IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier XX.XXXX/ACCESS.20XX.DOI

DeepBERT: LLM-Driven Fine-Grained Late Interaction for Multiple-Vector Neural Retrieval

ANRAN ZHANG [1,2] (Student Member, IEEE),
[1]Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, Shannxi 710049 China
[2]Suzhou Automotive Research Institute, Tsinghua University, Suzhou, Jiangsu 215200 China

Corresponding author: Anran Zhang (e-mail: 179404529@stu.xjtu.edu.cn).

*Work initially done during intern in Tsinghua University, then mainly revised in Xi'an Jiaotong University

ABSTRACT The multi-vector retrieval represented by ColBERT has made a significant impact in the field of neural retrieval, but it has also brought two major issues. Firstly, the storage overhead of multi-vectors is much larger than that of single-vector (dense retrieval) models. Secondly, the Top-1 mechanism in the later interaction stage limits the model's expressive power. To address these issues, this paper proposes the DeepBERT model. The main innovations of this model are as follows: Firstly, we designed a generation module based on DeepSeek LLM, which can generate importance weights for each token without the need for retraining or fine-tuning the model. Secondly, we devised a very simple pruning strategy, which retains only a certain proportion of tokens according to the aforementioned importance weights. Finally, we improved an adaptive method that can adjust the retrieval strategy on a small number of labeled samples to adapt to specific tasks without additional pre-training or fine-tuning. We tested the retrieval performance of the model on several mainstream datasets. Both the full version and the moderately pruned DeepBERT outperformed ColBERT, and even the aggressively pruned (10%) DeepBERT performed on par with the traditional ColBERT. Additionally, we tested the end-to-end performance of DeepBERT, and its performance based on DiskANN implementation also surpassed the traditional ColBERT and outperformed the latest XTR model.

INDEX TERMS Neural Retrieval, LLM, NLP.

I. INTRODUCTION

Information retrieval has always been a crucial field in natural language processing and database systems. Traditional information retrieval methods primarily rely on text matching, with the most classic being the BM25 model [2] based on TF-IDF [1]. With the advent of neural networks and deep learning, language models such as BERT [3] and GPT [4] have fundamentally transformed the state of the art, making neural retrieval the mainstream approach in information retrieval.

In the current landscape of neural retrieval, two types of pre-trained language models (PLMs) have emerged. The first is representation-based models (dense retrieval), where BERT encodes queries Q and passages P (also referred to as documents, but hereafter uniformly termed as passages) into single fixed-dimensional vectors, and the similarity between Q, P is calculated via the inner product of these vectors. A notable example of this approach is DistilBERT [5]. The second type is interaction-based models (multi-vector retrieval), where BERT generates a vector for each token in both the query and the passage, and the final similarity is determined through a late interaction mechanism such as MaxSim. This method was pioneered by the ColBERT [6].

Since the introduction of ColBERT, research and optimization of ColBERT and its late interaction mechanisms have remained a significant focus. Some studies in this area have particularly caught our attention. In terms of optimizing pre-trained models, ColBERTv2 [7] employed hard negative training and small-model distillation to further enhance ColBERT's performance in encoding text. Jina-ColBERT [8] explored the possibility of multilingual ColBERT by altering training strategies. More recently, BGE-M3, proposed by BAAI (Beijing Academy of Artificial Intelligence), achieved multilingual, multifunctional, and multi-granularity retrieval through a novel self-distillation framework.

In terms of system-level optimization, ColBERTv2 [7]

