An Analysis on Matching Mechanisms and Token Pruning for Late-interaction Models

QI LIU, GANG GUO, JIAXIN MAO, ZHICHENG DOU, and JI-RONG WEN, Renmin University of China, Beijing, China
HAO JIANG, XINYU ZHANG, and ZHAO CAO, Huawei, Hangzhou, China

With the development of pre-trained language models, the dense retrieval models have become promising alternatives to the traditional retrieval models that rely on exact match and sparse bag-of-words representations. Different from most dense retrieval models using a bi-encoder to encode each query or document into a dense vector, the recently proposed late-interaction multi-vector models (i.e., ColBERT and COIL) achieve state-of-the-art retrieval effectiveness by using all token embeddings to represent documents and queries and modeling their relevance with a sum-of-max operation. However, these fine-grained representations may cause unacceptable storage overhead for practical search systems. In this study, we systematically analyze the matching mechanism of these late-interaction models and show that the sum-of-max operation heavily relies on the co-occurrence signals and some important words in the document. Based on these findings, we then propose several simple document pruning methods to reduce the storage overhead and compare the effectiveness of different pruning methods on different late-interaction models. We also leverage query pruning methods to further reduce the retrieval latency. We conduct extensive experiments on both in-domain and out-domain datasets and show that some of the used pruning methods can significantly improve the efficiency of these late-interaction models without substantially hurting their retrieval effectiveness.

CCS Concepts: • **Information systems** → **Retrieval models and ranking;**

Additional Key Words and Phrases: Neural networks, pre-trained language model, late-interaction models, token pruning, efficiency optimization

ACM Reference Format:
Qi Liu, Gang Guo, Jiaxin Mao, Zhicheng Dou, Ji-Rong Wen, Hao Jiang, Xinyu Zhang, and Zhao Cao. 2024. An Analysis on Matching Mechanisms and Token Pruning for Late-interaction Models. *ACM Trans. Inf. Syst.* 42, 5, Article 118 (April 2024), 28 pages. https://doi.org/10.1145/3639818

Q. Liu and G. Guo Equal contribution.

This research was supported by the Natural Science Foundation of China (61902209, 62377044, U2001212), and Beijing Outstanding Young Scientist Program (NO. JJWZYJH012019100020098), Intelligent Social Governance Platform, Major Innovation Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China", the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China.

Authors' addresses: Q. Liu, G. Guo, J. Mao (Corresponding author), Z. Dou, and J.-R. Wen, Being Key Laboratory of Data Management and Analysis Methods, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, Beijing, China, No. 59 Zhongguancun Street,100872; e-mails: liuqi_67@ruc.edu.cn, gguogangg@gmail.com, maojiaxin@gmail.com, dou@ruc.edu.cn, jrwen@ruc.edu.cn; H. Jiang, X. Zhang, and Z. Cao, Distributed and Parallel Software Lab, Huawei, Hangzhou, Zhejiang, China, No. 360 Jiangshu Road, 310056; e-mails: jianghao66@huawei.com, zhangxinyu35@huawei.com, caozhao1@huawei.com.

ACM 1046-8188/2024/04-ART118
https://doi.org/10.1145/3639818

1 INTRODUCTION

Document ranking has always been one of the most important tasks in information retrieval. This task aims to rank documents according to their relevance to a given query. To achieve a good performance in terms of both effectiveness and efficiency, document ranking consists of two stages. The first stage is to retrieve thousands of documents to form a candidate set from a large corpus which may consist of millions or even billions of documents efficiently. Then in the second stage, the system usually uses a more sophisticated ranking model to re-rank the retrieved candidates to further boost the retrieval effectiveness.

Traditional **Bag-of-Words (BoW)** retrieval models (e.g., BM25 [40]) rely on sparse representations and exact matches between a query and documents to rank documents. While these methods are efficient, robust, and interpretable, the inverted index and the dependence on exact match signals may lead to the vocabulary mismatch problem [16] which may limit the overall retrieval effectiveness.

Recently, deep learning and **pre-trained language models (PLMs)** such as BERT [12], RoBERTa [29] and [24, 38, 47] have made impressive improvements in many fields. In information retrieval, researchers have also leveraged the contextualized representation learned by PLMs to tackle the vocabulary mismatch problem and improve retrieval effectiveness. Some recent studies [4, 10, 11, 14, 31] leverage neural models to expand terms for original documents by modifying the term frequency or directly re-computing the term weight used in the traditional retrieval models. Other works [17, 22, 23, 30, 50], known as dense retrieval, represent queries and documents as one or more dense vector(s) and solve the vocabulary mismatch problem by performing soft matching in the vector space. The dense retrieval models have become promising alternatives to the traditional retrieval models because by using the **approximate nearest neighbor search** (**ANNs**) algorithm, they can achieve a good trade-off between retrieval effectiveness and efficiency.

For BERT-based retrieval models, the *cross-encoder* architecture [32] shown in Figure 1(a) makes thorough interaction between all query tokens and all document tokens in every transformer layer. Although such a full-interaction architecture can achieve a good ranking performance, it imposes a high time cost which prevents it from being used as a first-stage retrieval model. To solve this problem, some studies [22, 46, 50] propose to use a *bi-encoder* architecture to independently encode queries and documents into dense vectors, and the architecture is illustrated in Figure 1(b). Because there is no interaction between the query tokens and document tokens, we can pre-compute the embeddings for documents. When doing retrieval, the model only needs to encode the query and then search for the matching documents with the help of ANNs algorithms (e.g., Faiss [21]), within an acceptable latency for the first stage retrieval. However Luan et al. [30] have theoretically proved and empirically demonstrated that using multiple vectors to represent documents is more effective than a single-vector representation. Therefore, recent studies [17, 20, 23] employ multiple vectors to represent a document in the first-stage retrieval and enhance the interaction between queries and documents. This *late-interaction* architecture is shown in Figure 1(c). Combining the advantages of cross-encoders and bi-encoders, Khattab and Zaharia [23] proposed a soft late-interaction model named ColBERT, and Gao et al. [17] proposed a hard late-interaction model named COIL. These two models are more effective than the bi-encoders and much more efficient than the cross-encoders so they can be used as first-stage retrieval models.

Although the above late-interaction models achieve a good balance between retrieval effectiveness and efficiency, their matching mechanisms are underinvestigated, which may undermine their interpretability and robustness, especially when compared to the traditional BoW-based retrieval models. Moreover, late-interaction models require to use all token embeddings

to represent documents, leading to a significant increase in storage requirements and a decrease in efficiency. Therefore, it's important to understand their matching mechanisms and investigate whether and how we can leverage such matching mechanisms to reduce storage overhead and enhance efficiency. Specifically, in this paper, we propose several heuristic pruning methods according to the analysis of the matching mechanism, and study the following research questions:

— **RQ1:** How do the late-interaction models work and perform matching at the token level?
— **RQ2:** What are the differences in performance among various models when applying pruning?
— **RQ3:** How do different pruning methods and pruning ratios affect the effectiveness and efficiency of retrieval models?

Regarding **RQ1**, in this study, we first investigate how late-interaction models compute the relevance score for the query-document pair to understand the matching process of late-interaction models. We find that the computation of the relevance score heavily relies on the co-occurrence signals and the important words in documents. Such behaviors are very similar to the traditional BoW-based retrieval models in which we can safely ignore words that provide little information for relevance estimation, and use some weighting scheme, such as IDF, to emphasize the important words in the documents.

Inspired by the matching mechanisms of traditional retrieval models and the similarity between late-interaction models and traditional retrieval models, we further ask: *can we achieve comparable effectiveness when only using the important words in documents to compute the relevance score?* Therefore, we propose and evaluate several simple document token pruning strategies, which prune document tokens based on their positions, IDF values, and attention scores, to improve storage efficiency. Furthermore, we study query token pruning methods that reduce the query latency in the retrieval stage.

To address **RQ2** and **RQ3**, we conduct extensive experiments on a popular ad-hoc search benchmark (i.e., the MS MARCO dataset [5]) and a popular zero-shot retrieval dataset (i.e., BEIR [43]) for different late-interaction models and different pruning strategies. Subsequently, we provide a thorough analysis of the results.

We believe the investigation of the matching mechanism can help us better understand these late-interaction models and the exploration of token pruning strategies is beneficial for reducing the storage overhead and query latency with minimal performance loss. In summary, our contributions are as follows:

— To dig into the matching process of late-interaction models, we systematically analyze how ColBERT and COIL rank the documents based on the sum-of-max scoring function and find that it heavily depends on the informative words in documents.
— We propose several simple but effective document token pruning methods and query token pruning methods for late-interaction models to improve efficiency.
— We conduct extensive experiments on both in-domain and out-of-domain datasets to compare the effectiveness of the different pruning methods on different neural retrieval models.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work. Section 3 gives a detailed analysis of the late-interaction models and further proposes several methods of document tokens pruning and query tokens pruning. Section 4 and Section 5 describe the experiment settings and the empirical results, respectively. Section 6 discusses the main findings in this paper and significant implications. Finally, we conclude this paper and list some potential limitations and future work directions in Section 7.

2 RELATED WORK

We will briefly review the previous works related to retrieval models and pruning techniques in this section.

2.1 Traditional Retrieval Models

Traditional retrieval models are based on sparse representations and hard-crafted rules such as exact lexical match models (e.g., BM25 [40]) and probability retrieval models (e.g., Query Likelihood [35]). Take BM25 as an example, to retrieve documents, this type of model computes the relevance score between the query and document based on the co-occurrence terms. Benefiting from the inverted index, these models can retrieve documents efficiently so they are widely used as the first-stage retrieval model in practice. Besides the efficiency, the matching mechanism of these traditional retrieval models is transparent so that the researchers and engineers can understand how these models work. However, it is difficult for traditional retrieval models to handle the vocabulary mismatch problem which may limit their ability in modeling the semantic relevance between queries and documents.

2.2 Neural Retrieval Models

Recently, neural network models have attracted much attention in both academia and industry and have made impressive improvements in many fields. The information retrieval field benefits from pre-trained language models such as BERT [12] and RoBERTa [29]. These neural retrieval models can be divided into two groups as below:

Neural Sparse Retrieval Models improve retrieval performance by changing the term importance in different ways. Two common methods are: a) leveraging neural language generation models (e.g., T5 [39]) to expand original documents with semantic-related words, or b) using pre-trained language models to re-compute the weights. Nogueira et al. [33, 34] modified the term frequency by using generative models to generate pseudo-queries of original documents and concatenate these pseudo-queries and original documents to form the new documents. Dai and Callan [10, 11] used the pre-trained language model to predict contextualized term weight through a regression task. Mallia et al. [31] first used a generator to expand new semantic-related words of the original text and then used a contextualized language model encoder to re-compute term weights. On the other hand, Bai et al. [4] proposed using an important predictor to re-predict term weights and a gating controller to filter or expand terms at the same time. Formal et al. [13, 14] further proposed a more simple but effective method, based on explicit sparsity regularization and a log-saturation effect on term weights. All of these works can significantly outperform BM25.

Neural Dense Retrieval Models use dense vectors to represent queries and documents. Two widely adopted architectures of dense retrieval models are cross-encoders and bi-encoders. Cross-encoders [32] concatenate the query and the document as the input of pre-trained language models, then use a function to map the output vector to a scalar score. Cross-encoders achieve superior retrieval effectiveness through the full interaction between all query tokens and all document tokens but the requirement of concatenating the query and each candidate document at the inference time makes cross-encoders impractical in the retrieval stage.

