

# SpaDE: Improving Sparse Representations using a Dual Document Encoder for First-stage Retrieval

Eunseong Choi\*  
Sungkyunkwan University  
Republic of Korea  
eunseong@skku.edu

Sunkyung Lee\*  
Sungkyunkwan University  
Republic of Korea  
sk1027@skku.edu

Minjin Choi  
Sungkyunkwan University  
Republic of Korea  
zxcvxd@skku.edu

Hyeseon Ko  
Naver Corp.  
Republic of Korea  
hyeseon.ko@navercorp.com

Young-In Song  
Naver Corp.  
Republic of Korea  
song.youngin@navercorp.com

Jongwuk Lee†  
Sungkyunkwan University  
Republic of Korea  
jongwuklee@skku.edu

## ABSTRACT

Sparse document representations have been widely used to retrieve relevant documents via *exact lexical matching*. Owing to the pre-computed inverted index, it supports fast ad-hoc search but incurs the *vocabulary mismatch problem*. Although recent neural ranking models using pre-trained language models can address this problem, they usually require expensive query inference costs, implying the trade-off between effectiveness and efficiency. Tackling the trade-off, we propose a novel uni-encoder ranking model, *Sparse retriever using a Dual document Encoder (SpaDE)*, learning document representation via the dual encoder. Each encoder plays a central role in (i) adjusting the importance of terms to improve *lexical matching* and (ii) expanding additional terms to support *semantic matching*. Furthermore, our *co-training* strategy trains the dual encoder effectively and avoids unnecessary intervention in training each other. Experimental results on several benchmarks show that SpaDE outperforms existing uni-encoder ranking models.

## CCS CONCEPTS

• Information systems → Retrieval models and ranking; Document representation.

## KEYWORDS

Neural ranking; Pre-trained language model; Sparse representations

### ACM Reference Format:

Eunseong Choi, Sunkyung Lee, Minjin Choi, Hyeseon Ko, Young-In Song, and Jongwuk Lee. 2022. SpaDE: Improving Sparse Representations using a Dual Document Encoder for First-stage Retrieval. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*

\*Equal contribution

†Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).

CIKM '22, October 17–21, 2022, Atlanta, GA, USA.

© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-9236-5/22/10...\$15.00

<https://doi.org/10.1145/3511808.3557456>

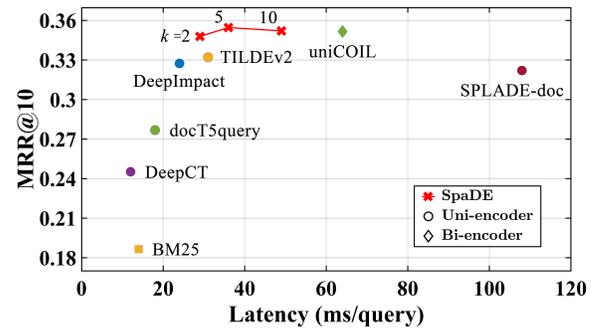


Figure 1: MRR@10 and average query latency (in ms) on MS MARCO development set of SpaDE varying the size of top-k masking for the document-level pruning and existing retrievers with sparse representations. The latency is measured with PISA [44] using Block-Max WAND [14].

(CIKM '22), October 17–21, 2022, Atlanta, GA, USA. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3511808.3557456>

## 1 INTRODUCTION

Sparse document representations have been commonly used in classical information retrieval (IR) models, such as TF-IDF [27] and BM25 [56]. Since sparse representations are compatible with the inverted index on the vocabulary space, query inference is remarkably fast for ad-hoc retrieval. However, it merely relies on exact *lexical matching* between a query and a document, inevitably incurring the *vocabulary mismatch problem*, i.e., different terms with similar meanings in queries and documents are mismatched.

Neural ranking models [21, 24, 45, 66, 72] have been studied to address the vocabulary mismatch problem. Given mismatched terms, e.g., dog vs. puppy, neural ranking models successfully match their semantics. Notably, pre-trained language models (PLM), e.g., BERT [12], using contextualized representations, have shown remarkable performance leaps on ad-hoc retrieval. As a simple baseline, monoBERT [47] adopts BERT for query-document matching; a *cross-encoder* carries out *all-to-all interactions* across query and document terms, effectively providing complex matching signals. However, it leads to too slow inference time, e.g., > 1,000 ms. Even worse, the cross-encoder model cannot leverage pre-computed indexes, which is impractical for the first-stage retrieval.

Supporting efficient query inference is critical for the first-stage retrieval of billion-scale documents. Recent studies, *e.g.*, ColBERT [29, 57], COIL [19], and SPLADE [15, 16], have proposed to separate query and document encoders, also known as a *bi-encoder*. Given a pair of (query, document), it encodes query and document representations independently and matches them via late interaction. Although the bi-encoder can utilize a pre-computed index for document representations, it is necessary to pass through the query encoder at the retrieval, requiring additional computing costs.

In another research direction, some studies [9, 43, 48] have designed *uni-encoder* models representing sparse document vectors without a complex query encoder. It is categorized into two directions: (i) *term weighting* to elaborate term scoring or (ii) *term expansion* to mitigate vocabulary mismatching. For term weighting, DeepCT [9] uses BERT [12] to adjust the importance of terms to improve the quality of conventional term frequency scores. However, it still suffers from the vocabulary mismatch problem. For term expansion, doc2query [50] and docT5query [48] generate relevant queries from a document using Transformer [60] or T5 [53] and then add generated queries for document embeddings to enrich document representation. Recently, DeepImpact [43] and TILDEV2 [76] adjust term scores on the document expanded by docT5query [48] or TILDE [77] respectively, mitigating the limitation of term weighting. While uni-encoder models achieve fast query inference by building the inverted index, they show lower accuracy than cross-encoder and bi-encoder models.

In this paper, we propose a novel uni-encoder ranking model, namely *Sparse retriever using a Dual document Encoder (SpaDE)*, to overcome the trade-off between effectiveness and efficiency. Specifically, it adopts a *dual* document encoder advocating two solutions, *i.e.*, *term weighting* and *term expansion*. While term weighting adjusts the term importance scores for lexical matching, term expansion enables semantic matching by appending additional terms to the document. It should be noted that each solution handles a different target scenario, but both scenarios exist in real-world datasets. Although each solution has been used in the literature, it is non-trivial to train the dual encoder effectively.

To address this problem, we devise a simple-yet-effective *co-training* strategy to enjoy the benefit of the dual encoder. Our key idea is to avoid unnecessary intervention in training each encoder with different aspects. Unlike joint training, where the dual encoder is trained with the same samples, our training strategy focuses on selectively training hard samples from another. We choose large-loss samples from one encoder and train another encoder using them, and vice versa. As a result, SpaDE can effectively train the dual encoder with the complementary relationship.

For efficiency, we further leverage a learning-based pruning method to enforce the sparsity of document representations. Using top-*k* masking for *document-level* sparsity and cutoff with approximate document frequency for *corpus-level* sparsity, we gradually prune unnecessary terms in the document and fully enjoy the advantage of the inverted index. Owing to the pruning method, SpaDE exhibits a better trade-off improving query latency by 3.4 times with little accuracy degradation over bi-encoder models.

Figure 1 illustrates how well SpaDE alleviates the trade-off between effectiveness and efficiency for the first-stage retrieval on the MS MARCO dataset. It can serve in 29–49 milliseconds when

**Table 1: Category of existing neural ranking models using PLM with two criteria; representation types and matching paradigms. Note that cross-encoder models using sparse representation and uni-encoder models using dense representation do not exist.**

	Sparse representations	Dense representations
<b>Cross-encoder</b>	-	monoBERT [47], duoBERT [49], Simplified TinyBERT [4], Birch [71], CEDR [39], BERT-MaxP [10], PreTTR [37], PARADE [31]
<b>Bi-encoder</b>	EPIC [38], SparTerm [1], uniCOIL [32], COIL-tok [19], SPLADE [16], SPLADE-max [15], DistilSPLADE-max [15], UHD-BERT [25]	Sentence-BERT [54], DPR [28], ANCE [67], ADORE [74], TAS-B [23], Condenser [17, 18], ColBERT(v1, v2) [29, 57], TCT-ColBERT [35], JPQ [73], COIL-full [19], CLEAR [20], CoRT [65], RepCONC [75], RocketQA(v1, v2) [52, 55]
<b>Uni-encoder</b>	doc2query [50], docT5query [48], DeepCT [9], DeepImpact [43], SPLADE-doc [15], TILDE(v1, v2) [76, 77]	-

the size of top-*k* masking is 2–10. Furthermore, it outperforms existing uni-encoder models and significantly reduces the latency with comparable effectiveness to bi-encoders. In this sense, SpaDE can be used as an alternative for the first-stage ranker in commercial search engines.

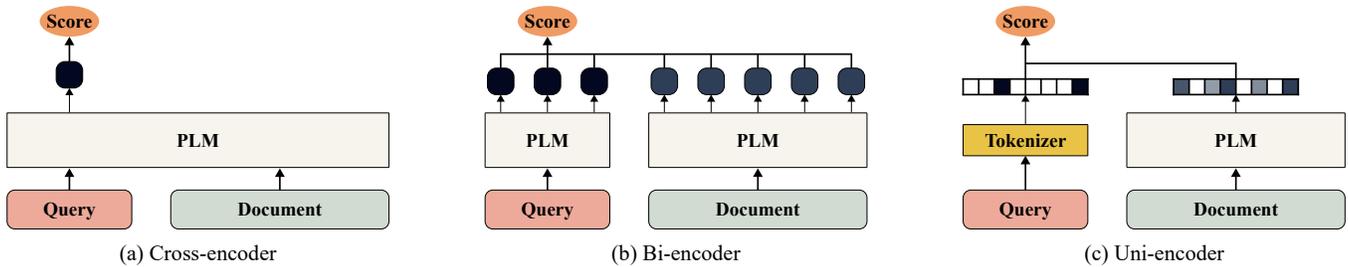
To summarize, the key contributions of this paper are as follows. (i) We propose a novel ranking model, called SpaDE, for effective and efficient first-stage retrieval (Section 3.1). (ii) We introduce a *dual* document encoder, enjoying both term weighting and term expansion for document representations (Section 3.2). (iii) We also present a *co-training* strategy to effectively train the dual encoder (Section 3.3) and a learning-based pruning method to build an efficient inverted index (Section 3.4). (iv) Lastly, we evaluate the effectiveness and efficiency of the proposed method on various benchmark datasets, such as MS MARCO Passage Ranking [46], TREC DL 2019 [8], TREC DL 2020 [7], and BEIR [58] (Sections 4 and 5).