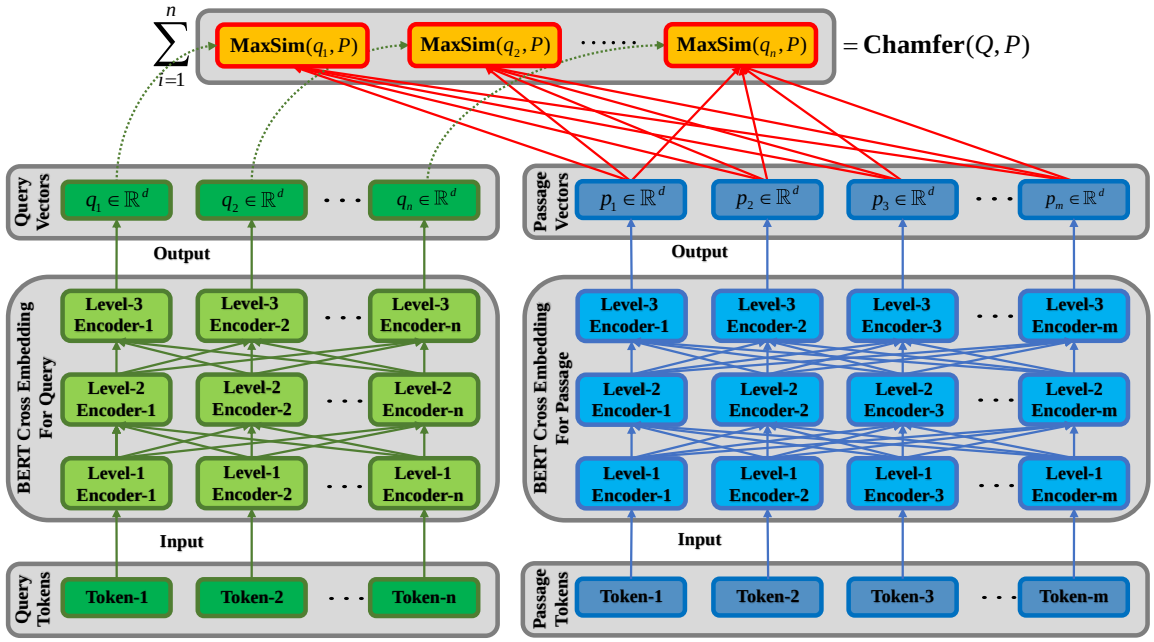

FIGURE 1. The Structure of ColBERT and its Late Iteraction Process.

significantly alleviates the pressure on bus transmission by introducing residual compression. Based on the framework of ColBERTv2, PLAID [9] further introduces centroid pruning and centroid interaction mechanisms, and accelerates the late interaction of ColBERTv2 through numerous fine-grained kernel optimizations. Further, EMVB [10] follows the direction of PLAID and designs a more efficient late interaction kernel, further improving the efficiency of late interaction.

However, we note that in the above methods and optimizations, two issues persist. First, compared to single-vector retrieval models, multi-vector retrieval requires storing significantly more vectors, leading to substantial memory overhead. Although models like Muvera [11] convert multi-vectors into single vectors using LSH [12], their performance remains suboptimal. Therefore, a good pruning strategy becomes necessary. Additionally, since ColBERT's late interaction based on MaxSim essentially involves a Top-1 selection for each Query Token, this Top-1 selection largely limits the model's expressiveness [13]. Hence, a richer and more fine-grained late interaction should also be considered as a direction for improvement.

Based on these considerations, we propose the DeepBERT multi-vector retrieval model, whose main innovations include:

- Building on the encoding framework of ColBERT to generate multi-vectors, we propose a mechanism that leverages LLMs (Large Language Models) to generate importance scores (weights) for each Token without the need for fine-tuning the model.
- Based on the Token weights, we propose an extremely simple pruning strategy that incurs almost no loss in the model's retrieval performance.
- We propose a method for dynamically adjusting the retrieval strategy to make our approach adaptable to specific tasks.

Our method surpasses the state-of-the-art methods on numerous datasets. Additionally, to pay homage to the open-source model DeepSeek [14] for its significant contributions to the development of large language models and for providing us with affordable LLM APIs, we name our model DeepBERT.

II. PROBLEM DEFINITION

As shown in Figure 1, the traditional late interaction in multi-vector retrieval, as represented by ColBERT, can be described as follows: Given a query Q and a passage P, they are encoded into multi-vector representations Q=$\{q_1, q_2, ..., q_n\}$ and P=$\{p_1, p_2, ..., p_m\}$ using a cross-encoder (e.g., BERT). For each $q_i \in Q$, a Maximum Inner Product Search (MIPS) is performed on the vectors in P to obtain MaxSim(q_i, P). Further, by summing the MaxSim scores for each q_i and applying appropriate normalization, the similarity score [1] between Q and P is obtained, i.e., Chamfer(Q, P)= $\sum_{i=1}^{n}$ MaxSim(q_i, P).

We note that using the sum of MaxSim as a similarity measure has a long history in computer vision [15] [16] [17] and graphics [18], where it is referred to as the Chamfer distance. Therefore, we choose to use the term "Chamfer."

As shown in Figure 2, we extend this process to a matrix interaction. We define the matrix $\mathbf{S} \in \mathbb{R}^{n \times m}$ as the similarity matrix, where each element s_{ij}=$\langle q_i, p_j \rangle$ represents the similarity between each sub-vector (Token) in Q and P. We define $\mathbf{W} \in \mathbb{R}^{n \times m}$ as the weight matrix, where each element w_{ij}=$w_{q_i} w_{p_j}$ represents some importance weight for each similarity s_{ij}, with w_{q_i} and w_{p_j} denoting the importance