Bi-encoders encode queries and documents independently so we can pre-compute the document representations (i.e., dense vectors), and then directly use these vectors for brute force search or ANNs in the retrieval phase. Bi-encoders can be further classified into two categories: the single-vector models and the multi-vector models. Single-vector models such as RepBERT [50] and DPR [22] use a single-vector to represent a document, while Luan et al. [30] theoretically proved and empirically demonstrated that using multi-vectors to represent documents can achieve better performance. The recently proposed late-interaction models (e.g., ColBERT [23, 41] and COIL [17])

achieve a great balance between effectiveness and efficiency by using all token embeddings to represent documents and queries, and modeling their relevance using a sum-of-max score. Although these late-interaction models achieved comparable effectiveness with cross-encoders, the unclear matching process and overhead storage are still problems that need to be solved.

2.3 Pruning for Inverted Index

Many studies have investigated the pruning of the traditional inverted index [3, 6–9]. According to the different components of an inverted index that be pruned, they can be classified into different types. Some early approaches proposed to completely remove unimportant terms from the inverted index, according to relevance score [9] or IDF scores [6]. Further, researchers explored pruning methods based on term posting lists or documents, named term-centric pruning and document-centric pruning, respectively. In term-centric methods, the posting lists are pruned a proportion of postings based on the posting scores [7, 9]. However, within document-centric approaches, the decision to prune is based on the rank of a posting within the corresponding document [2, 8, 44]. Meanwhile, Altin et al. [1] explored pruning the terms in documents before building the full index. Recently, Lassance et al. [25] analyzed the pruning techniques for the learned term impact by neural sparse retrieval models, such as DeepImpact [31] and SPLADE [13].

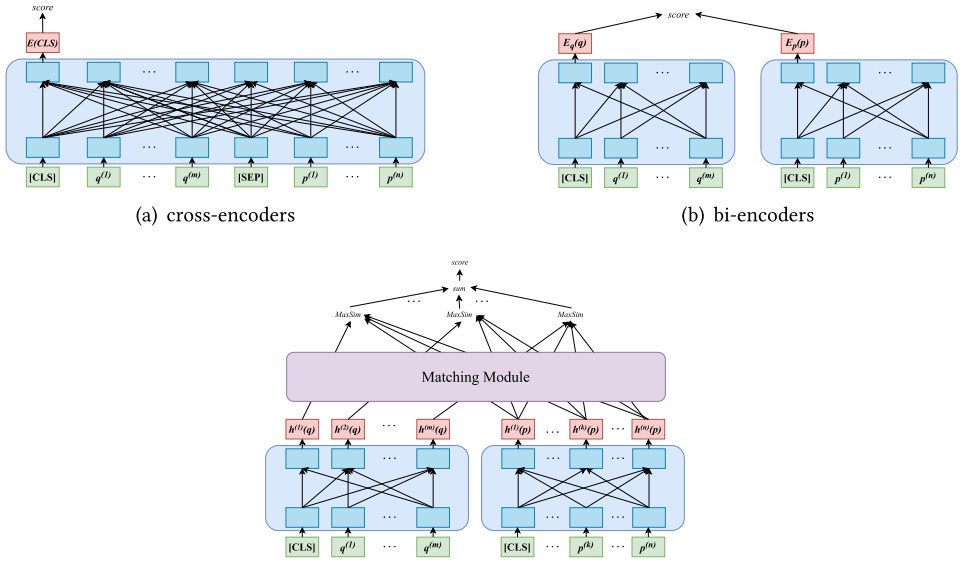
Our proposed document token pruning methods can be considered as a document-centric approach. Compared to the above work, our research primarily focuses on the pruning of dense vector indexed composed of token embeddings.

2.4 Analysis and Pruning for (Col)BERT

For BERT in information retrieval, Qiao et al. [37] and Zhan et al. [49] analyzed the behaviors of BERT when used as a cross-encoder for ranking. For ColBERT, researchers could analyze it at the token level as it uses the sum-of-max operation and some specific tokens will be activated during matching. Formal et al. [15] analyzed ColBERT's matching behavior through term importance and exact/soft matching patterns by computing the correlation between the ranking results after masking different tokens and proved ColBERT does have a notion of token importance. Different from [15], we analyze both COIL and ColBERT in a more fine-grained perspective by investigating the contribution of different tokens to the final sum-of-max score.

Such analytical studies have provided valuable heuristic insights for pruning ColBERT. Tonellotto and Macdonald [45] computed the IDF values of all documents and only used a few fixed query token embeddings according to the IDF values to retrieve fewer documents. The number of retrieved documents is much smaller than the original and experimental results show that query pruning can speed up retrieval of ColBERT without significant performance loss. Moreover, Lassance et al. [26] explored using several documents token pruning methods on ColBERT, results suggested that ColBERT indexes can be pruned up to 30% on the MSMARCO passage collection without a significant drop in performance. However, Lassance et al. [26] selected a fixed number of tokens for different-length passages, which is not reasonable, as long passages tend to have more complicated meanings. Different from [26], we use pruning methods in a dynamic way.

Additionally, some studies employed different methods to reduce the storage of ColBERT at training or indexing time. Hofstätter et al. [18] reduced the number of vectors by learning unique whole-word representations and using a learned gate vector. Qian et al. [36] employed entropy-regularized linear programming to achieve the sparsity of the matching matrix. On the other hand, Santhanam et al. [41] used an aggressive residual compression mechanism to improve the index storage. While these approaches require complex training and indexing techniques, our work focuses on simply pruning tokens without training and demonstrates that such pruning strategies are simple yet effective.



(a) cross-encoders (b) bi-encoders

(c) late-interaction models

Fig. 1. Architectures of dense retrieval models.

In addition to existing studies, we systematically analyze both COIL and ColBERT in a more fine-grained method, which investigates the scores of different tokens contributing to the final sum-of-max score. Moreover, we conduct extensive experiments which include several different token pruning methods on both COIL and ColBERT, as well as some sparse retrieval models.

3 METHODOLOGIES

In this section, we first provide a brief overview of the document retrieval task, as well as a discussion of three prevalent architectures of dense retrieval models utilized in this task. Then we conduct a detailed analysis of the late-interaction architecture, indicating how the late-interaction mechanism works. Building upon our analytical findings, we propose several methods of document token pruning and query token pruning.

3.1 Preliminaries

For the document retrieval task, given a query q and a large-scale document collection $\mathcal{D}$, we need to retrieve a set of documents that are most relevant to the query q. The core problem here is how to compute the relevance score, denoted as $s(q, d)$, between the given query q and the document $d \in \mathcal{D}$. In recent years, researchers have employed PLMs to effectively model the relevance between queries and documents. Consequently, they have proposed several different architectures for this purpose. Here, we will briefly introduce and give the formal definition of the three distinct architectures, which are shown in Figure 1.

The *cross-encoders* (Figure 1(a)) take the concatenation of the query tokens and the document tokens as input so that the models can compute the attention scores between all query tokens and all document tokens in every transformer layer. This architecture allows rich interaction between the query and the document, which leads to a great retrieval performance. On top of the transformer, there is a function, usually a linear projection, to map the output embedding, such as the average embeddings of all tokens or the [CLS] embedding, to a scalar value which is regarded as the relevance score of the given query and the document [32].

The *bi-encoders* (Figure 1(b)) encode queries and documents independently and then reduce the output of all token embeddings of the query to a single embedding and also for the document. The input is the independent query and document. While there are two encoders in this logical architecture, in practice, some use two different encoders, while others use a parameter-shared one. After the query and document are encoded by the encoder(s), we can get their corresponding token embeddings, and use the embedding of the [CLS] token [22] or mean values of all tokens embeddings [50] as the representation of the whole query or document. The inner product or cosine similarity between the query embedding and the document embedding is usually used as the final relevance score. Compared to cross-encoders, as the queries and documents can be computed independently, bi-encoders achieve efficient retrieval by encoding all documents offline and only using one embedding to represent queries and documents. Additionally, it allows the use of the approximate nearest neighborhood search (ANNs) techniques to further reduce the online retrieval latency. However, the heavy reduction of token embeddings may result in a great loss of information and limit the retrieval effectiveness.

The *late-interaction models* (Figure 1(c)) combine the advantages of the previous two methods, and independently encode the query and the document at the token level. It employs a matching module referred to as "late interaction" to compute the relevance score between the query and the document, which makes a good trade-off between efficiency and effectiveness as it can achieve comparable effectiveness as cross-encoders and comparable efficiency as bi-encoders. Most of the existing late-interaction models utilize a sum-of-max operation [23] to compute the ranking score, which can be described as:

$$Q = PLM(q) \in \mathbb{R}^{m \times h}, \quad D = PLM(d) \in \mathbb{R}^{n \times h}, \tag{1}$$

$$s(q,d) = Match(Q,D) = \sum_{q_i \in M_q} \max_{d_j \in M_d} Q_i \cdot D_j^T, \tag{2}$$

where Q represents the matrix of all query token embeddings, D represents the matrix of all document token embeddings, and m and n are the length of query and document, respectively. The two sets M_q and M_d represent the sets of query tokens and document tokens, respectively, which will be used in the matching. Equation (2) denotes the sum-of-max operation which is adopted in late-interaction models.

According to how M_q and M_d are constructed, the matching module can be implemented in two ways: *soft matching*, which aligns with ColBERT [23], or *hard matching*, which aligns with COIL [17]. For ColBERT, each query token embedding will interact with each document token embedding, regardless of if they are same. We regard this as the *soft matching*:

$$Soft\text{-}Match(Q,D) = \sum_{i=1}^{m} \max_{j=1}^{n} Q_i \cdot D_j^T. \tag{3}$$

As for COIL, only the token that occurs in both query and documents will be selected, and each query token only interacts with the same tokens in the document. As it performs like the traditional exact match, we regard this as the *hard matching*:

$$Hard\text{-}Match(Q,D) = \sum_{q_i \in q \cap d} \max_{d_j = q_i} Q_i \cdot D_j^T. \tag{4}$$

The documents encoder and queries encoder of ColBERT and COIL both share parameters so that they can map queries and documents into the same semantic space.

In order to perform the sum-of-max operation, the two models pre-compute and store all the token embeddings of document tokens. This sum-of-max operation allows the interaction between

all query tokens' embeddings and all document tokens' embeddings after the last transformer layer without embeddings reduction compression so that they can achieve better retrieval performance than bi-encoders. However, such late-interaction models store all document token embeddings through the interaction between all query tokens and all document tokens after the last transformer layer, which may result in a large storage overhead and a relatively long query latency, respectively. In this paper, we aim to tackle these two limitations with document token pruning and query token pruning strategies.

3.2 Analyses of Late-interaction Models

To understand how the late-interaction mechanism works, we first analyze the matching mechanism of ColBERT and COIL when performing document ranking. In particular, as the final relevance score is a sum of several matching scores between the query tokens and the document tokens selected by the max operation, we investigate *which* document tokens will be selected and *how much* they contribute to the final relevance score.