## 2 RELATED WORK

We review existing neural ranking models [34] using the pre-trained language model (PLM). As shown in Table 1, they are categorized into *cross-encoder*, *bi-encoder*, and *uni-encoder* (Figure 2).

### 2.1 Cross-encoder Approach

This approach regards a query and a document as a pair of sentences and performs all-to-all interactions across query and document terms. First, monoBERT [47] took the concatenation of the query and the document as input and computed a relevance score as the point-wise ranking problem. Meanwhile, duoBERT [49] formulated the pair-wise ranking problem to classify relevant and irrelevant documents for the query. The two models can be pipelined for a multi-stage ranking in an end-to-end fashion. Birch [71], BERT-MaxP [10], CEDR [39], and PARADE [31] extended passage ranking



**Figure 2: Illustration of neural ranking models using PLM. Existing PLM-based ranking models are categorized into three approaches depending on query-document matching paradigms.**

into document ranking by utilizing BERT’s representations obtained from the cross-encoder. Despite its effectiveness in capturing complex query-document interactions, it is known that query latency is 100-1,000x slower than BM25, as reported in [29]. Recently, to mitigate the computational cost of cross-encoder, PreTTR [37] stored term representations from an intermediate layer to delay all-to-all interactions, and Simplified TinyBERT [4] utilized knowledge distillation models to reduce query inference time.

## 2.2 Bi-encoder Approach

This approach encodes a query and a document separately and learns the interaction between query and document vectors.

**Dense representations.** The encoder converts the query and the document to continuous low-dimensional vectors, which naturally bypasses the vocabulary mismatch problem. Although SentenceBERT [54] does not directly tackle the retrieval problem, it is used as the basic design for the bi-encoder with dense representations. As the canonical example, DPR [28], ANCE [67], and RocketQA [52, 55] exploited the [CLS] vectors to compute the relevance score. These models increase effectiveness through more sophisticated negative sampling methods. Recently, ADORE [74] used dynamic sampling to adapt hard negative samples during model training. TAS-B [23] proposed topic-aware sampling to select informative queries from topic clusters. Instead of using single dense vector representation, ColBERT [29, 57] and COIL [19] leveraged late interactions between query token vectors and document token vectors.

Meanwhile, Condenser [17, 18] designed a pre-training strategy tailored for ad-hoc retrieval, and RocketQAv2 [55], TCT-ColBERT [35], TAS-B [23], and ColBERTv2 [57] adopted knowledge distillation to improve ranking performance further. To reduce query inference time, dense representation models utilized an approximate similarity search, *e.g.*, faiss [26, 42]. However, since the index size for dense vectors is much larger than the traditional inverted index, loading whole dense vectors requires huge memory space. To deal with the memory footprint problem, JPQ [73] and RepCONC [75] compressed dense representations.

**Sparse representations.** Recent studies have represented sparse query and document vectors. EPIC [38] matched a dense document vector and a sparse query vector to reduce the query latency. SparTerm [1] adopted sparse document representations by combining the importance distribution of terms with a binary gating vector. Later, SPLADE [16] improved SparTerm [1] using a FLOPS regularizer [51], enforcing the sparsity of term distributions. SPLADE-max [15] further improved SPLADE [16] by replacing the

pooling operation. Although COIL-tok [19] and uniCOIL [32] stem from COIL [19], they adopt sparse representations by excluding [CLS] matching or setting the dimension size of dense vectors as one, respectively. In another direction, UHD-BERT [25] adopted a winner-take-all module for binarized sparse representations in ultra-high dimensional space. Although the bi-encoder design using sparse representations can employ the inverted index, it requires additional costs to compute the query encoder. Therefore, they still have a bottleneck in reducing query inference time.

## 2.3 Uni-encoder Approach

For efficient query inference, it is preferred to avoid a complex query encoder, as pointed out in [76, 77]. The uni-encoder models minimize the burden of query inference time, especially enabling practical adoption of GPU-free models. Furthermore, the uni-encoder model is inherently suitable for the inverted index by using only a tokenizer to represent queries into a bag of words.

**Term expansion.** It is reformulated by the machine translation problem. Given a dataset of (query, relevant document) pairs, a sequence-to-sequence model is trained to generate the query from the relevant document. doc2query [50] and docT5query [48] used Transformer [60] and T5 [53] to predict queries and append them to the original document. Expanded documents can also be used to build inverted indexes or as the input of other ranking models [43, 76]. SPLADE-doc [15] performed document expansion in the BERT vocabulary space using the masked language modeling head of BERT. Besides, TILDE [77] utilized the query likelihood for semantic matching.

**Term weighting.** Although classical IR models, *e.g.*, TF-IDF [27] and BM25 [56], are effective for estimating the importance of terms in the document in an unsupervised manner, it does not reflect the term importance for relevant queries. DeepCT [9] formulated term weighting as the regression problem for measuring query term recall. It learned a mapping function from contextualized word representations to term weights in a supervised manner. To mitigate the limitations of the term weighting model, *i.e.*, vocabulary mismatch problem, recent studies [43, 76] adjusted term weighting scores on the expanded document. DeepImpact [43] improved DeepCT [9] by combining docT5query [48] and directly optimizing term weighting scores; it estimates the importance of terms in the document which is expanded by docT5query [48]. Likewise, TILDEv2 [76] followed contextualized term weighting but employed TILDE [77] alternatively for document expansion.

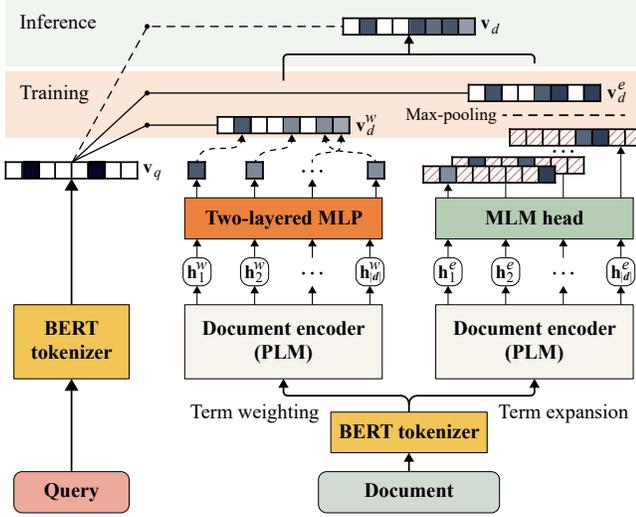


Figure 3: Model architecture of SpaDE with the dual document encoder for term weighting and term expansion. Each encoder returns a document vector after training. For inference, two vectors are aggregated to a document vector.

### 3 PROPOSED MODEL

In this section, we propose a novel ranking model for effective and efficient first-stage retrieval, namely *Sparse retriever using a Dual document Encoder (SpaDE)*, following the uni-encoder paradigm. To overcome the trade-off between effectiveness and efficiency, SpaDE adopts a dual document encoder for *term weighting* and *term expansion*, adjusting term importance for lexical matching and enriching additional terms for semantic matching. To train the dual encoder, we devise a *co-training* strategy to collectively enjoy the strength of the dual encoder. Specifically, it consists of two stages. At the warm-up stage, each encoder is first trained independently. At the fine-tuning stage, we choose large-loss samples from each encoder, and the selected samples are used to train another encoder. Despite its simplicity, it can induce a complementary relationship by selectively training each encoder with hard samples. Furthermore, we utilize a learning-based pruning method to enforce the document-level and the corpus-level sparsity, thereby fully enjoying the efficiency of the inverted index.

In the following, we first explain the overall model architecture (Section 3.1). We then introduce our key contributions, the dual document encoder (Section 3.2), and the co-training strategy (Section 3.3) respectively. Lastly, we explain learning-based pruning for building the efficient inverted index (Section 3.4).

#### 3.1 Model Architecture

Figure 3 depicts the overall architecture of SpaDE. Our goal is to estimate the relevance function for a query  $q$  and a document  $d$  and return the most relevant documents. It is formulated by

$$\text{sim}(q, d) = g(\psi(q), \phi(d)), \quad (1)$$

where  $\psi(q)$  returns a sparse query representation  $\mathbf{v}_q \in \mathbb{R}^{|V|}$ . Based on the BERT tokenizer, we can easily transform the query  $q$  into a boolean query vector in the WordPiece vocabulary space, e.g.,  $|V| =$

30,522 word-pieces. (Although we can define different vocabulary spaces depending on the tokenizer, for simplicity, we utilize the BERT tokenizer.) Note that it does not require additional costs, achieving fast query latency and reducing memory footprint to encode the query vector.

As the core part of SpaDE,  $\phi(d)$  represents the dual encoder for *term weighting* and *term expansion*. The term weighting is responsible for adjusting the importance of terms in the document. The term expansion plays a role in generating a few relevant terms from the document and assigning their weights. After the document  $d$  passes through the dual encoder, we combine two vectors into a sparse document representation  $\mathbf{v}_d \in \mathbb{R}^{|V|}$  in the vocabulary space. During model training, sparsity is enforced by controlling both document-level and corpus-level sparsity.

Lastly, the matching function  $g(\cdot, \cdot)$  is used to measure the similarity between the query vector  $\mathbf{v}_q$  and the document vector  $\mathbf{v}_d$ . In this work, we use a simple inner product for directly optimizing the relevance score between the query and the document.

#### 3.2 Dual Document Encoder

The architecture of the dual document encoder is based on the transformer encoder. Given a document  $d$ , we first tokenize it into sequential tokens as the input of BERT, including two special tokens [CLS] and [SEP].

$$\left[ \mathbf{e}_{[\text{CLS}]}^w, \mathbf{e}_1^w, \dots, \mathbf{e}_{|d|}^w, \mathbf{e}_{[\text{SEP}]}^w \right] \text{ and } \left[ \mathbf{e}_{[\text{CLS}]}^e, \mathbf{e}_1^e, \dots, \mathbf{e}_{|d|}^e, \mathbf{e}_{[\text{SEP}]}^e \right], \quad (2)$$

where each embedding vector  $\mathbf{e}_i^w$  for term weighting encoder (or  $\mathbf{e}_i^e$  for term expansion encoder) is combined with a token embedding vector and a positional embedding vector.