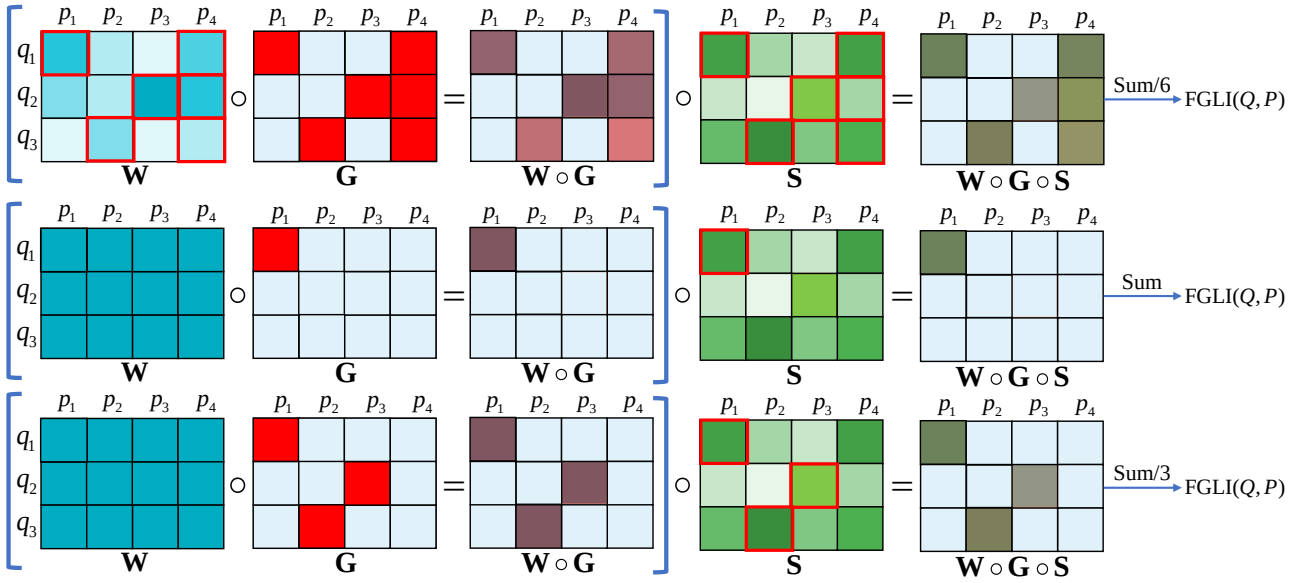

FIGURE 2. The Matrix Model for Late Interaction. From top to bottom, there is the complete fine-grained late interaction, ColBERT's late interaction, and dense retrieval's interaction.

weights for q_i and p_j, respectively, which will be elaborated later. We define $\mathbf{G} \in \mathbb{R}^{n\times m}$ as the gating matrix, used to activate the corresponding elements in $\mathbf{S}$ and $\mathbf{W}$ to participate in the final similarity calculation. Finally, we define the final similarity as the average of the non-zero elements in the Hadamard product matrix $(\mathbf{G} \circ \mathbf{W} \circ \mathbf{S})$, i.e., $\text{FGLI}(Q, P)=\frac{1}{Z}(\mathbf{G} \circ \mathbf{W} \circ \mathbf{S})$.

Under this framework, the traditional late interaction is equivalent to setting the weight matrix $\mathbf{W}=\mathbf{1} \in \mathbb{R}^{n\times m}$ and controlling $\mathbf{G}$ to select the element with the highest similarity in each row of $\mathbf{S}$ using a Top-1 strategy. The final similarity then becomes the average of the non-zero elements in the Hadamard product of the three matrices (as shown at the bottom of Figure 2). It is worth noting that this framework can also be compatible with dense retrieval by defaulting q_1 and p_1 to the <cls> prefix of the query and passage, respectively. This is achieved by setting $\mathbf{W}=\mathbf{1} \in \mathbb{R}^{n\times m}$ and $g_{11}=1$ in $\mathbf{G}$ while setting all other elements to 0. (as shown in the middle of Figure 2)

Based on this problem definition, our main direction for improvement is to introduce an importance weight for each Token in Q and P, making $\mathbf{W} \neq \mathbf{1} \in \mathbb{R}^{n\times m}$, and to dynamically adjust the gating strategy of $\mathbf{G}$ to adapt to different tasks. This enables fine-grained late interaction (FGLI) in multi-vector retrieval (as shown at the top of Figure 2).

III. METHODOLOGY

A. LLM-BASED WEIGHT GENERATION

As shown in Figure 3, we designed an LLM-based weight generation mechanism to assign an importance weight to each Token in Q and P.

In the first part, we used the pre-trained ColBERTv2 model to generate vector representations for each Token in Q and P. In the second part, we attempted to generate preliminary scores for each Token in Q and P. We designed two lightweight yet highly efficient scoring methods. The first is a matching-based score. Taking passage vectors as an example, we used the smoothed inverse document frequency of each p_j in the query set $\{P^{(1)}, P^{(2)}, ...\}$ as a score for p_j, denoted as $\text{IDF}(p_j)$, which measures the rarity of p_j in the overall passage set.