For the first question, we contemplate the position and IDF value of each token within a document. Specifically, due to the disparate lengths of the documents, we partition the tokens into ten bins of equal size according to their relative position or IDF values. For example, in the case of position-based partitioning, the first bin encompasses the initial 10% of tokens within the document, while in the case of IDF-based partitioning, the first bin comprises the tokens with 10% IDF values in the document. Note that we exclude all special tokens, such as the [CLS] and [SEP] tokens, in this analysis.

After partitioning tokens into different bins, we can assess their contribution to the final score for each bin, corresponding to the second question. In particular, we define two contribution metrics for one document token, named *indices contribution* and *score contribution*. The indices contribution represents whether the token is selected by the max operation in Equation (2), while the score contribution denotes the corresponding inner product score which contributes to the final score $s(q, d)$. Formally, we compute these two contribution metrics of each bin across a set of positive query-document pairs, and the whole process can be described as follows:

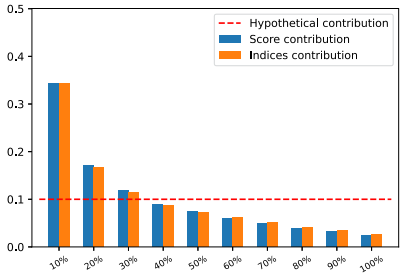
$$P_{Indice}(k) = \frac{1}{Z} \sum_{\{q,d\}} \sum_{q_i \in M_q} I\left(\arg\max_{d_j \in M_d} Q_i \cdot D_j^T \in Part(d,k) \right), \tag{5}$$

$$P_{Score}(k) = \frac{1}{Z} \sum_{\{q,d\}} \sum_{q_i \in M_q} \left(\max_{d_j \in M_d} Q_i \cdot D_j^T \right) \cdot I\left(\arg\max_{d_j \in M_d} Q_i \cdot D_j^T \in Part(d,k) \right) \tag{6}$$

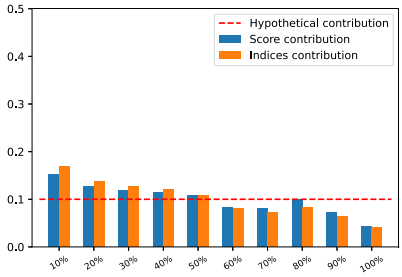
where I(condition) is an indicator function, i.e., I(condition) $= 1$ if the condition is true 0 otherwise, $Part(d, k)$ is the k-th bin of the document d partitioned by tokens' relative position or IDF values, and Z is the normalized factor.

For comparison, we include a hypothetical reference contribution in our experiment. If we assume that all tokens are contribute equally in the matching process, then for a sufficiently large set, each token within a bin should have the nearly same indices contribution and score contribution. In fact, both original ColBERT and COIL store the embeddings of all tokens. Therefore, we use the uniform contribution $P_{Hypo}(k) = \frac{1}{10}$ as a reference.

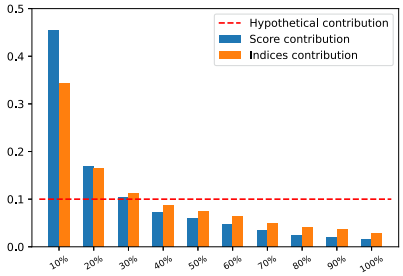
We empirically study how the document tokens with different positions and different IDF values contribute to the final relevance score and conduct the experiment on the training set of MS MARCO. The results are shown in Figure 2. In general, we can see that the tokens with different positions and IDF values contribute rather differently to the final relevance score which means that while the model is matching queries and documents, the priority of each token is different.



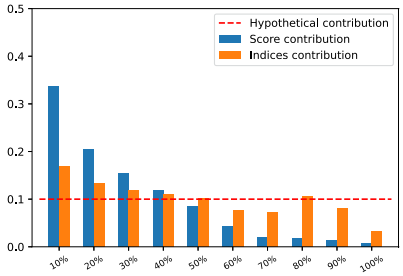
(a) Position Contribution of ColBERT



(b) IDF Contribution of ColBERT



(c) Position Contribution of COIL



(d) IDF Contribution of COIL

Fig. 2. The distribution of contributions of different tokens. Each bin consists of ten percent of the total tokens according to their relative position or IDF values. The y-axis shows the proportion contributing to the final scores of each bin.

Furthermore, the token contribution is almost negatively correlated with positions and IDF values, indicating that the words that have a higher IDF value or appear at the beginning of the documents are more important when computing the relevance score. Moreover, this phenomenon is more significant in COIL compared to ColBERT.

Look into ColBERT, when we do the position-matching analysis whose result is shown in Figure 2(a), both indices and score contributions of the ten position bins are decreasing. We suggest that this is because authors get used to writing important words at the beginning of a document. Another perspective is IDF-matching, as we can see in Figure 2(b), there are five bins that contribute more than the hypothetical contribution, and tokens in the last five bins fail to reach the hypothetical contribution, this can confirm that, while ColBERT matching, words having a higher IDF value are more significant.

For COIL, the position-matching contribution, as shown in Figure 2(c), demonstrates that both score and indices contribution consistently decrease according to the position. There are three bins' contributions higher than the hypothetical contribution, and the hypothetical contribution is not reached by the last seven bins. The score contribution drops faster than the indices contribution which means the average score of individual tokens in every bin decreases. We can deduce from the two mentioned explanations that the preceding token is more essential. Although there are modest swings in indices contribution in the IDF-matching shown in Figure 2(d), the tendency is that IDF-indices contribution decreases as IDF values decrease. Besides, the IDF-score contribution consistently decreases, and the indices contribution becomes smaller than the score contribution since the fifth bin. We can deduce from Figure 2(d) that tokens with low IDF values contribute less to the final score than tokens with high IDF values.

Table 1. Contribution to the Final Relevance Score

Co-occurrence	Others	Non-stop words	Stop words
69.1%	30.9%	74.3%	25.7%

Co-occurrence means the contributions of score that the token appears in both the query and the document.

Comparing the above two systems, when looking into the Position Contribution, we can see that ColBERT has a narrower contribution range and smoother volatility than COIL. Specifically, the score contribution and indices contribution of ColBERT is from (34.3%, 34.4%) to (2.4%, 2.6%) and COIL is from (45.4%, 34.2%) to (1.6%, 2.8%). For the IDF contribution, ColBERT's discrepancy between score contribution and indices contribution is substantially smaller than COIL's due to the different matching processes of ColBERT and COIL (soft matching vs. hard matching). Revisit the concept of soft matching and hard matching: soft matching in ColBERT will choose the highest-scoring token from all tokens, whereas in COIL, hard matching will choose only from the same tokens, so COIL may hold more similarities compared to traditional retrieval models.

Moreover, for ColBERT, we analyze how the tokens that occur in both the query and document as well as the stop words and non-stop words contribute to the relevance score. Table 1 shows the score contribution of concurrence tokens. From the results, we can see that ColBERT allows the sum-of-max operation to interact with any document tokens, but the document tokens which also appear in the query still obtain much higher attention scores. Table 1 also shows the contributions of stop words and non-stop words to the final relevance score. The results show that ColBERT also pays less attention to stop words just like traditional retrieval models.

In summary, these results demonstrate that the behavior of ColBERT and COIL is similar to the traditional retrieval models, that is, paying more attention to important words and exact match. Therefore, we further propose to prune some tokens for these late-interaction models.

3.3 Document Token Pruning

We describe the **Document Token Pruning (DTP)** pipeline in this section, shown in Figure 3. We employ the pruning methods only during the process of inference and indexing, which we name post-pruning, rather than pruning at the input of the model or combining pruning with model training. This is because we have found that the post-pruning methods result in less performance loss than pruning at the input or simply adding a regularization during training in the pilot experiment. Moreover, the utilization of post-pruning can save a significant amount of resources because it avoids additional training of the pruned model. We train late interaction models following the original settings in [17, 23], and more training details can be found in Section 4.3.

After training models, we add a pruning module to determine whether to store the token embedding. Based on the analysis results presented in Section 3.2, we propose three heuristic methods for pruning tokens to investigate how the performance of these late-interaction models changes with different pruning methods. All pruning methods perform in a dynamic way and tend to preserve a given portion of important tokens. More specifically, we will designate a *remaining ratio*, denoted as α in the following, which determines the percentage of tokens from various documents that will be retained in the index. In other words, for each document, only $l' = \lfloor l \times \alpha \rfloor$ tokens' embeddings from it will be stored. Here l is the length of the original tokenized document. In the following, we introduce the three pruning methods we propose in detail.

First-α Pruning. The first pruning method entails retaining the initial $\alpha \times 100\%$ proportion of tokens in each document. We employ this approach due to the tendency of document authors to include crucial information at the beginning of the document. This method will be a simple but

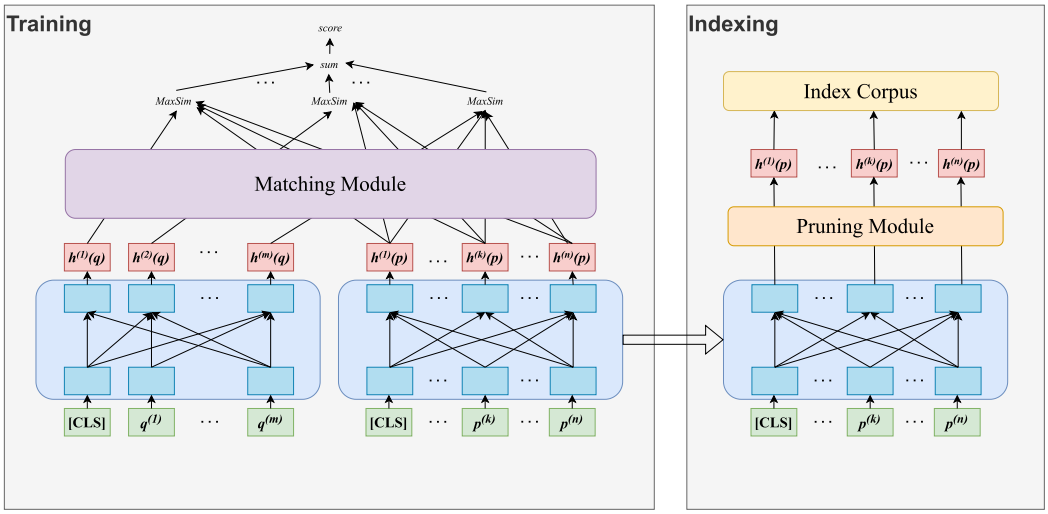


Fig. 3. The pipeline of document token pruning.

effective one.

$$D' = D[1 : l'] \tag{7}$$

IDF-Top-α Pruning. Another method we propose is IDF-Top-α pruning, where document tokens are pruned based on their IDF values. IDF is a commonly employed term weighting schema in traditional retrieval models, which is both straightforward and highly effective. To implement this method, we arrange the tokens of documents in descending order of their IDF values, then retain and store the embeddings of the document tokens that correspond to the top $\alpha \times 100\%$ IDF value.

$$\textit{Top-IDF-Index} = \textit{Top-}l'(IDF(d)) \tag{8}$$

$$D' = D[\textit{Top-IDF-Index}] \tag{9}$$

Attention-Top-α Pruning. The third method is Attention-Top-α pruning. Attention score is a common metric to evaluate the importance of a token, here we take the interaction between document token embeddings as the self-attention matrix and take the column sum of the matrix as the importance of a token. Note that the self-attention matrix is symmetric, so the column sum and the row sum are the same. The Attention-Top-α pruning can be formalized as:

$$A = softmax(DD^T) \cdot 1_n \tag{10}$$

$$\textit{Top-Att-Index} = \textit{Top-}l'(A) \tag{11}$$

$$D' = D[\textit{Top-Att-Index}] \tag{12}$$

Notice that we always retain the special tokens that appear at the beginning of the sequence, specifically [CLS] and [D] for ColBERT, and [CLS] for COIL. We do this because the embedding of special tokens is usually considered to encapsulate the information of the entire textual passage. Therefore, for the latter two approaches, the resulting index is, in fact, comprised of l' tokens that include special tokens as well as tokens with the highest IDF values or attention scores.