Each embedding vector is then passed into the transformer encoder. The corresponding output is represented by a sequence of hidden vectors.

$$\left[ \mathbf{h}_{[\text{CLS}]}^w, \mathbf{h}_1^w, \dots, \mathbf{h}_{|d|}^w, \mathbf{h}_{[\text{SEP}]}^w \right] \text{ and } \left[ \mathbf{h}_{[\text{CLS}]}^e, \mathbf{h}_1^e, \dots, \mathbf{h}_{|d|}^e, \mathbf{h}_{[\text{SEP}]}^e \right], \quad (3)$$

where  $\mathbf{h}_i^w, \mathbf{h}_i^e \in \mathbb{R}^m$  for  $i \in \{[\text{CLS}], 1, \dots, |d|, [\text{SEP}]\}$  and  $m$  is the dimension of embedding vectors. Note that we follow the conventional structure used in BERT [12].

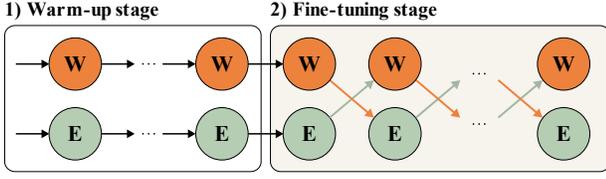
**Term weighting encoder.** As discussed in DeepCT [9], DeepImpact [43] and TILDEv2 [76], it is effective to learn a relevance score for query-document pairs by adjusting the importance of terms in the document. Since the contextualized hidden vector for each token from PLM can capture the word’s syntactic and semantic role in the local context, we compute the weighting score from hidden vectors using a two-layered MLP with the ReLU activation function. Each hidden vector  $\mathbf{h}_i^w$  is projected into a weighting score  $s_i$ .

$$s_i = f_{\text{MLP}}(\mathbf{h}_i^w), \text{ for } i \in \{1, \dots, |d|\}. \quad (4)$$

Conceptually, the weighting score  $s_i$  for each token can be represented by a one-hot vector in the vocabulary space as follows.

$$\mathbf{w}_i = f_{\text{one-hot}}(s_i), \text{ for } i \in \{1, \dots, |d|\}. \quad (5)$$

Here,  $\mathbf{w}_i \in \mathbb{R}^{|V|}$  is the one-hot vector for the  $i$ -th token. To build  $\mathbf{w}_i$ ,  $f_{\text{one-hot}}(\cdot)$  places  $s_i$  into the corresponding term for  $i$ -th token and zeroes the rest. Finally, we aggregate them using the max pooling operation and represent the document vector  $\mathbf{v}_d^w \in \mathbb{R}^{|V|}$  in the



**Figure 4: Schema of our co-training strategy. Let  $W$  and  $E$  denote the term weighting encoder and the term expansion encoder, respectively. Note that a learning-based pruning method is only applied for the fine-tuning stage.**

vocabulary space.

$$\mathbf{v}_d^w = f_{\text{MaxPool}}(\mathbf{w}_1, \dots, \mathbf{w}_{|d|}). \quad (6)$$

**Term expansion encoder.** Since the *masked language model* (MLM) head layer is pre-trained to predict a masked token, it is effective for capturing the semantic correlation between tokens. Motivated by SparTerm [1], we compute the importance vector  $\mathbf{v}_i \in \mathbb{R}^{|V|}$  for each hidden vector  $\mathbf{h}_i^e$  as follows.

$$\mathbf{v}_i = \text{ReLU}(f_{\text{MLM}}(\mathbf{h}_i^e)), \text{ for } i \in \{\{\text{CLS}\}, 1, \dots, |d|, \{\text{SEP}\}\}, \quad (7)$$

where  $f_{\text{MLM}}(\cdot)$  is the MLM head in BERT [12]. We represent the importance score by eliminating negative values using the ReLU activation.

Then, we utilize top- $k$  masking [25, 68] for each  $|V|$ -dimensional vector  $\{\mathbf{v}_{i,j}\}_{j=1}^{|V|}$  to control the sparsity.

$$\mathbf{v}'_{i,j} = \max_k \{\mathbf{v}_{i,j}\}_{j=1}^{|V|} \text{ for } i \in \{\{\text{CLS}\}, 1, \dots, |d|, \{\text{SEP}\}\}, \quad (8)$$

where  $k$  is the hyperparameter to adjust the number of non-zero values in the  $|V|$ -dimensional vector  $\mathbf{v}_i$  and  $\max_k$  is the operation to keep only top- $k$  values while others are set to zero. (Empirically,  $k$  is set as 2–10.) The top- $k$  masking method explicitly enforces the document-level sparsity, so irrelevant or redundant tokens are removed from the document. (Refer to Section 3.4 for further details.)

Finally, we aggregate them into a vocabulary-level vector by using a max-pooling operation.

$$\mathbf{v}_d^e = f_{\text{MaxPool}}(\mathbf{v}'_{\{\text{CLS}\}}, \mathbf{v}'_1, \dots, \mathbf{v}'_{|d|}, \mathbf{v}'_{\{\text{SEP}\}}), \quad (9)$$

where  $\mathbf{v}_d^e \in \mathbb{R}^{|V|}$ . Several pooling operations can be used to combine multiple vectors into a single vector. The max-pooling operation shows the best performance owing to capturing the most salient values for each token, as also reported in [15].

**Aggregation.** Finally, SpaDE combines two vectors,  $\mathbf{v}_d^w$  and  $\mathbf{v}_d^e$ , in the vocabulary space. In this process, we utilize an aggregating hyperparameter  $\alpha$  to properly combine two vectors.

$$\mathbf{v}_d = (1 - \alpha) \cdot \mathbf{v}_d^w + \alpha \cdot \mathbf{v}_d^e, \quad (10)$$

where  $\alpha$  is the hyperparameter to balance two document vectors. (Empirically, the best performance was shown when  $0.3 \leq \alpha \leq 0.5$ .) SpaDE derives the final document vector by summing two representations of different characteristics. In other words, SpaDE (i) adjusts the importance of terms for lexical matching and (ii) endows additional terms for semantic matching.

### 3.3 Co-training Strategy

We devise a *co-training* strategy to fully enjoy the different roles of the dual encoder. The idea of co-training comes from the observation that joint learning does not achieve a clear improvement compared to training with a single encoder. (Please refer to Table 6 for empirical results on the effect of training strategies.) Meanwhile, sequential training [5] helps improve the efficacy of training when handling two encoders. We conjecture that training with two encoders should be carefully handled, reducing the intervention from another encoder during training.

Based on the empirical observation, we design the co-training strategy with two stages (Figure 4). At the warm-up stage, each encoder is first trained independently. At the fine-tuning stage, we choose *large-loss* samples from each encoder and use those samples for fine-tuning another encoder. Each encoder focuses on learning different samples that their counter encoder is hard to learn, thereby compensating for the weakness of each encoder.

**Warm-up stage.** We train the two encoders independently with different objective functions. For the term weighting encoder, we adopt a negative log likelihood of the positive passage. By minimizing the loss function, we can optimize the query-document relevance score as

$$\mathcal{L}_{\text{weighting}} = -\log \frac{\exp(\mathbf{v}_q^\top \mathbf{v}_{d^+}^w)}{\exp(\mathbf{v}_q^\top \mathbf{v}_{d^+}^w) + \sum_{d^- \in \mathcal{N}(q)} \exp(\mathbf{v}_q^\top \mathbf{v}_{d^-}^w)}, \quad (11)$$

which defined over a relevant query-document pair  $(q, d^+) \in \mathcal{R}$  and a set of negative documents  $\mathcal{N}(q)$ .  $\mathcal{R}$  is a training set of relevant query-document pairs.

For the term expansion encoder, we also adopt the negative log likelihood. Since SpaDE defines the relevance score using only the query terms, the term expansion encoder may not learn relevance scores for the non-query terms. To resolve the problem, we additionally introduce a point-wise loss to minimize the loss between  $\mathbf{v}_{d^+}^e$  and  $\mathbf{v}_q$ ;  $\mathbf{v}_d^e$  is learned to enforce high scores for relevant query terms and low scores for irrelevant non-query terms. Note that the point-wise loss is similar to query likelihood estimation used in [77].

$$\mathcal{L}_{\text{expansion}} = -\log \frac{\exp(\mathbf{v}_q^\top \mathbf{v}_{d^+}^e)}{\exp(\mathbf{v}_q^\top \mathbf{v}_{d^+}^e) + \sum_{d^- \in \mathcal{N}(q)} \exp(\mathbf{v}_q^\top \mathbf{v}_{d^-}^e)} - \lambda \cdot \mathbf{v}_q \log \text{softmax}(\mathbf{v}_{d^+}^e), \quad (12)$$

where  $\lambda$  is the hyperparameter to adjust the importance of the point-wise loss. (Empirically, we set  $\lambda = 1$ .)

**Fine-tuning stage.** Given training data, one encoder first selects its large-loss samples as hard samples, and another encoder then uses those samples for fine-tuning. Specifically, each encoder extracts  $\tau\%$  large-loss samples from the training set  $\mathcal{R}$  for one another, *i.e.*,  $\mathcal{R}_{\text{weighting}}$  and  $\mathcal{R}_{\text{expansion}}$ , where  $\tau$  is the hyperparameter to control the ratio of the hard training sets. To provide more informative samples for each encoder, we utilize expanded documents using docT5query [48], as in DeepImpact [43]. For example, when the original document is used, the term weighting encoder may pass easily handled vocabulary mismatching samples to the term expansion encoder. (Empirically, we set  $\tau = 30$ .)

### 3.4 Learning-based Pruning

Building inverted indexes using dense document vectors drops the efficiency dramatically; not only do they occupy a lot of storage space, but all query terms should traverse all documents. As pointed out in [1, 16, 25, 68, 72], it is challenging to enforce sparse document representations. Previous studies exploited various techniques to obtain sparse document representations using regularization terms or parameter constraints, *e.g.*,  $L_1$  regularization [72], the FLOPS regularizer [16], learning-based gating vector [1], winner-take-all module [25], and top- $k$  masking [68].