The second is an attention-based score. We observed that BERT generates vector representations for each Token by applying a series of SoftMax operations to the attention tensor corresponding to that Token in the top encoder layer. However, this attention tensor not only generates vector representations but also contains an attention score for each Token. Specifically, at the top layer of BERT, we obtain a three-dimensional matrix of shape $A\times|P|\times|P|$, where the element at position (a, j, k) represents the attention between p_j and p_k in the a-th attention head. For any Token p_j, summing the attention scores related to it across all heads yields an attention score for q_j, which measures the importance of p_j in the entire passage P.

Finally, we combined the original text, the IDF scores of each Token, the attention scores of each Token, and a fixed prompt as input to the DeepSeek model to obtain the importance weights for each Token in the text.

B. WEIGHT-BASED PASSAGE VECTOR PRUNING

The pruning strategy is based on the assumption that in many NLP retrieval tasks, the answer to a query is often concentrated in a few Tokens within the passage (as shown in the top of Figure 4). The importance weights we designed are an efficient method to filter out Tokens that contain high-density important information in a sentence.

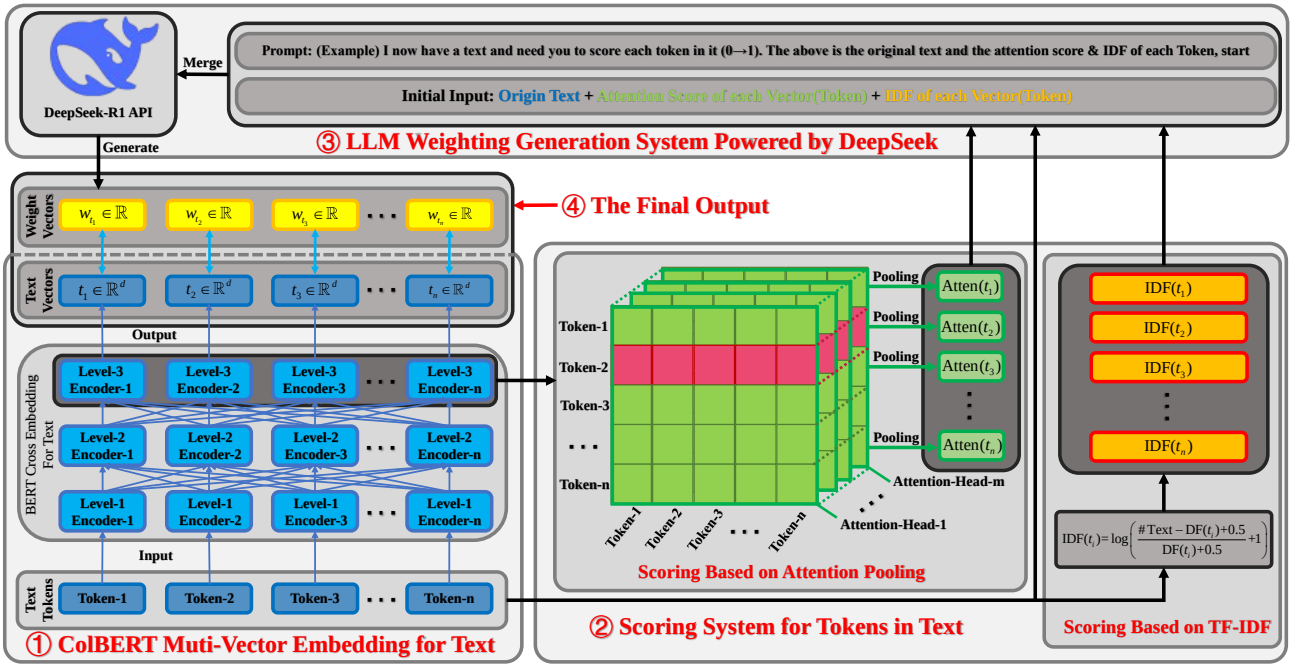

FIGURE 3. DeepBERT's whole process of generating Token vectors and Token weights

Fact Answering Task, Example Generated by OpenAI ChatGPT-4o

Q: What is the SOTA LLM Model

A: China's **DeepSeek R1** model, marking significant milestone in AI development. R1 employs reinforcement learning techniques to autonomously refine its performance without supervision, a feature still under development by other AI companies.

Analysis Task, Example Generated by OpenAI ChatGPT-4o

Q: What is the next SOTA LLM Model

A: Predicting the next state-of-the-art large language model (LLM) involves analyzing current trends and future possibilities in AI research, So, the next SOTA LLM might not be just "**bigger and better**," but smarter, more **adaptive**, and **efficient**, reshaping how we interact with AI.

FIGURE 4. Examples of Token Focus Hypothesis in NLP Retrieval Tasks

In practice, we set a pruning percentage $t_{cs}\%$ and retain only the top $t_{cs}\%$ of Tokens in P based on their importance weights. Subsequent experiments confirmed our hypothesis that even an aggressive pruning strategy (retaining only 10% of Tokens) achieved good retrieval performance.