To summarize, the document token pruning methods can be expressed as follows:

$$Q = PLM(q), \quad D' = DTP(PLM(d)), \\ DTP \in \{\text{First-}\alpha, \text{IDF-Top-}\alpha, \text{Attention-Top-}\alpha\} \tag{13}$$

$$s(q, d) = Match(Q, D'). \tag{14}$$

3.4 End-to-End Retrieval and Query Token Pruning

We introduce how different late-interaction models perform end-to-end retrieval in this section. Specifically, for ColBERT, we conduct a detailed analysis to investigate how the query token retrieves relevant documents. Based on the empirical findings, we further propose **Query Token Pruning (QTP)** methods for ColBERT to reduce retrieval latency.

Due to the requirement of token-level interaction and the utilization of sum-of-max operation in the late interaction model, we cannot directly retrieve the top-k relevant documents from a dense vector index built by toolkits like Faiss [21] which only support inner product and L2 distance. Instead, we need to employ more complex pipelines.

ColBERT employs a two-stage retrieval pipeline: in the first stage it approximates the final relevance score and retrieves a set of candidates, then in the second stage the accurate score will be computed for ranking. In particular, in the first stage, ColBERT retrieves a fixed number of top document tokens from the Faiss index for every query token, then collects the corresponding documents to construct the candidates set. Then, we gather all document tokens for each document in the candidate set to perform a sum-of-max operation and compute the final score between the query and the documents for ranking. To reduce approximation error introduced by different score functions, it is typical to retrieve more documents.

COIL is based on an inverted index, it first calculates scores for each exact matched token in the query, then aggregates all matched tokens to produce a document score and uses heap sort to keep k documents with the highest relevance score. While performing the sum-of-max operation, the inverted index can obtain an accurate score, so the inference stage and training stage can have the same score function.

As we discussed, COIL computes accurate scores while performing retrieval. However, ColBERT first approximates the relevance score and retrieves abundant documents for each query token, resulting in significant retrieval latency. Consequently, it is natural to explore the possibility of pruning certain query tokens for ColBERT to accumulate the retrieval process while minimizing any potential decrease in performance.

The pipeline of Query Token Pruning (QTP) is shown in Figure 4. ColBERT performs Top-k ANNs with every query token, so a large number of candidate documents will be retrieved. The model will spend a relatively long time transferring the embeddings of the candidate document from CPU to GPU to compute the sum-of-max ranking scores. However, Tonellotto and Macdonald [45] have shown that not every query embedding will bring useful documents for ranking. Therefore, performing Top-k ANNs with every query token will impose unnecessary retrieval latency. Tonellotto and Macdonald [45] have proved that only query embeddings with the highest IDF values can be used for effective retrieval, which also confirms the contribution of co-occurrence signals and non-stop words. In addition to this strategy, we try another QTP method based on a self-attention scores matrix of query tokens' embeddings, similar to Equation (10). We argue that, on the one hand, the token with a large value of average self-attention score represents the main semantics of the query, and therefore, would play a major role in retrieving relevant documents. On the other hand, the tokens with low self-attention scores may also convey some salient information that is different from other concrete query tokens but may

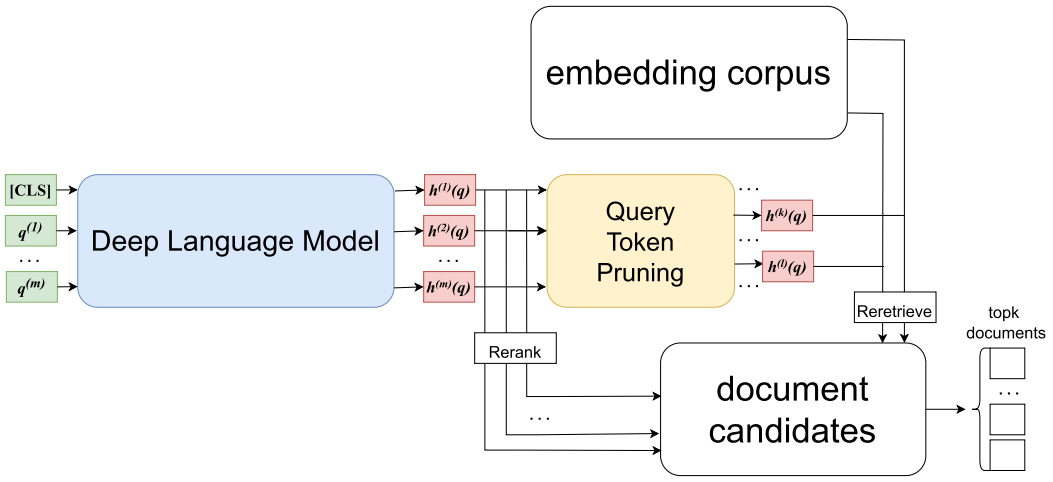


Fig. 4. The pipeline of Query Token Pruning for ColBERT.

be useful in retrieval. Therefore, in order to retain most of the information in the original query, we combine both the tokens with max self-attention scores and those with minimum self-attention scores.

4 EXPERIMENT

In this section, we describe the experimental settings, including the datasets, baselines, and the implementation details of training and evaluation.

4.1 Datasets

First, we conduct experiments on a large-scale ad-hoc retrieval dataset MS MARCO passage [5]. This dataset consists of 8.8M passages and 800K queries, of which 500K queries are labeled with one or a few relevant passages. We train our models, along with the baseline models, on these labeled data and evaluate them in the MS MARCO passage full ranking task and TREC 2019 DL task. The former task contains 6,980 test queries and needs the model to retrieve a candidate set consisting of the top 1,000 passages sorted by relevance from 8.8M passages corpus for each test query. The test set of TREC 2019 DL contains 200 queries, of which 43 have human relevance labels. For evaluation metrics, in addition to using commonly used MRR@10 for the MS MARCO passage ranking task and NDCG@10 for the TREC 2019 DL task, we also report Recall@100 for both two benchmarks. We choose Recall@100 instead of the usually used Recall@1000 because for both two benchmarks. We choose to do this because, for the state-of-the-art transformer-based reranker [32], the latency of reranking 1,000 candidates is not acceptable for the online systems.

Second, we also conduct experiments on BEIR [43], which is a heterogeneous zero-shot retrieval benchmark containing 18 datasets from diverse text retrieval tasks and domains. Most datasets in BEIR do not have a training set, so it is widely used for zero-shot retrieval which can evaluate models' robustness. A common pipeline is training the models on a large supervised dataset such as MS MARCO and then evaluating the models on BEIR, and we also follow this way. Following the previous work [13, 41], we conduct our evaluation only using the 13 publicly available datasets in BEIR. For evaluation metrics of BEIR, we report NDCG@10 to evaluate the gap between the ranking of the retrieved documents and the ideal ranking of documents.

4.2 Baselines

We compare and analyze several typical late-interaction models, including soft-matching models and hard-matching models. Additionally, we encompass sparse retrieval models for comparison, as they can be regarded as a scalar version of hard-matching models.

Soft-matching late-interaction models. ColBERT [23] is a type of accurate and fast late-interaction model using a soft-matching mechanism. ColBERTv2 [41] is an improved version of ColBERT, using an aggressive residual compression mechanism and distillation strategy to improve the index storage and retrieval performance. As these optimization techniques and the pruning methods we proposed are orthogonal, we also include ColBERTv2 as one of the baselines for experimentation.

Hard-matching late-interaction models. COIL [17] is an exact-matching based late-interaction model. COIL incorporates vector representation into the inverted index structure to accomplish exact token matching on the query and document. We also consider COIL-tok, which is COIL without [CLS] embeddings, evaluated to investigate the [CLS] embedding under various document token pruning settings.

Neural sparse retrieval models. DeepImpact [31] is ColBERT followed by a two-layer MLP to convert it into a scalar version and perform an exact match based on an inverted index. uniCOIL [27] is a scalar version of COIL which sets the dimension of COIL's embeddings to 1. Unlike the previous two models mapping each input token to a term impact, SPLADE [13, 14] is an alternative sparse retrieval model that projects dense vectors onto the dimensionality of the vocabulary and aggregates them to obtain term impacts. To ensure a comprehensive comparison, we still consider incorporating it as a baseline but only use a popular version of SPLADE, i.e., SPLADE-max, which uses a max pooling strategy.

4.3 Implementation Details

4.3.1 Model Training. For ColBERT[1], COIL[2], and SPLADE[3], we mainly follow the instructions provided by the official repositories and use the default settings to train our models.

Specifically, when training ColBERT, following Khattab and Zaharia [23], we use the official triples file[4]. Every Query is padded or truncated to 32 tokens and passages are truncated to the first 180 tokens. The final embedding dimension of all tokens is projected to 128 which is different from the default 768 dimensions. As for COIL, passages are truncated to the first 180 tokens. The dimension of [CLS] is 768 and the dimension of the other tokens' embeddings is projected to 32. Following Gao et al. [17], we sample seven hard negatives from BM25's top1000 results for each query to train the model. Additionally, we use the efficient in-batch negative training method [19, 48].

For DeepImpact[5] and uniCOIL[6], different from the original papers, we do not use a generative model to expand the original document so that we can compare the vector version model and scalar version model fairly.

4.3.2 Data Preprocessing. Before indexing, we first compute the document frequency of different tokens in the MS MARCO passage dataset and BEIR dataset. After getting the IDF value of

[1] https://github.com/stanford-futuredata/ColBERT
[2] https://github.com/luyug/COIL
[3] https://github.com/naver/splade
[4] https://microsoft.github.io/msmarco/
[5] https://github.com/DI4IR/SIGIR2021
[6] https://github.com/luyug/COIL/tree/main/uniCOIL

each token, we list them in descending order by their IDF values (i.e., ascending order by their document frequency values) and store the sorted indices in a file. We can directly read this file when using the model to encode corpus which can avoid sorting tokens in real time so that we can save a lot of time. The data preprocessing in this work is mainly for IDF-Top-α pruning although we can directly integrate it into the indexing without an external file, but, that will cause much time waste.

4.3.3 Indexing and Evaluation. Once the training process is completed, we can utilize these models to encode each passage, while employing the proposed pruning methods. For ColBERT and COIL, we do pruning while encoding, and save the remaining embeddings with float16 data type. For ColBERT, we use Faiss to build an IVFPQ index for all embeddings, and the index is trained on a randomly selected 5% sample of the document embeddings, while COIL is based on a special inverted index of embeddings. For all sparse model baselines, we use the Pyserini toolkit [28] to build the inverted index and perform the exact match retrieval. It should be noted that due to the different model architectures of SPLADE, we employ a unique approach of pruning the output of the MLM head before performing pooling operations.