In this paper, we adopt a learning-based pruning method to obtain sparse document representations. Specifically, SpaDE filters out some unnecessary terms to ensure the sparsity *within* the document and *across* the documents, *i.e.*, *document-* and *corpus-level* pruning. For document-level pruning, we use top- $k$  masking to explicitly control sparsity by  $k$  without complex modules. In the training process of the term expansion encoder, SpaDE leaves only top- $k$  values per token in the vocabulary-level vector after passing through the MLM layer as in Eq.(8). (Empirically, the average number of tokens per document is 189 when  $k=10$ .)

Besides, we utilize corpus-level pruning based on an *approximate* document frequency. Even with top- $k$  masking,  $v_d$  can still be inappropriate for building the inverted index in that it merely controls the document-level sparsity. As most of the document includes frequent terms, *e.g.*, ‘*the*’ or ‘*a*’, it can significantly hurt the benefit of the inverted index. It is crucial to consider the corpus-level sparsity for each document, *i.e.*, pruning terms with high document frequency. For that, SpaDE removes the terms that appear in too many documents and learns only with the remaining terms. Let  $|D'|$  denote the number of documents during training, and the terms that appear more than  $|D'| \times \gamma$  times in the document are removed. (Empirically, we set  $\gamma$  to 0.7 and omit a detailed result due to the space limitation.) During the fine-tuning stage, we apply the document- and corpus-level pruning, ensuring the sparsity of document representations.

## 4 EXPERIMENTAL SETUP

**Datasets.** We mainly conduct our experiments on the MS MARCO Passage Ranking dataset [46]. It consists of 8.8M passages collected from Bing’s results and 1M real-world queries. The average number of words in documents is 56. We use the official development set, which consists of 6,980 queries with 1.07 relevant documents per query on average. We use triples<sup>1</sup> sampled from the MS MARCO training dataset used in [43] and randomly split 3,000 queries for a validation set. We use a corpus consisting of BM25 top 100 passages to reduce a time for validation, *i.e.*,  $|D| \approx 300,000$ . Additionally, we evaluate the models with TREC 2019 Deep Learning (DL 2019) [8] and TREC 2020 Deep Learning (DL 2020) [7] queries. Each dataset has 43 and 54 evaluation queries. We also verify the zero-shot performance of ranking models using BEIR [58].

**Competitive models.** We compare SpaDE with thirteen sparse representation based models, including one traditional model [56], seven bi-encoders [1, 15, 19, 25, 32], and five uni-encoders [9, 15,

43, 48, 76]; we do not consider dense representation based models [4, 19, 29, 37] due to their high query inference costs. BM25 [56] is the conventional IR model using lexical matching. SparTerm [1] predicts the importance distribution in vocabulary space and represents them sparsely using a binary gating vector. UHD-BERT [25] learns high-dimensional sparse representations of a query and a document adopting a winner-take-all module to control sparsity. SPLADE [15, 16] is an extension of SparTerm [1] adopting FLOPS regularizer to ensure sparsity. It has three variants, *i.e.*, SPLADE-max, DistilSPLADE-max, and SPLADE-doc. COIL-tok [19] and uni-COIL [32] are based on COIL [19] while adopting sparse representations. DeepCT [9] leverages BERT [12] to adjust term frequencies in the document. Next, docT5query [48] extends the document terms by generating relevant queries from the documents using T5 [53]. DeepImpact [43] is an improved term weighting model of DeepCT [9] to directly learn the term scores by adopting docT5query [48] for document expansion. Besides, TILDEv2 [76] is a term weighting model that improves TILDE [77].

**Reproducibility.** We implemented SpaDE using PyTorch. The pre-trained language model of SpaDE was initialized with BERT<sub>base</sub>. We used the BERT WordPiece vocabulary ( $|V| = 30,522$ ) and set  $m$  to 768. We used the Adam optimizer with a learning rate of  $1e-5$  and  $5e-6$  for term weighting and term expansion, respectively. We set the max sequence length of BERT to 256, dropout rate to 0.1, and batch size to 32. We conducted a grid search for  $\lambda$  among  $\{0.1, 1, 10\}$  and  $\tau$  in  $[0, 100]$  with the step size 10 and set to 1 and 30, respectively. For an aggregating hyperparameter  $\alpha$ , we searched among  $[0, 1]$  with the step size 0.1 on the valid set for each run. We set the warm-up stage to 32,000 iterations. We conducted all the optimizations on the valid set and kept the best checkpoint using MRR@10 by evaluating every 1,600 iterations; we stopped training after MRR@10 had not been improved five times, which stops after 54,400 iterations on average. For the document-level pruning, top- $k$  masking is 2–10. For corpus-level pruning, we set  $\gamma=0.7$  after tuning in  $[0.1, 0.9]$ . For the MS MARCO passage set, we used expanded documents<sup>1</sup> by docT5query [48]. We quantized the token scores into eight bits for SpaDE. For all models, we built the inverted index by Anserini [69] and exported it to the Common Index File Format [33] before being imported into PISA [44] following [40]. We adopted in-batch negatives where all passages for other queries in the same batch are considered negative. Experimental results for SpaDE are averaged over three runs with different seeds. For BM25 [56], we followed the official guide<sup>2</sup>. For uniCOIL<sup>3</sup> [32], COIL-tok<sup>3</sup> [19], SPLADE-max<sup>4</sup> [15], DistilSPLADE-max<sup>5</sup> [15], DeepCT<sup>6</sup> [9], DeepImpact<sup>1</sup> [43], and TILDEv2<sup>7</sup> [76], we used the official code provided by the authors. For docT5query [48], we used the published predicted queries and added 40 predictions to each passage as recommended. For COIL-tok [19], we also reported the results combined with docT5query [48] expansion fair comparison. We do not report the efficiency for COIL-tok [19] because it is unable to compare in the same environment, *e.g.*, PISA [44]. For

<sup>2</sup><http://anserini.io/>

<sup>3</sup><https://github.com/luyug/COIL/tree/main/uniCOIL>

<sup>4</sup><https://github.com/naver/splade>

<sup>5</sup><https://github.com/castorini/pyserini/blob/master/docs/experiments-spladev2.md>

<sup>6</sup><https://github.com/AdeDZY/DeepCT>

<sup>7</sup><https://github.com/ielab/TILDE>

<sup>1</sup><https://github.com/DI4IR/SIGIR2021>

**Table 2: Effectiveness and efficiency comparisons of various models using sparse representations on MS MARCO passage ranking. The latency is measured by averaging time over each query of the MS MARCO development set with a single thread and a single batch. Among uni-encoder models, the best model is marked **bold**, and the second-best model is underlined. Let dT5q denote docT5query [48]. Results not available are denoted as ‘-’. Note that the latency is reported only for the models available for retrieval via PISA [44]. Statistical significant differences ( $p < 0.05$ ) with Bonferroni correction between the reproduced baseline models and the proposed models, *i.e.*, SpaDE ( $k=5$ ) and SpaDE ( $k=10$ ), are reported with \* and  $\diamond$ , respectively.**

Encoder	Model	MS MARCO dev		TREC DL 2019		TREC DL 2020		Latency (ms/query)	Index size (GB)
		MRR@10	Recall@1k	nDCG@10	MAP	nDCG@10	MAP		
-	BM25 [56]	0.187* $\diamond$	0.859* $\diamond$	0.500* $\diamond$	0.293	0.490* $\diamond$	0.289	14	0.8
Bi-encoder	SparTerm [1]	0.279	0.925	-	-	-	-	-	-
	UHD-BERT [25]	0.300	0.960	-	-	-	-	-	-
	COIL-tok [19]	0.341	0.948* $\diamond$	0.688	0.457	0.697	0.452	-	-
	COIL-tok w/ dT5q [19]	0.363	0.968	0.701	0.474	0.715	0.484	-	-
	uniCOIL [32]	0.352	0.958 $\diamond$	0.702	0.461	0.673	0.439	64	1.4
	SPLADE-max [15]	0.340	0.965	0.682	0.431	0.671	0.451	412	2.0
	DistilSPLADE-max [15]	0.369	0.979	0.728	0.485	0.710	0.490	1,606	4.6
Uni-encoder	DeepCT [9]	0.246* $\diamond$	0.911* $\diamond$	0.550	0.339	0.553*	0.344	12	0.8
	docT5query [48]	0.276* $\diamond$	0.946* $\diamond$	0.641	0.403	0.617	0.408	18	1.2
	DeepImpact [43]	0.327* $\diamond$	0.948* $\diamond$	<b>0.696</b>	<b>0.457</b>	0.652	0.426	24	1.6
	SPLADE-doc [15]	0.325* $\diamond$	0.939* $\diamond$	0.671	0.406	0.611	0.399	108	2.1
	TILDEv2 [76]	0.333* $\diamond$	0.958 $\diamond$	0.652	0.408	0.648	0.432	31	2.0
	SpaDE ( $k=2$ )	0.348	0.961	0.674	0.423	0.662	0.445	29	1.6
	SpaDE ( $k=5$ )	<b>0.355</b>	<u>0.965</u>	<u>0.682</u>	<u>0.437</u>	<b>0.677</b>	<u>0.453</u>	36	2.3
	SpaDE ( $k=10$ )	<u>0.352</u>	<b>0.968</b>	0.678	<u>0.437</u>	<u>0.665</u>	<b>0.454</b>	49	3.6

SPLADE-doc [15], we reproduced the model following the hyperparameters from the paper [15]. For SparTerm [1] and UHD-BERT [25], the results are obtained from the original paper. We conducted all experiments on a desktop with 2 NVidia GeForce RTX 3090, 512 GB memory, and a single Intel Xeon Gold 6226. All the source code is available<sup>8</sup>.