C. DYNAMIC GENERATION OF THE GATING MATRIX

We introduced another assumption that the distribution of important Tokens in a passage varies across different tasks (as shown in the top and bottom of Figure 4). Therefore, for different retrieval datasets, we should dynamically adjust the retrieval strategy. For example, in fact-based tasks, we typically control the sparsity of $\mathbf{G}$ to perform Top-1 retrieval, while in analytical tasks, we control the sparsity of $\mathbf{G}$ to perform Top-K (K > 1) retrieval.

The ColBERT model is trained uniformly with Top-1 as the training task. While we could fine-tune the pre-trained model, we improved a more efficient adjustment strategy based on the method in [13]: without changing parameters, we adjust the model for the target task through the following steps:

- **Input**. Target task-specific corpus $\{P^{(1)}, ..., P^{(N)}\}$ and a small amount of labeled data $\{(Q^1, P^1_+), ..., (Q^K, P^K_+)\}$.
- **Retrieval**. Use the pre-trained model to retrieve candidate passages $\{P^{(i_1)}, P^{(i_2)}, ...\}$ for each query Q^i.
- **Evaluation**. Recompute the scores of the candidate passages using different retrieval strategies (Top-1/Top-2/...) and re-rank them.
- **Adaptation**. Based on the labeled data, select the retrieval strategy (e.g., nDCG@10) that performs best in the evaluation phase as the alignment strategy for the task.

Once the retrieval strategy is determined, the gating matrix $\mathbf{G}$ can be used to control the retrieval strategy. The top part of Figure 2 illustrates a Top-2 retrieval controlled by $\mathbf{G}$.

IV. EXPERIMENTS AND MAIN RESULTS

A. EXPERIMENTAL SETUP

We followed the same approach as ColBERTv2, training our model based on MiniLM [19]. The model was trained for 200k steps, with a linearly varying learning rate. The learning rate started at 0, warmed up to the final learning rate (1e-5) within the first 50k steps, and then linearly decayed (from 2e-5 to 0) over the remaining 150k steps. For the deployment of ColBERT, we used the same kernel as PLAID, which is currently the most mature and fastest version of ColBERT.

Additionally, we used the DeepSeekR1 version of the LLM API. When adjusting the retrieval strategy, we uniformly selected 16 labeled samples from the dataset (our ex-