Training and evaluating are conducted on a server equipped with 2 RTX 3090 GPUs and each of them has 24GiBs of GPU memory.

5 RESULTS

With the experimental results in the MS MARCO passage ranking task, TREC 2019 DL task, and BEIR dataset, we investigate the effectiveness and efficiency of different document token pruning methods. We also conduct some ablation studies to show how the effectiveness and efficiency change with parameters and settings of DTP, respectively. Besides, we also show the result of QTP on ColBERT on MS MARCO.

5.1 Document Token Pruning

5.1.1 Results on MS MARCO and TREC DL. For MS MARCO, we show the effectiveness of different models after different DTP methods in Table 2 that keep 50% and 75% token embeddings, and the results reveal that all late-interaction models and sparse models are compatible with the three DTP methods. In the result cells, the result of $\alpha = 0.5$ is on the left of "/", and the result of $\alpha = 0.75$ is on the right. This subsection mainly focuses on the effectiveness of the $\alpha = 75\%$ version, and we will discuss the effectiveness of different remaining ratios in the later subsection.

Specifically, for most DTP methods, the performance decrease of both soft-matching and hard-matching late-interaction models is minimal when keeping 75% token embeddings. Statistical tests on the results from MS MARCO and TREC DL indicate the equivalence between the pruned approach and the unpruned setting.

For ColBERT, the weakest DTP method for MRR@10 in the MS MARCO passage ranking task is the Attention-Top pruning method, which loses 1.7% in MRR@10. In contrast, the other two DTP methods only have a loss of less than 1%. IDF-Top pruning method has the best performance for Recall@100 in the MS MARCO Passage Retrieval task which can have no loss. For pruning ColBERT on the TREC DL task, the First-Top DTP method outperforms Attention-Top and IDF-Top in NDCG@10, and the IDF-Top DTP method performs best in Recall@100. As for ColBERTv2, the IDF-Top DTP method results in a notable decline in effectiveness, while the other two methods have a little difference compared to the baseline.

COIL is a strong hybrid model that leverages [CLS] embedding to alleviate vocabulary mismatch problems and we always keep the [CLS] embedding when we perform DTP methods. As we can see, COIL is very robust to IDF-Top DTP and Attention-Top DTP. COIL after IDF-Top DTP

Table 2. Results of Document Pruning Methods with Remaining Ratio $\alpha = 0.5/0.75$ on MS MARCO Passage Ranking Task and TREC 2019 Deep Learning Track

Models	MS MARCO Dev		TREC DL 19	
	MRR@10	Recall@100	NDCG@10	Recall@100
ColBERT	.363	.874	.711	.460
ColBERT + First	.348*/**.360***	.839*/.866*	.699 /**.707**	.418 /.444
ColBERT + IDF-Top	.341*/**.360***	.860 /**.875***	.672 /.695	.442 /**.457**
ColBERT + Attention-Top	.325*/.356*	.832 /.870*	.667 /.704	.405 /.453
ColBERTv2	.397	.914	.749	.562
ColBERTv2 + First	.377*/**.393***	.863 /.901*	.723 /.740	.511 /.550
ColBERTv2 + IDF-Top	.326 /.360*	.838 /.878*	.702 /.715	.471 /.519
ColBERTv2 + Attention-Top	.360*/.390*	.884*/**.912***	.720 /**.747**	.543 /**.566**
COIL-full	.352	.874	.693	.505
COIL-full + First	.338*/.348*	.835*/.862*	.687 /.692	.462 /.492
COIL-full + IDF-Top	.336*/**.350***	.844*/**.874***	.651 /.693	.462 /**.509**
COIL-full + Attention-Top	.345*/**.350***	.870*/.873*	.689 /**.694**	.504 /.507
COIL-tok	.340	.851	.676	.500
COIL-tok + First	.321*/.338*	.800 /.835*	.659 /.662	.436 /.482
COIL-tok + IDF-Top	.315*/.340*	.824*/.851*	.639 /**.676**	.470 /**.501**
COIL-tok + Attention-Top	.336*/**.342***	.847*/**.853***	.675 /**.676**	.496 /**.501**
DeepImpact	.285	.770	.547	.413
DeepImpact + First	.257*/.279*	.684 /.741*	.511 /.528	.345 /.388
DeepImpact + IDF-Top	.217 /**.283***	.642 /**.766***	.536 /**.563**	.360 /**.430**
DeepImpact + Attention-Top	.199 /.232	.551 /.639	.449 /.472	.285 /.321
uniCOIL	.319	.809	.636	.479
uniCOIL + First	.297*/.314*	.746 /.790*	.624 /.630	.430 /.465
uniCOIL + IDF-Top	.275 /.312*	.757 /.803*	.578 /.632	.460 /**.482**
uniCOIL + Attention-Top	.313*/**.319***	.802*/**.809***	**.645** /.638	.473 /.476
SPLADE	.340	.871	.682	.487
SPLADE + First	.320*/.336*	.814 /.852*	.665 /**.681**	.447 /.469
SPLADE + IDF-Top	.203 /.272	.614 /.749	.530 /.611	.317 /.397
SPLADE + Attention-Top	.303*/**.338***	.802 /**.865***	.642 /**.681**	.445 /**.484**

The result of $\alpha = 0.5$ is on the left of "/", while the result of $\alpha = 0.75$ is on the right. The bolded numbers represent the optimal results achieved by applying pruning techniques in each model. * indicates equivalent to the unpruned settings at $p < 0.05$ level using the TOST test [42] with the equivalence bound $-\Delta_L = -0.05$ and $\Delta_U = 0.05$.

and Attention-Top has comparable and even better effectiveness than original COIL on both tasks which means there may be some noise in the exact matching mechanism and filtering the noise brings some gain. COIL-tok is a modified version of COIL that does not have [CLS] embedding. The results show that COIL-tok has a consistent decrease than COIL, which can prove that [CLS] can alleviate the vocabulary mismatch problem by introducing [CLS] semantic matching. Similar to COIL, COIL-tok after IDF-Top DTP and Attention-Top have equivalent and even slightly superior effectiveness than the original COIL-tok on both tasks, indicating that there may be some noise in the exact matching process and DTP can filter the noise then result in some gain.

However, the sparse retrieval models exhibit significantly lower robustness to pruning, with some pruning methods resulting in a substantial performance decrease. DeepImpact has a large loss when compared to its vector counterpart (i.e., ColBERT), indicating that vectors have stronger

representation capabilities. Different from ColBERT, the Attention-Top DTP method has a significant loss compared to the unpruned setting, and the IDF-Top DTP method on DeepImpact performs best in both MS MARCO and TREC. These differences between ColBERT and DeepImpact may come from different representations (i.e., vector vs. scalar) and different matching mechanisms (i.e., soft-matching vs. hard-matching). The uniCOIL model can be seen as a scalar version of COIL-tok, and the aim of evaluating uniCOIL is to compare the robustness of scalar-version models and vector-version models on DTP based on an inverted index. The results suggest that uniCOIL is weaker than COIL-tok when we employ DTP on uniCOIL. Specifically, there is a gap of 2.2% in MRR@10 when we employ IDF-top on uniCOIl while there is no loss in Recall@100 when we employ IDF-top on COIL-tok. As for SPLADE, due to its different architecture, it has a different behavior compared to DeepImpact and uniCOIL. The IDF-Top DTP method results in a significant loss when applied to SPLADE. We suggest that this may be attributed to the loss of information due to pruning with IDF before aggregation. In contrast, because the other two DTP methods perform better, it can be inferred that the projection of tokens positioned towards the rear (or with lower attention scores) provides less informative content.

The main results on MS MARCO show that most late-interaction models are robust to most document token pruning methods, that is to say, these models can save a lot of storage space by DTP with little performance loss. More specifically, the soft-matching late-interaction vector model (ColBERT and ColBERTv2) and the hard-matching late-interaction vector exact model (COIL and COIL-tok) are particularly resistant to DTP. However, the scalar version models (DeepImpact and uniCOIL) are more vulnerable. We believe this is because vector representation models contain some duplicate information and are hence more robust to DTP. Additionally, the Attention-Top and IDF-Top pruning methods seem to be more robust across all different models.

5.1.2 Results on BEIR. In addition to MS MARCO and TREC DL 2019, we also conduct experiments using these DTP methods for the above models on BEIR to evaluate the out-of-domain robustness. Table 3 shows the effectiveness of different models after different DTP methods. We only report the results that contain 75% token embeddings due to the paper length.

Specifically, when we keep 75% token embeddings, for ColBERT, the worst DTP method is the Attention-Top pruning method according to the average value of these datasets, which is 0.391 in NDCG@10 and loses 7.3% performance compared to the original ColBERT, while other two DTP methods are both slightly better than the not pruned model, this means there are some harmful data which mislead the original model. More specifically, there are five datasets (i.e., FQ, NF, SF, TC, and TO) that pruning can improve the performance, like the effectiveness after First-Top Pruning can outperform 0.27 in NDCG@10 on Trec-Covid (i.e., TC) dataset. The First pruning method only has the four best results compared to the other two pruning methods, which is the least one among the three, while it also has a good performance in other non-best datasets, so it is more robust and gets the best average performance. The Attention-Top pruning has the six best results among these three pruning methods, however, it has the worst average performance because of its inferior results in several datasets. Compared to the First-Top and Attention-Top pruning methods, the results of the IDF-Top method placed in the middle which is comparable with the First pruning method, and the average value is better than the non-pruned ColBERT.

Different from the results on MS MARCO, we found that COIL has better average performance on tested BEIR datasets than ColBERT, which may mean COIL has more robust representations and the exact matches based on the traditional inverted index also improves the robustness so it can outperform ColBERT. However, COIL is sensitive about pruning on BEIR tasks. As we can see in the average results, for COIL-full, the best pruning method is the Attention-Top, but it still loses 11.1% NDCG@10 compared to the original COIL. Surprisingly, COIL-tok performs more

Table 3. NDCG@10 Results of Document Pruning Methods with Remaining Ratio $\alpha = 0.75$ on BEIR Dataset