**Evaluation metrics.** To measure the effectiveness, we use recall, mean reciprocal rank (MRR), normalized discounted cumulative gain (nDCG), and mean average precision (MAP) with retrieval size  $K$ . Recall is defined as  $\frac{\sum_{i=1}^N rel_i}{k}$ , where  $i$  is the position in the list,  $k$  is the number of relevant documents and  $rel_i \in \{0, 1\}$  indicates whether the  $i$ -th document is relevant to the query or not. We report Recall for  $K=1000$ . MRR is defined as  $\frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$ , where  $rank_i$  refers to the rank position of the first relevant document for the  $i$ -th query. nDCG considers the order of retrieved documents in the list. DCG@ $K$  is defined as  $\sum_{i=1}^K \frac{2^{rel_i} - 1}{\log_2(i+1)}$  where  $rel_i$  is the graded relevance of the result at position  $i$ . nDCG is the ratio of DCG to the maximum possible DCG for the query, which occurs when the retrieved documents are presented in decreasing order of relevance. MRR and nDCG is the official metric of MS MARCO Passage Ranking and TREC Deep Learning Track, respectively. We also report MAP used in existing studies [29, 43]. For MRR and nDCG, we set  $K=10$ . To measure the efficiency, we report the average latency time for processing each query with a single thread and a single batch.

<sup>8</sup><https://github.com/eunseongc/SpaDE>

## 5 RESULTS AND ANALYSIS

We evaluate the effectiveness and efficiency of SpaDE with competing models and summarize meaningful results.

- SpaDE achieves state-of-the-art performance with acceptable latency. On the MS MARCO development set, SpaDE achieves MRR@10 and Recall@1k of 0.352 and 0.968, outperforming the best competitive uni-encoder model by 6.67% and 1.04% with fast query inference time. (Section 5.1)
- SpaDE proves generalization capabilities in the zero-shot setting. For search tasks from the BEIR dataset, SpaDE shows nDCG@10 of 0.462 on average, outperforming the best competing model by 3.13%. (Section 5.1)
- SpaDE addresses the trade-off between effectiveness and efficiency by showing higher accuracy than other baselines with comparable latency. (Section 5.2)
- The co-training strategy for the dual document encoder shows better accuracy than simple joint learning by 15.4% in MRR@10. (Section 5.3)
- The learning-based pruning method improves efficiency by 3.4x faster in query latency without sacrificing the retrieval effectiveness. (Section 5.3)

### 5.1 Effectiveness vs. Efficiency

**Full-ranking evaluation on MS MARCO.** Table 2 reports the first-stage retrieval accuracy on the MS MARCO passage ranking dataset. The key observations are as follows: (i) SpaDE shows the best performance among all uni-encoder models and is comparable to bi-encoder models. It is well-suited as a first-stage retriever in

**Table 3: nDCG@10 of uni-encoder models on BEIR [58]. The best model is marked **bold**, and the second-best model is underlined. Let dT5q and S-doc denote docT5query and SPLADE-doc. For fair evaluation, we exclude expanded terms using docT5query in SpaDE ( $k=10$ ).**

Corpus	BM25	DeepCT	dT5q	S-doc	SpaDE
BEIR Search Tasks					
DBPedia [22]	0.273	0.177	0.331	<u>0.338</u>	<b>0.353</b>
FiQA-2018 [41]	0.236	0.191	<b>0.336</b>	0.233	<u>0.288</u>
HotpotQA [70]	<u>0.603</u>	0.503	0.580	0.537	<b>0.629</b>
NFCorpus [3]	0.325	0.283	<b>0.328</b>	0.310	<u>0.327</u>
NQ [30]	0.329	0.188	0.399	<u>0.400</u>	<b>0.476</b>
TREC-COVID [61]	0.656	0.406	<b>0.713</b>	0.568	<u>0.700</u>
Average	0.404	0.291	<u>0.448</u>	0.398	<b>0.462</b>
BEIR Semantic Relatedness Tasks					
Touché(v2) [2]	<b>0.367</b>	0.156	<u>0.347</u>	0.241	0.276
ArguAna [62]	<u>0.315</u>	0.309	<b>0.349</b>	0.230	0.288
Climate-FEVER [13]	<b>0.213</b>	0.066	<u>0.201</u>	0.091	0.167
FEVER [59]	<b>0.753</b>	0.353	<u>0.714</u>	0.627	0.691
SCIDOCS [6]	<u>0.158</u>	0.124	<b>0.162</b>	0.137	0.149
SciFact [63]	0.665	0.630	<u>0.675</u>	0.649	<b>0.676</b>
Average	<b>0.412</b>	0.273	<u>0.408</u>	0.329	0.374

terms of high recall and low computational cost. In particular, when expanding two terms per token, it achieves higher recall with lower query latency than other models except for models using BM25 algorithms, *e.g.*, BM25 [56], DeepCT [9], and docT5query [48]. (ii) Other than uniCOIL [32], bi-encoder models took much more query processing time than uni-encoder models since they expand query terms, increasing the number of matching terms. Although SpaDE belongs to the uni-encoder models, it achieves comparable retrieval effectiveness to bi-encoders with much lower query latency. Query encoding time is excluded for bi-encoder models in measuring latency. Note that uniCOIL [32] squeezes the dimension of a token embedding of COIL-tok [19] to one and is always more efficient or similar. (iii) Increasing the expanded number of terms per token  $k$  improves the recall, which indicates that the term expanding encoder effectively produces essential terms that are not covered by the term weighting encoder.

In MS MARCO development queries, SpaDE achieves the best performance among the uni-encoder models showing a clear improvement, *e.g.*, +0.022 in MRR@10. This gain is 3.6x bigger than the improvement between the best and the second-best competitive models. TREC DL 2019 and 2020 can be considered closer to the real-world scenarios since the average number of relevant documents per query is 58.2 and 30.9, and the relevance score is judged on a four-point scale. As reported in Table 2, SpaDE mostly shows better performance than all uni-encoders on TREC DL 2019 and 2020, except for DeepImpact [43].

**Zero-shot evaluation on BEIR.** Table 3 reports the zero-shot performance of the first-stage retrieval models on the BEIR [58] dataset. Among uni-encoder models, the models that adopt the document expansion are excluded from comparison if the results are

**Table 4: Effectiveness of the uni-encoder models on the subsets of the MS MARCO development set, considering the degree of query and document overlapping. The best model is marked **bold**, and the second-best model is underlined. The number in parentheses indicates the number of queries. For SpaDE, significant differences ( $p < 0.05$ ) with Bonferroni correction are reported with  $\diamond$ .**

Dataset	Model	MRR@10	Recall@1k	MAP
Match $\leq 1$ (291)	BM25 [56]	0.016 $\diamond$	0.397 $\diamond$	0.012 $\diamond$
	DeepCT [9]	0.034 $\diamond$	0.604 $\diamond$	0.029 $\diamond$
	docT5query [48]	0.106	0.785	0.102
	DeepImpact [43]	0.135	0.721	0.128
	SPLADE-doc [15]	0.130	0.706 $\diamond$	0.124
	TILDEv2 [76]	<u>0.144</u>	<u>0.793</u>	<b>0.140</b>
	SpaDE ( $k=10$ )	<b>0.145</b>	<b>0.799</b>	<u>0.137</u>
Match $> 1$ (5,959)	BM25 [56]	0.154 $\diamond$	0.847 $\diamond$	0.150 $\diamond$
	DeepCT [9]	0.223 $\diamond$	0.915 $\diamond$	0.218 $\diamond$
	docT5query [48]	0.261 $\diamond$	0.948 $\diamond$	0.257 $\diamond$
	DeepImpact [43]	0.311 $\diamond$	0.953 $\diamond$	0.306 $\diamond$
	SPLADE-doc [15]	0.305 $\diamond$	0.943 $\diamond$	0.299 $\diamond$
	TILDEv2 [76]	<u>0.313</u>	<u>0.961</u>	<u>0.307</u>
	SpaDE ( $k=10$ )	<b>0.337</b>	<b>0.973</b>	<b>0.332</b>
Match $_{all}$ (730)	BM25 [56]	0.449 $\diamond$	0.984 $\diamond$	0.442 $\diamond$
	DeepCT [9]	0.516	<u>0.996</u>	0.507
	docT5query [48]	0.461 $\diamond$	<b>0.997</b>	0.454 $\diamond$
	DeepImpact [43]	0.535	<u>0.996</u>	0.528
	SPLADE-doc [15]	<u>0.560</u>	<b>0.997</b>	<u>0.553</u>
	TILDEv2 [76]	<b>0.565</b>	<b>0.997</b>	<b>0.556</b>
	SpaDE ( $k=10$ )	0.557	<b>0.997</b>	0.550

not available on the BEIR leaderboard<sup>9</sup>. We divide the datasets into two categories by task types in [58]; search tasks (*i.e.*, bio-medical information retrieval and question answering) and semantic relatedness tasks (*i.e.*, argument retrieval, citation-relatedness, and claim verification) considering the nature of queries following the existing work [57]. It is found that SpaDE shows competitive performance for search tasks, which is similar to passage retrieval. SpaDE shows higher average accuracy than the other baselines, implying it generalizes well to diverse datasets. For semantic relatedness tasks, the models using term frequency scores, *e.g.*, BM25 [56] and docT5query [48], show better results. Compared to the search task, SpaDE is less effective for the semantic relatedness task, but it still shows better generalization capabilities compared to other PLM-based models, *i.e.*, DeepCT [9] and SPLADE-doc [15].

**Break-down analysis on MS MARCO.** Table 4 shows a break-down analysis of the first-stage retrieval accuracy on the MS MARCO development set over uni-encoders. Depending on the number of overlapping terms between the queries and the documents, we divided the queries into three subsets: Match $\leq 1$ , Match $> 1$ , and Match $_{all}$ . A query is classified as Match $_{all}$  if all query terms are matched with the relevant document in the WordPiece vocabulary. When the number of query terms matched with the document is zero (23 queries) or one (268 queries), the query is classified

<sup>9</sup><https://github.com/beir-cellar/beir>

**Table 5: MRR@10 of re-ranking results using MiniLM [64] on the MS MARCO development set over various depths. The best model is marked **bold**.**

Depth	BM25	DeepImpact	TILDEv2	SpaDE ( $k=10$ )
First-stage	0.187	0.327	0.333	<b>0.352</b>
10	0.285	0.378	0.383	<b>0.391</b>
20	0.322	0.391	0.396	<b>0.400</b>
50	0.351	0.398	0.403	<b>0.405</b>
100	0.367	0.402	0.404	<b>0.405</b>

into a  $\text{Match}_{\leq 1}$ . The rest of the queries are classified as  $\text{Match}_{> 1}$ . The query set may vary depending on the vocabulary set, but we judged that the performance comparison is reasonable based on the performance of  $\text{Match}_{\text{all}}$ .

