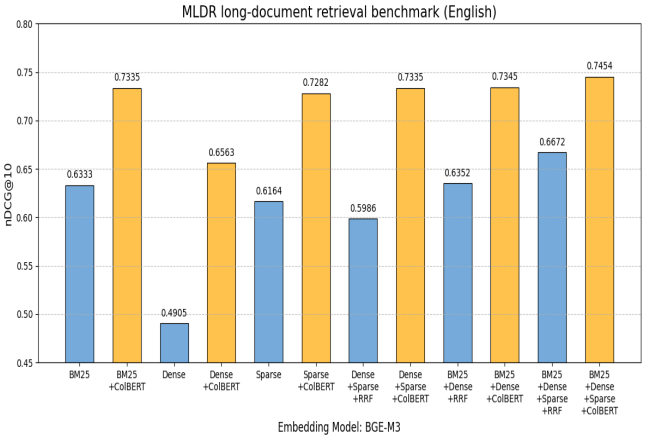

Fig. 6. Analysis of the effect of ColBERT as a Reranker.

As can be seen from Figure 6, regardless of the recall strategy, the retrieval effect has been significantly improved after the introduction of ColBERT Reranker. As a delayed interaction model, ColBERT can not only provide ranking quality comparable to the top models on the MTEB Reranker ranking list, but also far exceeds these models in terms of performance, with an efficiency improvement of 100 times. This advantage enables ColBERT to perform efficient reranking in a larger range.

3) Analysis of the performance of ColBERT as a Ranker and Reranker

Figure 7 shows the comparison of the effects of ColBERT as a Ranker and a Reranker. When used as a Ranker, nDCG@10 using the EMVB index is 0.7223, which is lower than 0.7335 when BM25 and ColBERT are combined as Rerankers. Even in the case of brute force search, nDCG@10 is only slightly improved to 0.7411, and the gain is not obvious. It is worth noting that the computing resource requirements and time cost of ColBERT as a Ranker are much higher than those of the Reranker mode.

Especially when processing large-scale document data, the Ranker mode significantly reduces the query speed due to processing larger-scale vector data. Therefore, although in theory ColBERT as a Ranker may bring a small improvement in accuracy, in practical applications, the Reranker mode has more advantages in balancing effect and efficiency, and is more suitable for improving the accuracy of retrieval results.

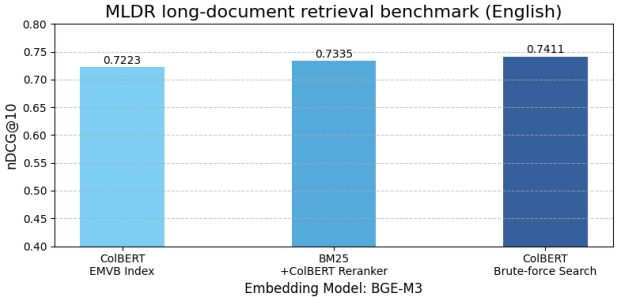

Fig. 7. Comparison of the performance of ColBERT as a Ranker and a Reranker.

C. Results Analysis

Based on the above experimental results, we can draw the following conclusions: The current best RAG retrieval strategy is to use a three-way hybrid search (full-text search, vector search, and sparse vector search) combined with Tensor + ColBERT as a Reranker. This strategy performs well in all evaluation indicators and significantly improves the retrieval effect.

First, Figure 5 shows the retrieval effect under different recall strategies. Compared with a single path, the three-way hybrid search strategy significantly improves the retrieval quality, especially when processing complex queries. Second, Figure 6 proves that after the introduction of ColBERT Reranker, the effects of various recall strategies have been significantly enhanced. The results of Figure 7 further emphasize the superiority of ColBERT as a Reranker. Although ColBERT as a Ranker (using EMVB index or brute force search) can also improve accuracy in some cases, its resource consumption and time cost are much higher than the Reranker mode. Therefore, in practical applications, the Reranker mode has more advantages in balancing effect and efficiency, especially in the scenario of large-scale document processing, where it performs better.

IV. Conclusion

This study proposed and verified a retrieval strategy that combines three-way hybrid search (full-text search, vector search, sparse vector search) and tensor + ColBERT Reranker, which significantly improved the retrieval effect of the RAG system. Through systematic experimental analysis, the study found that this strategy performed well in various evaluation indicators, especially in complex queries and long document scenarios, and can effectively improve the quality of retrieval results. In addition, although ColBERT has improved in accuracy as a Ranker, its high computing resources and time cost limit the breadth of its practical application. Therefore, we recommend using the Reranker mode in practical applications to better balance retrieval effect and computational efficiency. This study provides a new perspective for the optimization of the RAG system and provides a reference for future research and applications.

References

[1] P. Lewis et al, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Proc. NeurIPS*, *Vancouver*, 2020, pp. 9459–9474.

[2] H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu, "A survey on retrieval-augmented text generation," arXiv preprint arXiv:2202.01110, 2022.

[3] Y. Gao et al., "Retrieval-augmented generation for large language models: A survey," arXiv preprint arXiv:2312.10997, 2023.

[4] A. Agarwalla et al., "Enabling High-Sparsity Foundational Llama Models with Efficient Pretraining and Deployment," arXiv preprint arXiv:2405.03594, 2024.

[5] K. Sawarkar, A. Mangal, and S. R. Solanki, "Blended RAG: Improving RAG (Retriever-Augmented Generation) Accuracy with Semantic Search and Hybrid Query-Based Retrievers," arXiv preprint arXiv:2404.07220, 2024.

[6] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," Found. Trends Inf. Ret., vol. 3, no. 4, pp. 333–389, December 2009.

[7] Y. Zhu et al. "Large language models for information retrieval: A survey," arXiv preprint arXiv:2308.07107, 2023.

[8] G. V. Cormack, C. L. A. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," *Proc. 32nd Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, Boston, MA, 2009, pp. 758–759.

[9] O. Khattab and M. Zaharia, "Colbert: Efficient and effective passage search via contextualized late interaction over bert," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, New York, 2020, pp. 39–48.

[10] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, and M. Zaharia, "Colbertv2: Effective and efficient retrieval via lightweight late interaction," in *Proc. 2022 Conf. North Am. Chap. Assoc. Comput. Linguist.*, Seattle, 2022, pp. 3715–3734.

[11] F. M. Nardini, C. Rulli, and R. Venturini, "Efficient Multi-vector Dense Retrieval with Bit Vectors," in *Advances in Information Retrieval*. vol. 14609, N. Goharian et al. Cham: Springer, 2024, pp. 3–17.

[12] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive text embedding benchmark," in *Proc. 17th Conf. Eur. Chap. Assoc. Comput. Linguist.*, Dubrovnik, 2023, pp. 2014–2037.

[13] D. Valcarce, A. Bellogín, J. Parapar, and P. Castells, "Assessing ranking metrics in top-N recommendation," Inf. Retrieval J., vol. 23, pp. 411–448, June 2020.

[14] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, "Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," in *Findings of the Association for Computational Linguistics ACL 2024*, Bangk, 2024, pp. 2318–2335.