	AR	CF	DB	FE	FQ	HQ	NF	NQ	QU	SD	SF	TC	TO	Avg.
ColBERT	.398	.148	.377	.747	.299	.570	.291	.516	.841	.145	.604	.600	.205	.422
First	.387	.138	.367	.717	.289	.527	.290	**.499**	.788	**.141**	.580	**.627**	**.211**	**.428**
IDF-Top	.390	.138	.371	.719	**.299**	.534	**.293**	.399	**.834**	**.141**	**.612**	.600	**.211**	.426
Att-Top	**.391**	**.146**	**.373**	**.740**	.192	**.561**	.190	.385	.732	.098	.497	.443	.203	.391
ColBERTv2	.463	.176	.446	.785	.356	.667	.338	.562	.852	.154	.693	.738	.263	.499
First	.444	.155	.440	.730	.324	.628	.324	.532	.787	.152	.616	**.721**	.261	.470
IDF-Top	**.449**	**.170**	**.451**	**.753**	**.340**	**.673**	**.326**	**.550**	**.853**	**.155**	**.648**	.717	**.262**	**.488**
Att-Top	.445	.167	.444	.739	.336	.658	**.326**	.547	.844	**.155**	.640	.700	**.262**	.482
COIL	.372	.149	.393	.751	.268	.593	.307	.598	.827	.149	.555	.631	.240	.449
First	.327	**.079**	.263	.683	.219	.536	.306	.451	.557	**.138**	.518	.504	.225	.370
IDF-Top	**.342**	.072	.246	.605	**.234**	.473	**.319**	.444	.498	.135	.542	.490	.219	.355
Att-Top	.336	**.079**	**.270**	**.753**	.232	**.557**	.311	**.471**	**.754**	.135	**.545**	**.517**	**.233**	**.399**
COIL-tok	.351	.124	.370	.725	.245	.568	.306	.468	.796	.144	.547	.582	.244	.421
First	.333	.124	.356	.709	.234	.539	.302	.444	.621	.142	.533	**.580**	.235	.396
IDF-Top	**.350**	.109	.327	.649	**.247**	.477	**.307**	.434	.555	.143	**.550**	.568	**.260**	.383
Att-Top	**.350**	**.125**	**.368**	**.724**	.241	**.562**	.305	**.465**	**.791**	**.144**	.546	.575	.256	**.419**
DeepImpact	.375	.043	.167	.027	.108	.092	.232	.329	.248	.090	.390	.336	.195	.202
First	**.384**	.040	.164	**.026**	.099	.082	.227	.311	.160	**.091**	.364	.322	**.190**	.189
IDF-Top	.372	**.060**	**.165**	.021	**.121**	.075	**.287**	.312	**.286**	.080	.362	.329	.152	**.202**
Att-Top	**.384**	.039	**.165**	.025	.102	**.083**	.232	**.315**	.231	.089	**.382**	**.332**	.188	.197
uniCOIL	.330	.107	.340	.727	.226	.550	.302	.413	.650	.136	.527	.561	.241	.393
First	.315	.107	.330	.709	.213	.525	.298	.391	.435	.134	.506	.564	.239	.367
IDF-Top	**.334**	.101	.302	.643	.226	.457	.302	.379	.406	.136	.527	.549	**.254**	.355
Att-Top	.329	**.109**	**.340**	**.727**	**.226**	**.547**	**.302**	**.413**	**.646**	**.137**	**.529**	**.566**	.241	**.393**
SPLADE	.439	.119	.366	.730	.287	.636	.313	.469	.835	.145	.628	.673	.256	.456
First	.454	.154	.366	.678	.235	.610	.303	.439	.693	.144	.555	.632	.184	.419
IDF-Top	**.479**	.159	**.368**	.685	**.250**	**.642**	**.308**	.451	**.829**	**.148**	**.581**	.631	.199	**.441**
Att-Top	.477	**.161**	**.368**	**.694**	.248	.639	.306	**.454**	.635	.146	.580	**.634**	**.220**	.428

(AR: ArguAna. CF: Climate-Fever. DB: DBPedia. FE: Fever. FQ: FiQA-2018. HQ: HotpotQA. NF: NFCorpus. NQ: Natural Questions. QU: Quora. SD: SCIDOCS. SF: Scifact. TC: TREC-COVID. TO: Touché-2020.)

robustly when applied pruning compared to COIL. Within COIL-tok, Attention-Top is the best pruning method and only has a 0.5% NDCG@10 loss, while the other two pruning methods have much worse effectiveness than the original COIL-tok, however, still better than COIL on the same settings. Regarding the differences between the two models, we suggest this may be attributed to the fact that for COIL, the [CLS] embedding plays a main role after pruning. However, existing research has indicated that the zero-shot performance of solely using the [CLS] embedding from fine-tuned BERT is subpar [43]. Conversely, COIL-tok relies entirely on the exact match of token embeddings and this approach may potentially yield greater robustness in a zero-shot setting.

As for the sparse models, the best pruning method is IDF-Top for both DeepImpact and SPLADE and is Attention-Top for uniCOIL. DeepImpact and uniCOIL have a large loss compared to ColBERT and COIL, indicating that vectors have stronger representation capabilities no matter whether in the in-domain dataset or out-domain dataset. However, SPLADE shows a comparable performance, which may derive from the different architecture.

Additionally, we analyze the pruning methods across the different types of datasets. Following Santhanam et al. [41], we classify the 13 datasets into two types: search tasks and semantic relatedness tasks. The first type of dataset includes DB, FQ, HQ, NF, NQ, and TC, and the second type includes AR, CF, FE, QU, SD, and SF. The datasets of search tasks are underlined in Table 3.

We count the number of datasets in which different pruning methods achieved the best performance on the two types of datasets, and the results are shown in Table 4. We can see that the First

Table 4. The Number of Datasets in which Different Pruning Methods Achieved the Best Performance on the Two Types of Datasets

DTP Method	ColBERT		ColBERTv2		COIL		COIL-tok		DeepImpact		uniCOIL		SPLADE	
	S	SR	S	SR	S	SR	S	SR	S	SR	S	SR	S	SR
First	3	1	1	0	0	2	1	0	1	3	0	0	0	0
IDF	3	3	5	7	2	1	3	2	3	2	1	1	4	4
Attention	2	3	2	1	5	4	3	5	4	2	6	5	4	2

S indicates search tasks including 6 datasets and **SR** indicates semantic relatedness tasks including 7 datasets. Note that since there are cases where the results of both methods are the same, the sum of each column may exceed the number of datasets.

pruning is the weakest, which is consistent with the results on MS MARCO. Comparing the other two pruning methods, we find that for soft-matching models (i.e., ColBERT and ColBERTv2), the IDF-Top pruning seems to be better, while Attention-Top pruning achieves better performance for other models based on exact match. This result is surprising because IDF value is first designed for the inverted index which is also based on exact match. We think this may be attributed to the difference in the distribution between IDF values and learned impact or embeddings. However, there is no obvious conclusion can be drawn about which pruning method is preferred for different types of datasets.

The main results on BEIR show some different results compared to MS MARCO. Specifically, COIL-full is the best model on the BEIR task, and it reveals that COIL-full has more robust representations and the exact matches based on the traditional inverted index also improve the robustness. Besides, the First-Top DTP method on ColBERT gives the best performance which means we can use the First pruning method to filter some harmful information. The best DTP method for all three COIL-related models is Attention-Top and it may mean this method can extract better representations more effectively.

5.2 Query Token Pruning

We show the results of query token pruning in Table 5. The motivation for query token pruning is to speed up retrieval without significantly affecting performance. Therefore, we report the performance and latency of the original ColBERT model without query token pruning and with query token pruning. In addition to IDF-Top QTP used in [45], we also evaluate the attention-based query token pruning method which is only based on the information of the given query.

For the original ColBERT, each query retrieves 5,282 documents on average, while after IDF QTP, the average number of documents retrieved by each query is reduced to 1,972. This reduces retrieval time from 125ms to 86ms, and the speed is increased by 1.45× times with no performance loss on MRR@10. After Attention QTP, the average number of documents retrieved by each query is reduced to 2,731. This leads to a reduction of retrieval time from 125ms to 98ms, and the speed is increased by 1.27× times and also with no performance loss on MRR@10.

The results can be comprehended with relative ease: ColBERT employs a two-step retrieval process. Initially, it retrieves an approximate set of candidates, followed by utilizing the sum-of-max operation to accurately compute the relevance score and rerank the candidates accordingly. Our QTP methods only prune the query tokens during the initial stage, while employing all token embeddings to compute the final score. Consequently, once the relevant document is retrieved as a candidate, its final score will be identical to that of the no QTP settings. Thus, these findings

Table 5. Results of Query Tokens Pruning on MS MARCO Dev. A.R.D. means Average Retrieved Documents per Query

Models	MRR@10	Recall@100	A.R.D.	Latency (ms)
ColBERT	0.363	0.874	5282	125
+ IDF [45]	0.363*	0.860*	1972	86
+ Attention	0.363*	0.868*	2731	98
ColBERT-First-50%	0.348	0.838	5856	94
+ IDF [45]	0.348*	0.829*	2210	72
+ Attention	0.348*	0.836*	3033	77

Latency is the average of all query latency. * indicates equivalent to the unpruned settings at $p < 0.05$ level using the TOST test [42] with the equivalence bound $-\Delta_L = -0.05$ and $\Delta_U = 0.05$.

indicate that when ColBERT performs well (at the very least, for queries with MRR@10 > 0), the addition of QTP does not lead to a performance decrease on MRR. However, it is worth noting that QTP may result in a decrease in Recall@100 as the average number of retrieved documents is substantially smaller than the no QTP settings.

Furthermore, we have conducted additional experiments to confirm whether the application of QTP exhibits similar behavior when on the index constructed using the DTP method. We choose the pruned ColBERT index established using the First DTP method because it is one of the best DTP methods, while the remaining ratio α is set to 50%, denoted as **ColBERT-First-50%**. As shown in Table 5, the results of QTP on the pruned index are similar to those on the full index. Specifically, IDF QTP and Attention QTP both lead to a reduction in retrieval time and increase the retrieval speed by 1.31× times and 1.22× times, respectively, with little performance loss.

In summary, DTP actually not only reduces storage overhead but also reduces latency because there are fewer embeddings that need the model to traverse. Additionally, QTP can further improve the efficiency of the models with little loss of retrieval effectiveness.

5.3 Ablation and Analysis

5.3.1 *Analysis of Remaining Ratio for Document Token Pruning.*

Analysis on MS MARCO. We report the results of the ablation study in Figure 5. We did a series of ablation experiments on the MS MARCO to explore the impact of the remaining ratio of tokens on the final performance of different late-interaction models. We prune on different remaining ratios and verify the effectiveness of different models. In general, the result shows that the performance of models is positively correlated with the number of representations, while the correlation is not linear which means we can further make an effectiveness-storage trade-off, for example, we can use 50% or 75% tokens to represent the document with only a little performance loss. However, if we use only 25% tokens, the effectiveness of all models will have a relatively obvious decrease.

Specifically, for the results of three DTP methods on soft-matching late-interaction models (i.e., ColBERT and ColBERTv2) shown in Figure 5(a) and Figure 5(b), the First pruning method is the best for different remaining ratios. In contrast, the IDF-Top pruning and Attention-Top pruning perform differently on the two models. It's worth noting that the IDF-Top pruning method leads to a significant performance loss even if we keep 75% embeddings.

As for the results of three DTP methods on hard-matching late-interaction models (i.e., COIL and COIL-tok) shown in Figure 5(c) and Figure 5(d), except for IDF-Top pruning, the other two pruning methods show a better robustness, even though for keeping only 25% embeddings. Concretely, the Attention-Top method outperforms the First method, while the First method outperforms the IDF-Top method. We also notice that the curve of COIL-tok decreases faster as the remaining ratio decreases, we posit that it's because of the lack of [CLS] embedding.