The key observations are as follows: (i) SpaDE shows competitive performance with the other baselines, especially in  $\text{Match}_{\leq 1}$  and  $\text{Match}_{> 1}$ , and remarkably outperforms the others in most of the metrics. It implies that SpaDE expands meaningful terms to the document by effectively using the dual encoder. (ii) By comparing DeepCT [9] and docT5query [48], we can observe that term expanding is essential for semantic matching ( $\text{Match}_{\leq 1}$  and  $\text{Match}_{> 1}$ ) and term weighting is important for effective lexical matching ( $\text{Match}_{\text{all}}$ ). (iii) The models considering both term weighting and term expansion, *i.e.*, SpaDE, DeepImpact [43], SPLADE-doc [15], and TILDEv2 [76], show effective performance for all query sets, proving that adopting both solutions can be effective.

**Re-ranking evaluation.** Table 5 shows the re-ranking performance on the MS MARCO development set over various depths with MiniLM [64] as a cross-encoder re-ranker. Although the original re-ranking task utilizes 1000 passages, it is still a high cost for cross-encoder models. Therefore, in the real-world scenario, re-ranking with fewer passages can be more practical. It is attractive that SpaDE shows outstanding performance in shallow depths, *e.g.*, 0.391 of MRR@10 in the depth of 10, compared to other baselines.

## 5.2 In-depth Analysis

Figure 1 depicts the performance of SpaDE varying the number of expanded terms per token  $k$  with other baselines. (i) As  $k$  increases, SpaDE improves the accuracy, and its performance tends to converge when  $k=10$ . Considering the trade-off between effectiveness and efficiency, it seems appropriate to expand ten terms per token, leading to 189 terms per document on average. (ii) By varying  $k$ , SpaDE achieves higher efficiency than other baselines with comparable effectiveness. Specifically, its latency time is much faster than uniCOIL [32] when  $k=5$  while achieving better performance even without a complex query encoder.

## 5.3 Ablation Study

Table 6 shows the effectiveness of SpaDE with various training methods. (i) The proposed co-training strategy remarkably improves the accuracy compared to using a single encoder by more than 11.4% in MRR@10. Using a dual encoder always shows better performance than using a mere single encoder. It implies that

**Table 6: Ablation study of SpaDE on the MS MARCO development set in training the dual encoder. Let dT5q denote docT5query [48].**

Model	MRR@10	Recall@1k	MAP	Latency
SpaDE ( $k=10$ )	<b>0.352</b>	<b>0.968</b>	<b>0.347</b>	49
Term weighting only	0.309	0.952	0.307	16
Term expansion only	0.316	0.950	0.309	42
Joint learning	0.305	0.950	0.300	56
w/o co-training	0.342	0.965	0.336	48
w/o dT5q expansion	0.344	0.958	0.339	45
w/o corpus-level pruning	0.352	0.965	0.336	165

each encoder successfully captures complementary information to another. However, when we jointly train the dual encoder, each encoder fails to perform well due to unnecessary intervention during training. Instead of minimizing Eq. (11) and Eq. (12), joint training minimizes  $\mathcal{L}_{\text{joint}}$  which replaces  $\mathbf{v}_d^e$  in Eq. (12) to  $\mathbf{v}_d$ . We fix  $\alpha$  to 0.3 to aggregate two vectors in joint training, and it directly learns combined document representations. (ii) When we do not use the co-training strategy, MRR@10 drops from 0.352 to 0.342, indicating that the co-training strategy is effective in accuracy gains. Note that we apply the pruning method from the start of training in this setting. (iii) The term expansion component can alleviate the vocabulary mismatch problem without expanded documents beforehand using docT5query [48]. (iv) Without the corpus-level pruning, query latency is increased by 3.4x while there is no gain in performance. Empirically, there is a trade-off between efficiency and effectiveness over varying cutoff ratios  $\gamma$ , and the performance seems to be converged when  $\gamma=0.7$  at rapid query latency. We set the cutoff ratio  $\gamma$  to 0.7 which leads to 60 stopwords, *e.g.*, {'the', 'it', 'what', 'for', 'as', ...}, to be removed on average.

## 6 CONCLUSION

In this paper, we proposed a novel uni-encoder model, *Sparse retriever using a Dual document Encoder (SpaDE)*, to alleviate the trade-off between effectiveness and efficiency of the IR system. We adopted a dual document encoder for lexical and semantic matching and developed a co-training strategy to mitigate the training intervention between encoders. We also utilized document- and corpus-level pruning during model training, enabling efficient retrieval using the inverted index. Experimental results showed that SpaDE achieves state-of-the-art performance among uni-encoder models with acceptable query latency, notably preferable for commercial IR systems.

## ACKNOWLEDGMENTS

This work was supported by Naver Corporation. Also, it was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00421 AI Graduate School Support Program (SKKU) and 2022-0-00680), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2018R1A5A1060031).

## REFERENCES

- [1] 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. *CoRR abs/2010.00768* (2020).
- [2] Alexander Bondarenko, Lukas Gienapp, Maik Fröbe, Meriem Beloucif, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2021. Overview of Touché 2021: Argument Retrieval. In *CLEF (Lecture Notes in Computer Science, Vol. 12880)*. Springer, 450–467.
- [3] Vera Boteva, Demian Gholipour Ghalandari, Artem Sokolov, and Stefan Riezler. 2016. A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In *ECIR (Lecture Notes in Computer Science, Vol. 9626)*. Springer, 716–722.
- [4] Xuanang Chen, Ben He, Kai Hui, Le Sun, and Yingfei Sun. 2021. Simplified TinyBERT: Knowledge Distillation for Document Retrieval. In *ECIR*, Vol. 12657. 241–248.
- [5] Xilun Chen, Kushal Lakhotia, Barlas Oguz, Anchit Gupta, Patrick S. H. Lewis, Stan Peshterliev, Yashar Mehdad, Sonal Gupta, and Wen-tau Yih. 2021. Salient Phrase Aware Dense Retrieval: Can a Dense Retriever Imitate a Sparse One? *CoRR abs/2110.06918* (2021).
- [6] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *ACL*. 2270–2282.
- [7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. *CoRR abs/2102.07662* (2021).
- [8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *CoRR abs/2003.07820* (2020).
- [9] Zhuyun Dai and Jamie Callan. 2019. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. *CoRR abs/1910.10687* (2019).
- [10] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *SIGIR*. 985–988.
- [11] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Document Term Weighting for Ad-Hoc Search. In *WWW*. 1897–1907.
- [12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.
- [13] Thomas Diggelmann, Jannis Bulian, Massimiliano Ciaramita, and Markus Leipold. 2020. CLIMATE-FEVER: A Dataset for Verification of Real-World Climate Claims. *CoRR abs/2012.00614* (2020).
- [14] Shuai Ding and Torsten Suel. 2011. Faster top-k document retrieval using block-max indexes. In *SIGIR*. 993–1002.
- [15] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. *CoRR* (2021).
- [16] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *SIGIR*. 2288–2292.
- [17] Luyu Gao and Jamie Callan. 2021. Condenser: a Pre-training Architecture for Dense Retrieval. In *EMNLP*. 981–993.
- [18] Luyu Gao and Jamie Callan. 2021. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. *CoRR abs/2108.05540* (2021).
- [19] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. In *NAACL-HLT*. 3030–3042.
- [20] Luyu Gao, Zhuyun Dai, Zhen Fan, and Jamie Callan. 2020. Complementing Lexical Retrieval with Semantic Residual Embedding. *CoRR abs/2004.13969* (2020).
- [21] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM*. 55–64.
- [22] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-Entity v2: A Test Collection for Entity Search. In *SIGIR*. 1265–1268.
- [23] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *SIGIR*. 113–122.
- [24] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A Position-Aware Neural IR Model for Relevance Matching. In *EMNLP*. 1049–1058.
- [25] Kyoungrok Jang, Junmo Kang, Giwon Hong, Sung-Hyon Myaeng, Joohee Park, Taewon Yoon, and Hee-Cheol Seo. 2021. Ultra-High Dimensional Sparse Representations with Binarization for Efficient Text Retrieval. In *EMNLP*. 1016–1029.
- [26] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data* 7, 3 (2021), 535–547.
- [27] Karen Sparck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation* 60, 5 (2004), 493–502.
- [28] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *EMNLP*. 6769–6781.
- [29] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *SIGIR*. 39–48.
- [30] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: a Benchmark for Question Answering Research. *Trans. Assoc. Comput. Linguistics* 7 (2019), 452–466.
- [31] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. PARADE: Passage Representation Aggregation for Document Reranking. *CoRR abs/2008.09093* (2020).
- [32] Jimmy Lin and Xueguang Ma. 2021. A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques. *CoRR abs/2106.14807* (2021).
- [33] Jimmy Lin, Joel M. Mackenzie, Chris Kamphuis, Craig Macdonald, Antonio Mallia, Michal Siedlaczek, Andrew Trotman, and Arjen P. de Vries. 2020. Supporting Interoperability Between Open-Source Search Engines with the Common Index File Format. In *SIGIR*. 2149–2152.
- [34] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2020. Pretrained Transformers for Text Ranking: BERT and Beyond. *CoRR abs/2010.06467* (2020).
- [35] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *CoRR abs/2010.11386* (2020).
- [36] 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. *CoRR abs/1907.11692* (2019).
- [37] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonello, Nazli Goharian, and Ophir Frieder. 2020. Efficient Document Re-Ranking for Transformers by Precomputing Term Representations. In *SIGIR*. 49–58.
- [38] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonello, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. In *SIGIR*. 1573–1576.
- [39] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. In *SIGIR*. 1101–1104.
- [40] Joel Mackenzie, Andrew Trotman, and Jimmy Lin. 2021. Wacky Weights in Learned Sparse Representations and the Revenge of Score-at-a-Time Query Evaluation. *CoRR abs/2110.11540* (2021).
- [41] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. WWW'18 Open Challenge: Financial Opinion Mining and Question Answering. In *WWW*. 1941–1942.
- [42] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.
- [43] Antonio Mallia, Omar Khattab, Nicola Tonello, and Torsten Suel. 2021. Learning Passage Impacts for Inverted Indexes. In *SIGIR*. 1723–1727.
- [44] Antonio Mallia, Michal Siedlaczek, Joel M. Mackenzie, and Torsten Suel. 2019. PISA: Performant Indexes and Search for Academia. In *OSIRRC@SIGIR*, Vol. 2409. 50–56.
- [45] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match using Local and Distributed Representations of Text for Web Search. In *WWW*. 1291–1299.
- [46] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. In *NeurIPS*.
- [47] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *CoRR abs/1901.04085* (2019).
- [48] Rodrigo Nogueira and Jimmy Lin. 2020. From doc2query to docTTTTTquery. *Online preprint* (2020).
- [49] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. *CoRR abs/1910.14424* (2019).
- [50] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *CoRR abs/1904.08375* (2019).
- [51] Biswajit Paria, Chih-Kuan Yeh, Ian En-Hsu Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing FLOPs to Learn Efficient Sparse Representations. In *ICLR*.
- [52] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In *NAACL-HLT*. 5835–5847.
- [53] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.
- [54] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*. 3980–3990.
- [55] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking. In *EMNLP*. 2825–2835.
- [56] Stephen E. Robertson and Steve Walker. 1994. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *SIGIR*.