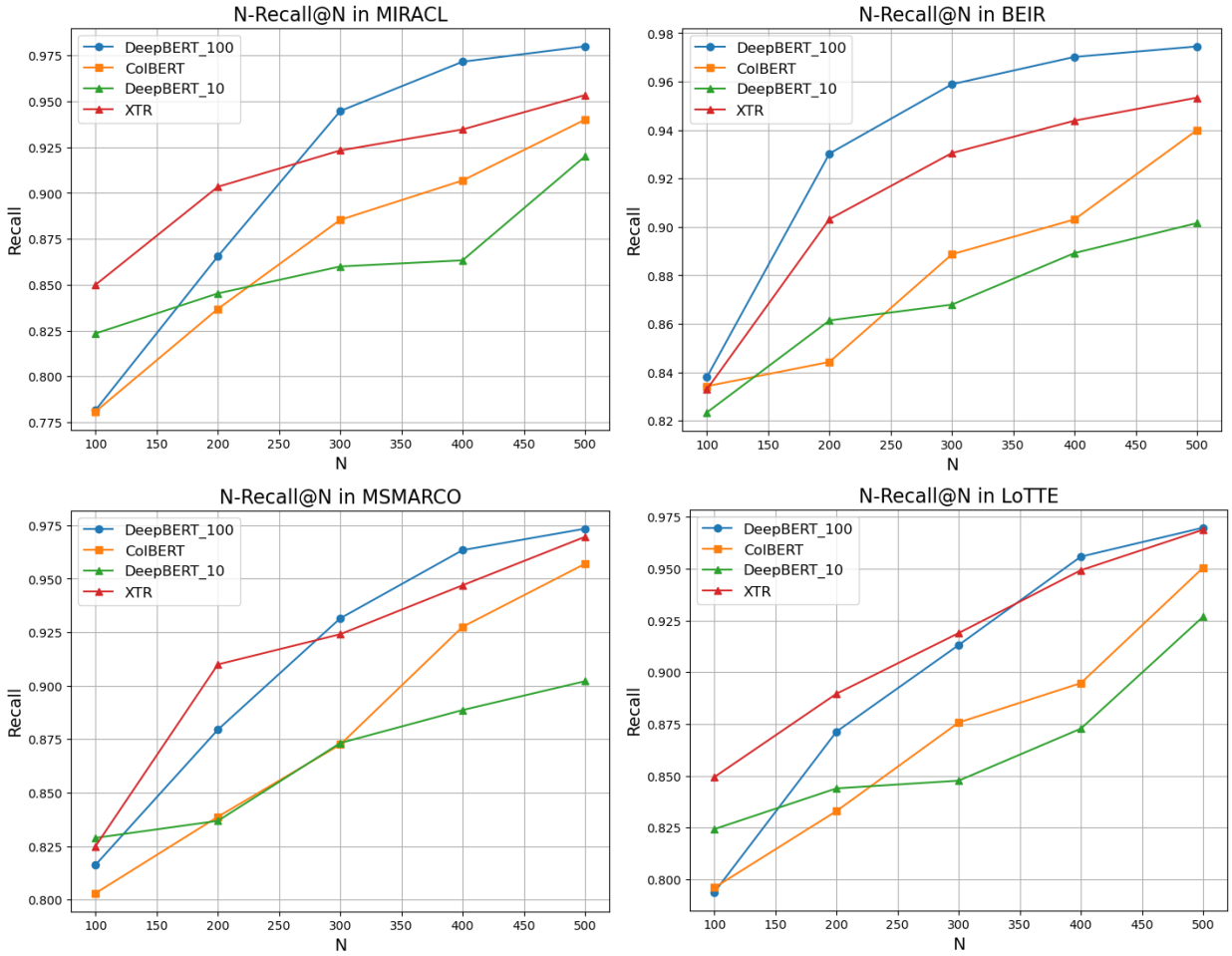

FIGURE 5. End-2-End Result Comparison of DeepBERT and other SOTA Models

periments show that this scale of labeled samples is sufficient for selecting the retrieval strategy).

Finally, to ensure the model's generalization ability, we selected four widely used datasets with diverse domain tasks to test our model: MIRACL [20] (Dev Set), known for its multilingual focus; LoTTE [8], known for its long-tail and domain-specific characteristics; and the widely used Ms-Marco [21] (v2 version) and BEIR [22]. Results on other datasets were similar and are not shown here.

B. MODEL PERFORMANCE

We first tested the model's performance relative to ColBERT without end-to-end optimization.

Pruning Results. Starting from 100%, we pruned the passage vectors in 10% increments (and finally 5%) and tested DeepBERT's Recall and nDCG on four datasets for each pruning level. From the experimental results (Table 1), under slight and moderate pruning (greater than 30%), DeepBERT's retrieval performance comprehensively outperformed ColBERT, demonstrating that fine-grained interaction can enhance the model's expressiveness. Additionally, under aggressive pruning (10%), although the model's performance decreased, it still had a significant advantage over ColBERT, and this advantage was achieved while saving nearly 90% of memory. Only when the pruning percentage reached 5% did the model's performance significantly decline.

Ablation Study. We conducted several ablation strategies, including directly using IDF values as importance weights, directly using attention scores as importance weights, and directly having an LLM generate importance scores for all tokens in a sentence. We normalized the Recall@500 results of the full version of DeepBERT to 100%, and the experimental results are shown in Table 2. It can be observed that when the LLM is provided with inputs derived from IDF and Attention, the performance is significantly suboptimal and unstable. This highlights the substantial impact of prompt design on LLM performance [23]. On the other hand, the versions incorporating IDF and Attention achieved promising results, particularly the Attention-based version, demonstrating the powerful expressive capabilities of Transformers in text processing.

TABLE 1. Performance Comparison Between ColBERT and DeepBERT (Pruned)

Dataset	Metric	ColBERT	DeepBERT Pruned									
			100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	5%
MIRACL	Recall@500	85.4	95.4	95.2	95.2	94.8	94.5	94.4	93.8	92.5	91.0	89.7	72.5
	MRR@10	0.357	0.427	0.425	0.426	0.421	0.419	0.414	0.415	0.403	0.382	0.350	0.225
	nDCG@10	0.360	0.435	0.433	0.434	0.429	0.426	0.422	0.423	0.411	0.390	0.358	0.230
LoTTE	Recall@500	81.4	93.4	91.2	90.3	90.8	90.5	89.2	88.5	87.0	85.7	82.3	68.5
	MRR@10	0.326	0.452	0.445	0.446	0.425	0.421	0.426	0.408	0.403	0.399	0.329	0.198
	nDCG@10	0.335	0.465	0.458	0.459	0.437	0.432	0.437	0.418	0.412	0.408	0.337	0.205
MsMarco	Recall@500	86.4	96.9	96.3	96.1	95.7	93.2	91.7	92.1	90.8	89.3	89.1	75.4
	MRR@10	0.384	0.447	0.439	0.428	0.431	0.412	0.406	0.395	0.386	0.388	0.379	0.302
	nDCG@10	0.395	0.460	0.452	0.440	0.443	0.424	0.418	0.407	0.398	0.400	0.391	0.310
BEIR	Recall@500	91.4	95.3	94.3	94.5	94.3	92.9	90.6	89.7	88.3	89.4	88.1	80.5
	MRR@10	0.377	0.433	0.440	0.434	0.403	0.407	0.399	0.402	0.398	0.390	0.386	0.316
	nDCG@10	0.385	0.445	0.452	0.446	0.414	0.418	0.410	0.412	0.409	0.401	0.393	0.323