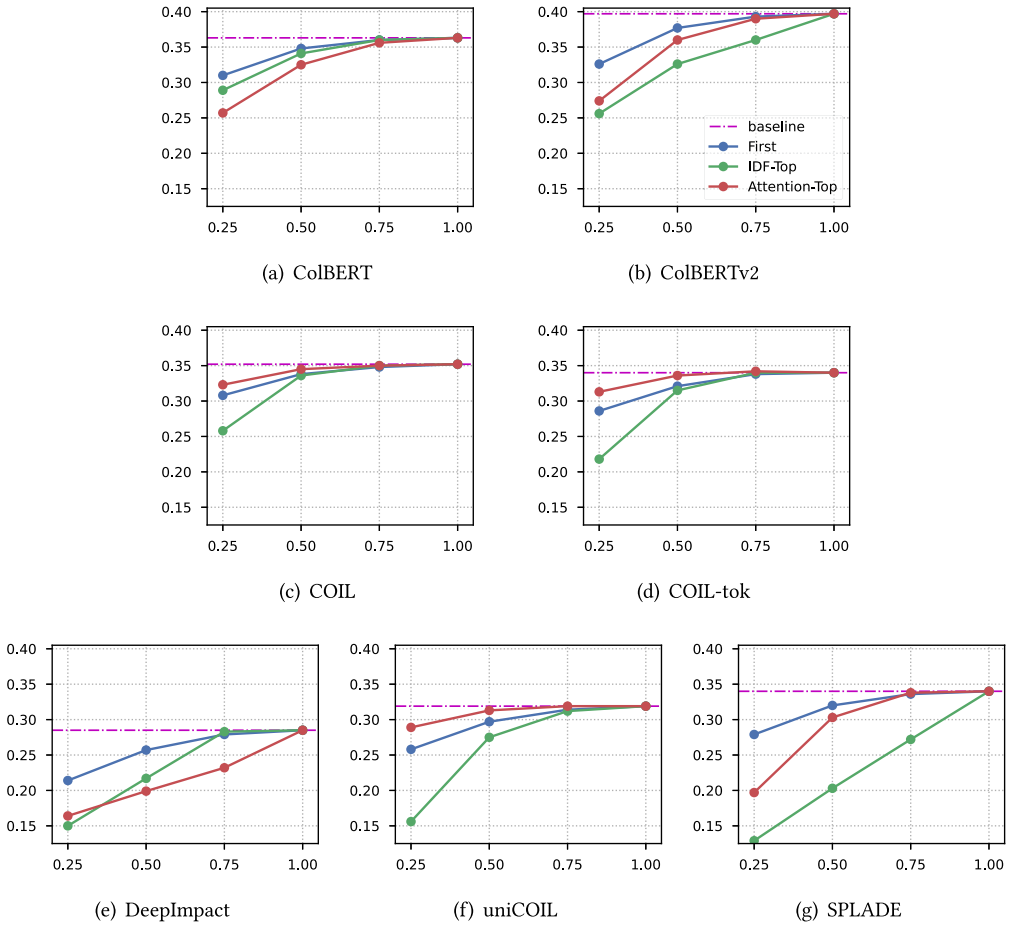
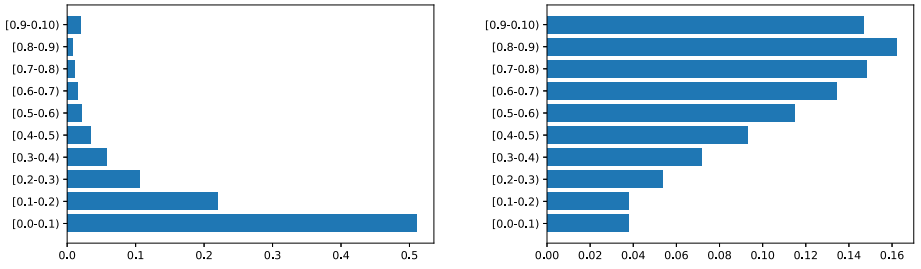


Fig. 5. Effectiveness of different DTP methods of different models on MSMARCO Dev. The dashed line denotes the performance of models without DTP. The x-axis is the remaining ratio α and the y-axis of each sub-figure is the MRR@10 metric.

Sparse retrieval models (i.e., DeepImpact, uniCOIL, and SPLADE) hold a similar conclusion to hard-matching models, that is, the First pruning method and the Attention-Top pruning method still have good robustness while we remain more tokens (i.e., $\alpha = 0.5/0.75$), and the only exception is the Attention-Top on DeepImpact. We guess this is because DeepImpact was not trained enough. However, the curve of IDF-Top pruning on all three models decreases fast as the remaining ratio decreases to 25%, showing that it's not enough to capture all information of a passage in an inverted index using only a small proportion of words that have the highest IDF values.

In summary, for various models, the First pruning method demonstrates relatively robust performance. In other words, as the proportion of retained tokens gradually decreases, the decline in retrieval performance is not as rapid, and it is possible to achieve performance comparable to that of unpruned settings when the remaining rate is 75%. The other two pruning methods have different performances on different models, but the Attention-Top pruning method shows a more promising robustness.

Additionally, we also do a comparison across all models, and we found that ColBERT only outperforms DeepImpact and SPLADE and is inferior to the other three models on MS MARCO Dev



(a) COIL-tok (b) ColBERT

Fig. 6. The self-attention distribution of COIL-tok and ColBERT. We use softmax to scale all values to [0,1] and divide them into ten bins.

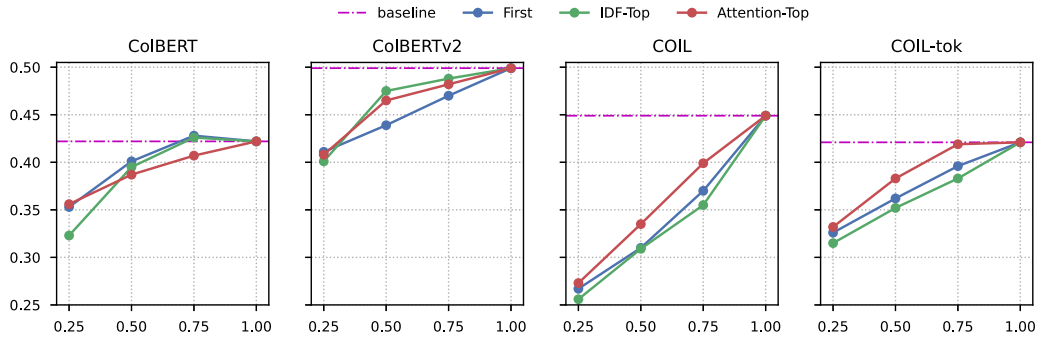
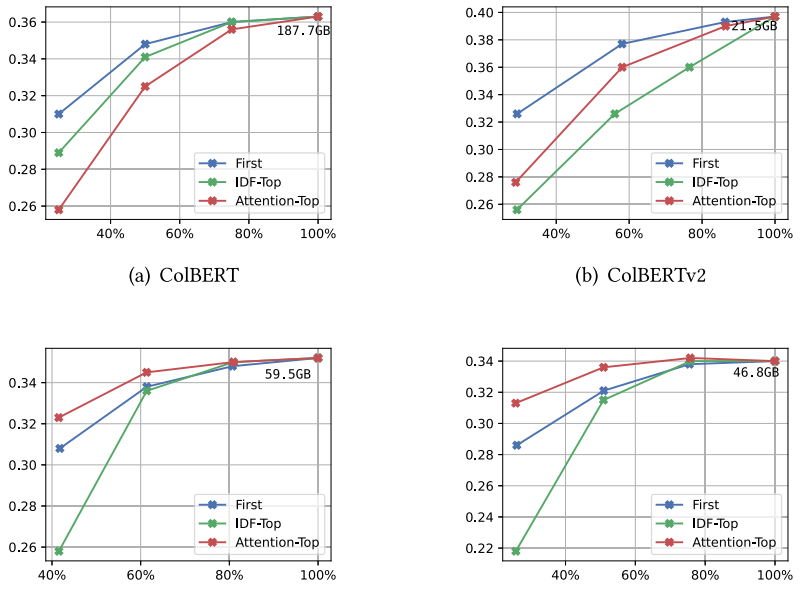


Fig. 7. Average effectiveness of different DTP methods on different models on BEIR. The dashed line denotes the performance of late-interaction models without DTP. The x-axis is the remaining ratio α and the y-axis of each sub-figure is the average NDCG@10 metric on BEIR.

when we only keep 25% token embeddings using Attention-Top-α pruning which is shown by Figure 5. And if we keep more embeddings such as 75%, the effectiveness of ColBERT will reach the top place. The reason may be the distribution of ColBERT's representation is more compact than other models, so the Attention-Top-25% is somewhat redundant while embeddings of other models may be more scattered, so the Attention-Top-25% embeddings of them are more informative. To confirm this, we visualize the distribution of COIL-tok and ColBERT self-attention score which is shown in Figure 6. The figure tells us that the distribution of ColBERT's representation is more compact than COIL-tok's. Specifically, ColBERT's self-attention score is largely concentrated in areas greater than 0.5 whereas most COIL-tok's self-attention score is less than 0.2 which means it is scattered in the representation space. As a result, the attention-top-25% embeddings of COIL-tok can get comparable performance with COIL-tok without pruning, however, the attention-top-25% embeddings of ColBERT have a significant loss in most metrics.

Analysis on BEIR task. In addition to experiments on large-scale supervised datasets MS MARCO, we also conduct experiments on BEIR, and the results are shown in Figure 7. From the results, we can find that the Attention-Top pruning method always performs well across the late-interaction models when we keep 75% embeddings, especially for COIL and COIL-tok on which it outperforms the other two methods. The First pruning and IDF-Top pruning perform differently on soft-matching models and hard-matching models, specifically, First pruning is always better than IDF-Top pruning on COIL and COIL-tok, while each has its own winner or loser on the ColBERT and



(a) ColBERT (b) ColBERTv2

(c) COIL (d) COIL-tok

Fig. 8. Index size vs. MRR@10 on MSMARCO Dev for the three pruning methods. The number on the figure is the size of the full index. The x-axis denotes the percentage of the pruned index size in relation to the full index size, while the y-axis represents the MRR@10.

ColBERTv2. However, if we only keep 25% embeddings, all three pruning methods will increase fast and lead to a significant NDCG@10 decrease for all models.

Analysis of the Trade-off between Effectiveness and Efficiency. Figure 8 shows a trade-off between the effectiveness (i.e., the retrieval performance on MS MARCO) and the storage efficiency (i.e., the index size). As the figure shows, we can achieve a good trade-off by keeping 75% of token embeddings, reducing a lot of storage overhead yet with little performance loss.

5.3.2 Analysis of Query Token Pruning. For the method of pruning query tokens based on attention, we investigate various configurations. Specifically, we use three ways of embedding query tokens to retrieve documents: tokens with the minimum 3 attention scores, tokens with the maximum 3 attention scores, and a combination of the first two. The experiment results are shown in Table 6.

For the results, if we only use the three tokens with maximum attention scores, there will be a large performance loss compared to the original ones. Observing the much smaller average number of retrieved documents, we believe that these tokens do not capture the diversity distribution of the original query representations. Moreover, we also use the three token embeddings with minimum attention scores to retrieve the document candidates. Surprisingly, as the result shows, we can use these token embeddings to retrieve most of the useful documents that heavily influence the final ranking performance. We will qualitatively analyze this phenomenon later.

Moreover, we combine the minimum and maximum attention query tokens to retrieve more documents, the results show that it takes only a little improvement but more latency.