- 232–241.
- [57] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. *CoRR* (2021).
- [58] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. *CoRR* (2021).
- [59] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *NAACL-HLT*. 809–819.
- [60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 5998–6008.
- [61] Ellen M. Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2020. TREC-COVID: constructing a pandemic information retrieval test collection. *SIGIR Forum* 54, 1 (2020), 1:1–1:12.
- [62] Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018. Retrieval of the Best Counterargument without Prior Topic Knowledge. In *ACL*, Iryna Gurevych and Yusuke Miyao (Eds.). 241–251.
- [63] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or Fiction: Verifying Scientific Claims. In *EMNLP*. 7534–7550.
- [64] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pretrained Transformers. In *Findings of ACL*. 2140–2151.
- [65] Marco Wrzalik and Dirk Krechel. 2021. CoRT: Complementary Rankings from Transformers. In *NAACL-HLT*. 4194–4204.
- [66] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR*. 55–64.
- [67] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *ICLR*.
- [68] Jheng-Hong Yang, Xueguang Ma, and Jimmy Lin. 2021. Sparsifying Sparse Representations for Passage Retrieval by Top-k Masking. *CoRR* abs/2112.09628 (2021).
- [69] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *SIGIR*. 1253–1256.
- [70] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP 2018*. 2369–2380.
- [71] Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying BERT to Document Retrieval with Birch. In *EMNLP-IJCNLP*. 19–24.
- [72] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik G. Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *CIKM*. 497–506.
- [73] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Jointly Optimizing Query Encoder and Product Quantization to Improve Retrieval Performance. In *CIKM*. 2487–2496.
- [74] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. In *SIGIR*. 1503–1512.
- [75] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Learning Discrete Representations via Constrained Clustering for Effective and Efficient Dense Retrieval. In *WSDM*. ACM, 1328–1336.
- [76] Shengyao Zhuang and Guido Zuccon. 2021. Fast Passage Re-ranking with Contextualized Exact Term Matching and Efficient Passage Expansion. *CoRR* abs/2108.08513 (2021).
- [77] Shengyao Zhuang and Guido Zuccon. 2021. TILDE: Term Independent Likelihood model for Passage Re-ranking. In *SIGIR*. 1483–1492.

**Table 7: Effectiveness of various models on MS MARCO document ranking. Among models using sparse representations, the best model is marked **bold**. Let DR and dT5q denote Dense Retrieval and docT5query [48]. Results not available are denoted as ‘-’.**

Rep.	Encoder	Model	MRR@100	R@100
Dense	Bi	DR (Rand Neg) [74]	0.330	0.859
		DR (BM25 Neg) [74]	0.316	0.794
		ANCE [67]	0.377	0.894
		ADORE [74]	0.405	0.919
		RepCONC [75]	0.399	0.911
Sparse	Bi	COIL [19]	-	-
		uniCOIL [32]	0.341	0.864
		uniCOIL-dT5q [32]	0.353	0.886
	-	BM25 [56]	0.278	0.807
	Uni	HDCT [11]	0.320	0.843
	docT5query [48]	0.327	0.861	
	SpaDE ( $k=5$ )	<b>0.369</b>	<b>0.899</b>	

## A DOCUMENT RETRIEVAL

Table 7 shows the performance of SpaDE with other baselines on the MS MARCO document ranking dataset. DR (Random Neg) and DR (BM25 Neg) represent Dense Retrieval models trained with random and BM25 negatives, respectively. All dense retrieval models use RoBERTa-base [36] as an encoder and [CLS] token embedding for query and document representations. The experimental results of them are copied from [74, 75]. uniCOIL and uniCOIL-dT5q performed indexing on segmented passages, and the score of the segmented passage that obtained the highest score is used as the score of the document, *i.e.*, MaxP technique [10]. Other models, including SpaDE, use only the first part of the document truncated to BERT’s max length. We use official metrics for MS MARCO document ranking task, *i.e.*, MRR@100 and Recall@100. As a result, SpaDE achieves the best performance among baselines using sparse representations. It may seem similar to the passage ranking task results, but spaDE’s effectiveness is more highlighted in long documents. For example, in the passage ranking, MRR@10 performance of uniCOIL and SpaDE were 0.351 and 0.353, respectively, but in the document ranking, SpaDE significantly outperforms it by 0.369 versus 0.353. Secondly, ANCE shows high ranking performance, *i.e.*, MRR, compared to SpaDE, but SpaDE has better recall performance. If compared considering the same MRR performance, uni-encoder outperforms bi-encoder in terms of Recall. This trend, which was not shown before, suggests that focusing on intrinsic lexical matching signals may be more effective than learning representations of both documents and queries in the long document environment where vocabulary mismatching occurs relatively less. In other words, as the first-stage retriever in document ranking, the uni-encoder method may be more favorable than the bi-encoder method.

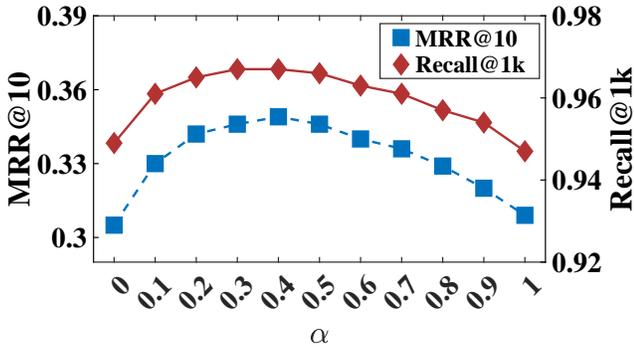


Figure 5: MRR@10 and Recall@1k of SpaDE ( $k=10$ ) over varying  $\alpha$ . Note that we chose  $\alpha$  based on the valid set.

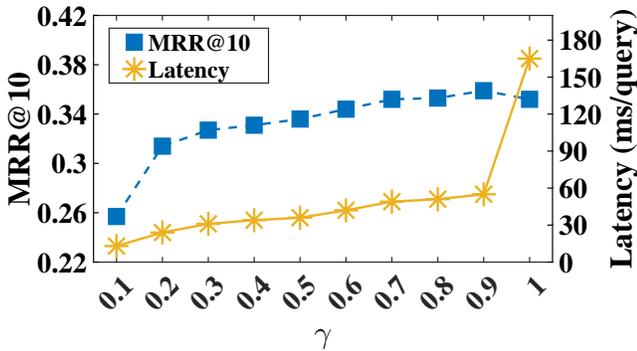


Figure 6: MRR@10 and query latency (in ms) of SpaDE ( $k=10$ ) over varying  $\gamma$  of corpus-level pruning. When  $\gamma=0.7$ , we prune terms appearing in over 70% of documents while training. The latency is measured with PISA [44] using Block-Max WAND [14].

## B EFFECTS OF $\alpha$

Figure 5 shows the effect of the aggregating hyperparameter  $\alpha$ . Using the dual encoder, *i.e.*,  $0 < \alpha < 1$ , mostly shows better performance than using a mere single encoder, *i.e.*,  $\alpha = 0$  or  $\alpha = 1$ , depicting that each encoder only captures complementary information to another. SpaDE ( $k=10$ ) shows the best performance at  $\alpha = 0.4$  with 0.352 in MRR@10, implying that the term weighting encoder is more dominant than the term expansion encoder.

## C EFFECTS OF CORPUS-LEVEL PRUNING

Figure 6 shows the effect of corpus-level pruning of SpaDE. Since new relevant terms are appended through the term expansion component without cutoff based on an approximate document frequency, the number of elements in the term-document matrix is enormous, at about 1.1 billion, implying query processing is very costly, *e.g.*, 510 ms per query. To reduce the query latency, we use corpus-level pruning introduced in Section 3.4. Specifically, during the model training, terms with high document frequency are pruned, and the model learns document representations only with the remaining terms. There is a trade-off between effectiveness and efficiency depending on the cutoff ratio. When the threshold is set

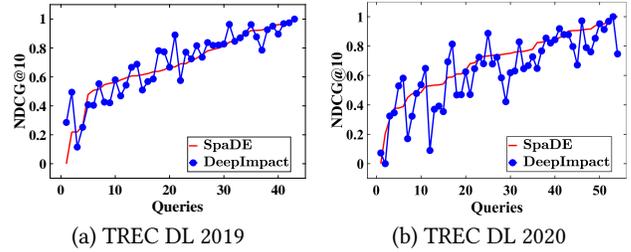


Figure 7: NDCG@10 per query of SpaDE ( $k=10$ ) and DeepImpact [43] on TREC DL 2019 and TREC DL 2020. We sort the queries by increasing order of NDCG@10 scores in SpaDE. If the blue points are above than the red line, it means that DeepImpact [43] shows higher NDCG@10 score than SpaDE. For 51.2% (22 out of 43) and 66.7% (36 out of 54) queries on both datasets, SpaDE outperforms DeepImpact [43].

from 1.0 to 0.2, MRR@10 drops from 0.352 to 0.314, while the inference is about 6.9x faster. Also, it shows the highest performance in MRR@10 when  $\gamma=0.9$ , implying that it can be helpful for retrieval effectiveness to learn document representations, excluding redundant terms. We set the cutoff ratio  $\gamma$  to 0.7 which leads to 60 stop words, *e.g.*, {‘the’, ‘type’, ‘it’, ‘what’, ‘for’, ‘as’, ...}, to be removed on average and the average number of tokens per document is 189.