TABLE 2. Ablation Study of DeepBERT

Dataset	Individual Components				Combined Components			
	IDF	Attention	LLM	Base (DeepBERT)	(IDF+Attn)/2	IDF+LLM	Attn+LLM	Base (DeepBERT)
MIRACL	79.4%	85.8%	40.3%	100.0%	86.5%	90.3%	95.6%	100.0%
LoTTE	82.7%	88.2%	59.3%	100.0%	90.6%	85.3%	97.4%	100.0%
MsMarco	76.6%	93.7%	20.4%	100.0%	90.1%	84.8%	95.2%	100.0%
BEIR	74.3%	91.6%	42.4%	100.0%	92.2%	89.8%	94.3%	100.0%

C. PERFORMANCE OF THE END-TO-END MODEL

In this section, we evaluate the end-to-end performance of the model. Specifically, the end-to-end framework is designed to handle large-scale datasets. Our implementation process is as follows: given a query Q and a large-scale collection of N paragraphs $\mathscr{P}$=$\{P^{(1)}, \ldots, P^{(N)}\}$:

- **Input**. Target corpus $\mathscr{P}$=$\{P^{(1)}, P^{(2)}, \ldots, P^{(N)}\}$ and a query Q=$\{q_1, q_2, \ldots, q_n\}$.
- **Embedding**. Generate a vector for each token in Q and $\mathscr{P}$, i.e., Q=$\{q_1, \ldots, q_n\}$ and $P^{(j)}$=$\{p_1^{(j)}, \ldots, p_m^{(j)}\}$.
- **Retrieval**. For each $q_i \in Q$, perform Maximum Inner Product Search (MIPS) across all paragraph sub-vector sets $P^{(1)} \cup \ldots \cup P^{(N)}$ to obtain the Top-K paragraph sub-vectors.
- **Backtracking**. Merge the MIPS results of the n q_i's to obtain $n \times K$ paragraph sub-vectors. Backtrack each sub-vector to its corresponding paragraph to obtain $\mathscr{P}'$=$\{P^{(1)}, \ldots, P^{(M)}\}$.
- **Re-ranking**. Re-rank the candidate paragraphs $\mathscr{P}'$ based on the exact distance (i.e., the FGLI score in our model) to obtain the most similar paragraphs.

We selected DiskANN [24] as our MIPS search algorithm, which is currently the state-of-the-art graph-based nearest neighbor search algorithm. And we set Top-5000 MIPS search here. For comparison, we selected some widely recognized end-to-end implementations of ColBERT, such as ColBERTv2 and PLAID, as well as the recently proposed and promising new end-to-end method XTR [25].

The results are shown in Figure 5. The full version of DeepBERT achieved outstanding performance across all datasets, and even the aggressively pruned (10%) version of DeepBERT demonstrated remarkable results.

D. CONCLUSION

The DeepBERT model proposed in this paper has made three significant advancements in the field of multi-vector neural retrieval: First, through LLM-driven fine-grained weight generation, it effectively integrates semantic statistical features with attention priors, achieving precise localization of key semantic units. Second, the dynamic pruning-based vector compression method maintains over 80% retrieval performance even at a 10% retention rate, breaking through the memory bottleneck of traditional multi-vector retrieval. Third, the dynamic adaptation mechanism of the gating strategy enables the model to autonomously select Top-K interaction modes based on task characteristics.

Experiments demonstrate that this method exhibits strong robustness in complex scenarios such as cross-language

and long-tail distributions. Its two-stage retrieval framework (coarse screening + fine re-ranking) provides a feasible solution for real-time retrieval in billion-scale document libraries. Future work will explore end-to-end training of the weight generation module and multi-strategy parallel computing based on the MoE architecture.

Ps. Our model was primarily evaluated on English datasets, as publicly available corpora are predominantly in English. This approach does not involve any legal declarations or ethical concerns.

V. ACKNOWLEDGMENT

We would like to express our sincere gratitude to Hongyi Duan (dannhiroaki.github.io) for his invaluable support throughout this research. We engaged in extensive discussions, during which he contributed numerous constructive and insightful ideas and suggestions. Additionally, we hereby declare that any content in this paper related to our collaborative discussions has been modified and published with the explicit consent of Hongyi Duan.