5.3.3 Qualitative Analysis and Case Study. In order to gain a deeper comprehension of the pruning process, we qualitatively analyze the document tokens stored by ColBERT and COIL after

Table 6. Results of Query Tokens Pruning on MS MARCO Dev. A.R.D. Means Average Retrieved Documents per Query

Model	MRR@10	Recall@100	A.R.D.	Latency (ms)
ColBERT	0.363	0.874	5282	125
+ MinA-3	0.363	0.856	1952	99
+ MaxA-3	0.340	0.631	1527	85
+ MinA-3+MaxA-3	0.363	0.868	2731	98
ColBERT-First-50%	0.348	0.836	5856	94
+ MinA-3	0.347	0.822	2191	72
+ MaxA-3	0.326	0.608	1664	69
+ MinA-3+MaxA-3	0.348	0.836	3033	78

Latency is the average of all query latency. Min(Max)A-k means using k tokens of minimum (maximum) attention scores to retrieve candidate documents.

Table 7. Visualization of Document Token Attention Scores

	top 25% top 50% top 75% top 100%
Query	**[CLS] [Q] average income for north salem new york**
Document (ColBERT)	teacher **north salem** , ny salary . teacher **north salem** , ny **average** salary is $ 74 , 97 ##3 , median salary is $ 80 , 000 with a salary range from $ 49 , 92 ##0 to $ 86 , 40 ##6 . teacher **north salem** , ny salaries are collected from government agencies and companies . each salary is associated with a real job position .
Document (COIL)	teacher **north salem** , ny salary . teacher **north salem** , ny **average** salary is $ 74 , 97 ##3 , median salary is $ 80 , 000 with a salary range from $ 49 , 92 ##0 to $ 86 , 40 ##6 . teacher **north salem** , ny salaries are collected from government agencies and companies . each salary is associated with a real job position .

Red shades reflect the tokens' relative ranking of attention scores, the darker the color, the higher the ranking. Query tokens are bold. The query tokens marked blue are the three tokens with the highest attention scores calculated using ColBERT, and the ones marked orange are the three lowest.

pruning. Specifically, we only analyze the Attention-Top pruning method, as the results of the other two pruning methods are easy to obtain. In Table 7, we select one typical example to demonstrate how the attention scores of tokens in documents are distributed.

One of the important findings is that COIL gives higher attention scores to tokens that appear in both the query and document (i.e., "*north*" and "*salem*" in this example), but ColBERT does the opposite. This may explain the phenomenon that COIL performs well for Attention-Top pruning even though only 25% tokens remained but ColBERT performs poorly, as our analysis shows that the contribution of co-concurrence is important for ColBERT (reference Table 1).

Another observation is that both COIL and ColBERT give some stop words high attention scores. However, our previous experimental results (reference Table 1 and the part of IDF-Top pruning on Table 2) indicate that stop-words are not essential in the matching mechanism of late-interaction models. This situation may be one of the factors limiting the robustness of Attention-Top pruning.

Additionally, we compute the attention scores of the query tokens for ColBERT. We find that this distribution is similar to the attention score of tokens in the document, that is, the special tokens and stop words hold the highest attention score but the tokens with co-concurrence signals hold the lowest attention score. This finding may potentially explain the better retrieval performance observed in QTP for tokens with lower attention scores.

6 DISCUSSION

In this section, we will discuss the results presented in Section 5, expounding upon several key discoveries that address the research questions proposed in Section 1.

Regarding **RQ1** (*How do the late-interaction models work and perform matching at the token level?*), we conducted experiments to explore the matching mechanisms of the late-interaction models, specifically to explore what document tokens will be selected and how much they contribute to the final relevance score. Our findings indicate that both soft-matching models such as ColBERT, and hard-matching models such as COIL, heavily rely on important tokens for computing relevance scores. Specifically, tokens that are positioned towards the beginning of a paragraph or have a higher IDF value contribute more significantly to the calculation of the final relevance score.

Inspired by the similarity between late-interaction models and traditional retrieval models, we propose to prune document tokens in order to achieve better efficiency while minimizing the loss of effectiveness. We conduct comprehensive experiments to examine the performance of various pruning techniques when employed on distinct models (**RQ2** and **RQ3**).

Regarding **RQ2** (*What are the differences in performance among various models when applying pruning?*), we observe that the hard-matching late-interaction models, i.e., COIL and COIL-tok, are the most robust models when pruned on the supervised dataset. They can consistently achieve acceptable retrieval performance with different pruning methods and remaining ratios. The soft-matching late-interaction models, i.e., ColBERT and ColBERTv2, come next in terms of stability, while the sparse retrieval model is the least stable. We suggest this is because the hard-matching late-interaction models leverage the strengths of both dense and sparse retrieval techniques by utilizing the dense representations and exact match to compute the relevance score, thus achieving a better robustness when applying pruning although their original effectiveness is worse than the soft-matching late-interaction models. However, for experiments in the zero-shot setting, all models exhibited a decline in robustness to token pruning.

Regarding **RQ3** (*How do different pruning methods and pruning ratios affect the effectiveness and efficiency of retrieval models?*), when we employ DTP and set the remaining ratio to a higher number, e.g., 50% or 75%, the First pruning is a simple but effective method across various models, the IDF-Top pruning shows unstable performance on different models, and the Attention-Top pruning method shows promising robustness on the supervised dataset but performs poorly on the zero-shot settings. This means that we can achieve a good trade-off between effectiveness and efficiency by pruning a small proportion of unimportant tokens. However, for all three methods, as the remaining ratio decreases to a smaller value, specifically 25%, the retrieval performance has a significant loss, which might be unacceptable. Further, the QTP methods can reduce the retrieval latency with little loss of performance for ColBERT.

7 CONCLUSION

This paper studies the late-interaction models when we employ different document token pruning methods. Firstly, we analyze late interaction models (i.e., ColBERT and COIL) from more traditional perspectives such as co-occurrence matches and term importance. Through the analysis, we further reveal the similarity between traditional models and late interaction models. Therefore, we are inspired by the match pattern of the traditional information retrieval models and study whether we can safely prune some tokens that are similar to the stop words in traditional retrieval models and which pruning method can achieve the best trade-off between effectiveness and efficiency. Results on the MS MARCO and BEIR show that we can reduce storage overhead with little performance loss when we use these late-interaction models. We further investigate the inference time, and query token pruning methods which skip some unnecessary ANNs with query tokens. Experimental results on the MS MARCO show that the query pruning method can speed up the retrieval with little loss of performance.

We acknowledge some potential limitations of this work. First of all, our proposed pruning methods are based on some simple yet effective heuristics, such as retaining the tokens with high IDF values or high attention sore. Therefore, they may not be the optimal pruning method as Attention-Top and IDF-Top exhibit different distributions on tokens. Secondly, when we conduct experiments on sparse models, we find that these pruning methods may not be as effective as directly removing lower-weighted tokens from the document. This indicates that the pruning strategies of sparse retrieval models could be rather different from those of typical late-interaction models with dense representations. We leave a more thorough investigation of the matching mechanism and effective pruning strategies for the sparse retrieval models to future work as they are out of the scope of this paper.

REFERENCES

[1] Soner Altin, Ricardo Baeza-Yates, and B. Barla Cambazoglu. 2020. Pre-indexing pruning strategies. In *International Symposium on String Processing and Information Retrieval*. Springer, 177–193.

[2] Ismail S. Altingovde, Rifat Ozcan, and Özgür Ulusoy. 2012. Static index pruning in web search engines: Combining term and document popularities with query views. *ACM Transactions on Information Systems (TOIS)* 30, 1 (2012), 1–28.

[3] Ricardo Baeza-Yates, Aristides Gionis, Flavio P. Junqueira, Vanessa Murdock, Vassilis Plachouras, and Fabrizio Silvestri. 2008. Design trade-offs for search engine caching. *ACM Transactions on the Web (TWEB)* 2, 4 (2008), 1–28.

[4] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. SparTerm: Learning term-based sparse representation for fast text retrieval. *arXiv preprint arXiv:2010.00768* (2020).

[5] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A human generated machine reading comprehension dataset. arXiv preprint arXiv:1611.09268 (2018).

[6] Roi Blanco and Alvaro Barreiro. 2007. Static pruning of terms in inverted files. In *Advances in Information Retrieval: 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007. Proceedings 29*. Springer, 64–75.

[7] Roi Blanco and Alvaro Barreiro. 2010. Probabilistic static pruning of inverted files. *ACM Transactions on Information Systems (TOIS)* 28, 1 (2010), 1–33.

[8] Stefan Büttcher and Charles L. A. Clarke. 2006. A document-centric approach to static index pruning in text retrieval systems. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*. 182–189.

[9] David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoelle S. Maarek, and Aya Soffer. 2001. Static index pruning for information retrieval systems. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 43–50.

[10] Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687* (2019).

[11] Zhuyun Dai and Jamie Callan. 2020. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*. 1897–1907.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086* (2021).
[14] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2288–2292.
[15] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. A white box analysis of ColBERT. In *European Conference on Information Retrieval*. Springer, 257–263.
[16] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (1987), 964–971.
[17] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3030–3042.
[18] Sebastian Hofstätter, Omar Khattab, Sophia Althammer, Mete Sertkan, and Allan Hanbury. 2022. Introducing neural bag of whole-words with ColBERTer: Contextualized late interactions using enhanced reduction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 737–747.
[19] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in Facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.
[20] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*.
[21] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* (2019).
[22] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 6769–6781.
[23] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 39–48.
[24] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.
[25] Carlos Lassance, Simon Lupart, Hervé Dejean, Stéphane Clinchant, and Nicola Tonellotto. 2023. A static pruning study on sparse neural retrievers. *arXiv preprint arXiv:2304.12702* (2023).
[26] Carlos Lassance, Maroua Maachou, Joohee Park, and Stéphane Clinchant. 2021. A study on token pruning for ColBERT. *arXiv preprint arXiv:2112.06540* (2021).
[27] Jimmy Lin and Xueguang Ma. 2021. A few brief notes on DeepImpact, COIL, and a conceptual framework for information retrieval techniques. *arXiv preprint arXiv:2106.14807* (2021).
[28] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. 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)*. 2356–2362.
[29] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[30] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics* 9 (2021), 329–345.
[31] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning passage impacts for inverted indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1723–1727.
[32] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).
[33] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online preprint* (2019).
[34] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375* (2019).
[35] Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 275–281.

[36] Yujie Qian, Jinhyuk Lee, Sai Meher Karthik Duddu, Zhuyun Dai, Siddhartha Brahma, Iftekhar Naim, Tao Lei, and Vincent Y. Zhao. 2022. Multi-vector retrieval as sparse alignment. *arXiv preprint arXiv:2211.01267* (2022).
[37] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of BERT in ranking. *arXiv preprint arXiv:1904.07531* (2019).
[38] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
[39] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* (2019).
[40] 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.
[41] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3715–3734.
[42] Donald J. Schuirmann. 1987. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics* 15 (1987), 657–680.
[43] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
[44] Sree Lekha Thota and Ben Carterette. 2011. Within-document term-based index pruning with statistical hypothesis testing. In *Advances in Information Retrieval: 33rd European Conference on IR Research, ECIR 2011, Dublin, Ireland, April 18-21, 2011. Proceedings 33*. Springer, 543–554.
[45] Nicola Tonellotto and Craig Macdonald. 2021. Query embedding pruning for dense retrieval. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3453–3457.
[46] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.
[47] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems* 32 (2019).
[48] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1503–1512.
[49] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. An analysis of BERT in document ranking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1941–1944.
[50] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. RepBERT: Contextualized text embeddings for first-stage retrieval. *arXiv preprint arXiv:2006.15498* (2020).

Received 15 March 2023; revised 31 October 2023; accepted 21 December 2023