## D DETAILED TREC DL EVALUATION

Figure 7 shows the NDCG@10 on each query of SpaDE and DeepImpact [43]. While DeepImpact [43] is better than SpaDE in TREC DL 2019, SpaDE is better than DeepImpact [43] in TREC DL 2020. Our analysis found that incorrect queries of SpaDE on TREC DL 2019 mostly include relatively rare words. We conjecture that the term weighting component is difficult to assign high weighting scores on rare words because our learning scheme heavily depends on the document corpus.

## E VISUALIZATION

Table 8 shows the weighted document terms and expanded terms. SpaDE can identify important terms and expand new terms of SpaDE and DeepImpact [43]. For the query “What is a nonconformity? earth science”, SpaDE gives high scores for the important terms, *e.g.*, “nonconformity” and expanding the terms, *e.g.*, “science”, which are matched to the query. Owing to enriched document representations, SpaDE can rank a given relevant document in the 27th place, where the core terms of the query, *e.g.*, “earth”, “science”, do not appear in the document. Meanwhile, even though DeepImpact [43] identifies important document terms, the given document is ranked 77th since query terms are not properly expanded by docT5query [48]. SpaDE accurately expands the query terms by inferring the relevant words. For the query “Dog day afternoon meaning”, SpaDE weights for the important terms, *e.g.*, “dog”, “day”, “afternoon”. Moreover, SpaDE successfully expands terms such as “warm”, “weather”, “hottest”, which are highly related to the query. For the query “What is the most popular food in Switzerland”, both DeepImpact and Spade give high weights for the important terms and expand relevant terms which are matched to the query, *e.g.*, “Switzerland”, “popular”, “most”.

**Table 8: Visualization of term weighting and document expansion by SpaDE ( $k=10$ ) and DeepImpact [43] on TREC DL 2019 and 2020. The number in parentheses denotes the rank of the document. The tokenization process is simplified and only top-50 expanded words are visualized. Yellow shades reflect the normalized term weights and query terms are boxed in red.**

Query	What is a nonconformity? earth science	
Model	DeepImpact [43] (#77)	SpaDE ( $k=10$ ) (#27)
Original document (Rel=3)	1 lack of conformity ; <span style="background-color: yellow;">nonconformity</span> . 2 geology <span style="background-color: yellow;">a</span> surface between successive strata representing <span style="background-color: yellow;">a</span> missing interval in the geologic record of time , and produced either by an interruption in deposition or by the erosion of <span style="background-color: yellow;">depositionally continuous strata</span> followed by renewed deposition .	lack of conformity ; <span style="background-color: yellow;">nonconformity</span> . 2 . geology <span style="background-color: yellow;">a</span> surface between successive strata representing <span style="background-color: yellow;">a</span> missing interval in the geologic record of time , and produced either by an interruption in <span style="background-color: yellow;">deposition</span> or by the erosion of <span style="background-color: yellow;">depositionally continuous strata</span> followed by renewed deposition .
Expanded terms	meant sediment on meaning rock <span style="background-color: yellow;">conformity</span> <span style="border: 1px solid red; padding: 2px;">what</span> <span style="background-color: yellow;">nonconform</span> ----- <span style="border: 1px solid red; padding: 2px;">is</span> <span style="background-color: yellow;">nonconformities</span> <span style="background-color: yellow;">definition</span> <span style="background-color: yellow;">define</span>	<span style="background-color: yellow;">definition</span> <span style="background-color: yellow;">define</span> <span style="background-color: yellow;">meaning</span> <span style="background-color: yellow;">term</span> <span style="background-color: yellow;">absence</span> <span style="background-color: yellow;">apical</span> <span style="background-color: yellow;">geography</span> <span style="border: 1px solid red; padding: 2px;">science</span> <span style="background-color: yellow;">geological</span> <span style="background-color: yellow;">mean</span> : <span style="background-color: yellow;">gap</span> <span style="background-color: yellow;">con</span> <span style="background-color: yellow;">chemistry</span> <span style="background-color: yellow;">lacks</span> <span style="background-color: yellow;">sum</span> <span style="background-color: yellow;">geologist</span> <span style="background-color: yellow;">_rat</span> <span style="background-color: yellow;">lacking</span> <span style="background-color: yellow;">st</span> <span style="background-color: yellow;">formation</span> <span style="background-color: yellow;">definedness</span> <span style="background-color: yellow;">period</span> <span style="background-color: yellow;">rock</span> <span style="background-color: yellow;">following</span> <span style="background-color: yellow;">forms</span> <span style="background-color: yellow;">ism</span> <span style="background-color: yellow;">characterized</span> <span style="background-color: yellow;">not</span> <span style="background-color: yellow;">lacked</span> <span style="background-color: yellow;">rocks</span> <span style="background-color: yellow;">found</span> <span style="background-color: yellow;">why</span> <span style="background-color: yellow;">consecutive</span> <span style="background-color: yellow;">layers</span> <span style="background-color: yellow;">theory</span> <span style="background-color: yellow;">occur</span> <span style="background-color: yellow;">soil</span> <span style="background-color: yellow;">eco</span> <span style="background-color: yellow;">associated</span> <span style="background-color: yellow;">dance</span> <span style="background-color: yellow;">form</span> <span style="background-color: yellow;">history</span> <span style="background-color: yellow;">two</span> <span style="background-color: yellow;">processivity</span> <span style="background-color: yellow;">no</span> <span style="background-color: yellow;">produces</span> <span style="background-color: yellow;">layer</span> <span style="background-color: yellow;">chemical</span> <span style="background-color: yellow;">sedimentary</span> <span style="background-color: yellow;">un</span> <span style="background-color: yellow;">state</span> <span style="background-color: yellow;">uniform</span> <span style="background-color: yellow;">deposition</span> <span style="background-color: yellow;">ation</span>
Query	dog day afternoon meaning	
Model	DeepImpact [43] (#26)	SpaDE ( $k=10$ ) (#9)
Original document (Rel=3)	its when its terribly hot that the <span style="background-color: yellow;">dogs</span> don ' t do anything but lay around . a <span style="border: 1px solid red; padding: 2px;">dog</span> ' s <span style="border: 1px solid red; padding: 2px;">day</span> is normally laying around and doing nothing . if you are doing the same , you are having a <span style="border: 1px solid red; padding: 2px;">dog</span> <span style="border: 1px solid red; padding: 2px;">day</span> <span style="border: 1px solid red; padding: 2px;">afternoon</span> . a hot an humid <span style="border: 1px solid red; padding: 2px;">day</span> when even <span style="background-color: yellow;">dogs</span> have no energy .	its when its terribly hot that the <span style="background-color: yellow;">dogs</span> don ' t do anything but lay around . a <span style="border: 1px solid red; padding: 2px;">dog</span> ' s <span style="border: 1px solid red; padding: 2px;">day</span> is normally laying around and doing nothing . if you are doing the same , you are having a <span style="border: 1px solid red; padding: 2px;">dog</span> <span style="border: 1px solid red; padding: 2px;">day</span> <span style="border: 1px solid red; padding: 2px;">afternoon</span> . a hot an humid <span style="border: 1px solid red; padding: 2px;">day</span> when even <span style="background-color: yellow;">dogs</span> have no energy .
Expanded terms	how yahoo leave days should too what ' s it eat begin after can home <span style="background-color: yellow;">puppy</span> all while puppies isn ' t lot to normal their what definition does why causing feel my linger makes mean sleeps for answers lazy	<span style="background-color: yellow;">puppy</span> <span style="background-color: yellow;">pup</span> why mean dayspies definition home too my <span style="border: 1px solid red; padding: 2px;">meaning</span> their normal your causes should after feel eat while makes about define lying warm get need means cause happens before very called canine weather much cold all time lie average hottest like heat early just so kind laid would morning
Query	what is the most popular food in switzerland	
Model	DeepImpact [43] (#9)	SpaDE ( $k=10$ ) (#10)
Original document (Rel=3)	some other well known <span style="background-color: yellow;">swiss</span> dishes are: 1 <span style="background-color: yellow;">geschnetzeltes</span> : sliced pieces of meat ( pork , veal , chicken ) served with a sauce and often rsti ; 2 cheese fondue : various kinds of <span style="background-color: yellow;">swiss</span> cheese melted <span style="border: 1px solid red; padding: 2px;">in</span> a pot . 3 raclette : various kinds of <span style="border: 1px solid red; padding: 2px;">food</span> ( bread , mushrooms , meat , potatoes ) covered <span style="border: 1px solid red; padding: 2px;">in</span> molten or scalloped cheese .	some other well known <span style="background-color: yellow;">swiss</span> dishes are: 1 <span style="background-color: yellow;">geschnetzeltes</span> : sliced pieces of meat ( pork , veal , chicken ) served with a sauce and often ra sti ; 2 cheese fondue : various kinds of <span style="background-color: yellow;">swiss</span> cheese melted <span style="border: 1px solid red; padding: 2px;">in</span> a pot . 3 raclette : various kinds of <span style="border: 1px solid red; padding: 2px;">food</span> ( bread , mushrooms , <span style="background-color: yellow;">meat</span> , potatoes ) covered <span style="border: 1px solid red; padding: 2px;">in</span> molten or scalloped <span style="background-color: yellow;">cheese</span> .
Expanded terms	how culture common called are used there name people <span style="border: 1px solid red; padding: 2px;">what</span> ' s eat typical they do foods kind <span style="border: 1px solid red; padding: 2px;">the</span> like appetizer different <span style="border: 1px solid red; padding: 2px;">what</span> industry <span style="border: 1px solid red; padding: 2px;">switzerland</span> types meal <span style="border: 1px solid red; padding: 2px;">most</span> for <span style="border: 1px solid red; padding: 2px;">popular</span> <span style="border: 1px solid red; padding: 2px;">is</span> made	how culture common called are used there name people <span style="border: 1px solid red; padding: 2px;">what</span> ' s eat typical they do foods kind <span style="border: 1px solid red; padding: 2px;">the</span> like appetizer different <span style="border: 1px solid red; padding: 2px;">what</span> industry <span style="border: 1px solid red; padding: 2px;">switzerland</span> types meal <span style="border: 1px solid red; padding: 2px;">most</span> for <span style="border: 1px solid red; padding: 2px;">popular</span> <span style="border: 1px solid red; padding: 2px;">is</span> made