REFERENCES

[1] Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 1972, 28(1): 11-21.doi.org/10.1108/eb026526

[2] Robertson, S. E., Spärck Jones, K. Simple, proven approaches to text retrieval. Technical Report, University of Cambridge, Computer Laboratory, 1994. microsoft.com/research/text-retrieval

[3] Devlin, J., Chang, M. W., Lee, K., et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), 2019: 4171-4186. doi.org/10.18653/v1/N19-1423

[4] Radford, A., Narasimhan, K., Salimans, T., et al. Improving Language Understanding by Generative Pre-Training. Technical Report, OpenAI, 2018. openai.com/research-covers/language-unsupervised

[5] Sanh, V., Debut, L., Chaumond, J., et al. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. arXiv preprint arXiv:1910.01108, 2019. doi.org/10.48550/arXiv.1910.01108

[6] Khattab, O., Zaharia, M. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2020: 39-48.2019.doi.org/10.1145/3397271.3401075

[7] Santhanam, K., Khattab, O., Saad-Falcon, J., et al. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. arXiv preprint arXiv:2112.01488, 2021. doi.org/10.18653/v1/2022.naacl-main.272

[8] Jha, R., Wang, B., Günther, M., et al. Jina-ColBERT-v2: A General Purpose Multilingual Late Interaction Retriever. arXiv preprint arXiv:2408.16672, 2024. doi.org/10.48550/arXiv.2408.16672

[9] Santhanam, K., Khattab, O., Potts, C., et al. PLAID: An Efficient Engine for Late Interaction Retrieval. Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM), 2022: 1747-1756. doi.org/10.1145/3511808.3557325

[10] Nardini, F. M., Rulli, C., Venturini, R. Efficient multi-vector dense retrieval with bit vectors. European Conference on Information Retrieval. Cham: Springer Nature Switzerland, 2024: 3-17. doi.org/10.1007/978-3-031-56060-6_1

[11] Dhulipala L, Hadian M, Jayaram R, et al. MUVERA: Multi-vector retrieval via fixed dimensional encodings[J]. arXiv preprint arXiv:2405.19504, 2024. doi.org/10.48550/arXiv.2405.19504

[12] Indyk P, Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality[C]//Proceedings of the thirtieth annual ACM symposium on Theory of computing. 1998: 604-613.doi.org/10.1145/276698.276876

[13] Qian Y, Lee J, Duddu S M K, et al. Multi-vector retrieval as sparse alignment[J]. arXiv preprint, 2022. doi.org/10.48550/arXiv.2211.01267

[14] Guo D, Yang D, Zhang H, et al. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning[J]. arXiv preprint arXiv:2501.12948, 2025. doi.org/10.48550/arXiv.2501.12948

[15] Athitsos V, Sclaroff S. Estimating 3D hand pose from a cluttered image[C]//Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, 2003, 2: II-432.doi.ieeecomputersociety.org/10.1109/CVPR.2003.1211500

[16] Barrow H G, Tenenbaum J M, Bolles R C, et al. Parametric correspondence and chamfer matching: Two new techniques for image matching[C]//Proceedings of the Image Understanding Workshop. Science Applications, Inc, 1977: 21-27. dl.acm.org/doi/10.5555/1622943.1622971

[17] Fan H, Su H, Guibas L J. A point set generation network for 3D object reconstruction from a single image[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 605-613.doi.org/10.48550/arXiv.1612.00603

[18] Li C L, Simon T, Saragih J, et al. LBS autoencoder: Self-supervised fitting of articulated meshes to point clouds[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 11967-11976. doi.org/10.48550/arXiv.1904.10037

[19] Wang W, Wei F, Dong L, et al. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers[J]. Advances in Neural Information Processing Systems, 2020, 33: 5776-5788.doi.org/10.48550/arXiv.2002.10957

[20] Zhang X, Thakur N, Ogundepo O, et al. Making a MIRACL: Multilingual information retrieval across a continuum of languages[J]. arXiv preprint arXiv:2210.09984, 2022. doi.org/10.1162/tacl_a_00595

[21] Bajaj P, Campos D, Craswell N, et al. MS MARCO: A human generated machine reading comprehension dataset[J]. arXiv preprint arXiv:1611.09268, 2016. doi.org/10.48550/arXiv.1611.09268

[22] Thakur N, Reimers N, Rücklé A, et al. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models[J]. arXiv preprint arXiv:2104.08663, 2021. doi.org/10.48550/arXiv.2104.08663

[23] Yao Y, Duan J, Xu K, et al. A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly[J]. High-Confidence Computing, 2024: 100211. https://doi.org/10.1016/j.hcc.2024.100211

[24] Jayaram Subramanya S, Devvrit F, Simhadri H V, et al. DiskANN: Fast accurate billion-point nearest neighbor search on a single node[J]. Advances in Neural Information Processing Systems, 2019, 32.https://papers.nips.cc/2019/09853c7fb1d

[25] Lee J, Dai Z, Duddu S M K, et al. Rethinking the role of token retrieval in multi-vector retrieval[J]. Advances in Neural Information Processing Systems, 2023, 36: 15384-15405. doi.org/10.48550/arXiv.2304.01982