Unsupervised Domain Adaptation for Sparse Retrieval by Filling Vocabulary and Word Frequency Gaps

Hiroki Iida and **Naoaki Okazaki**
Department of Computer Science, School of Computing, Tokyo Institute of Technology
{hiroki.iida@nlp.c., okazaki@c.}titech.ac.jp

Abstract

IR models using a pretrained language model significantly outperform lexical approaches like BM25. In particular, SPLADE, which encodes texts to sparse vectors, is an effective model for practical use because it shows robustness to out-of-domain datasets. However, SPLADE still struggles with exact matching of low-frequency words in training data. In addition, domain shifts in vocabulary and word frequencies deteriorate the IR performance of SPLADE. Because supervision data are scarce in the target domain, addressing the domain shifts without supervision data is necessary. This paper proposes an unsupervised domain adaptation method by filling vocabulary and word-frequency gaps. First, we expand a vocabulary and execute continual pretraining with a masked language model on a corpus of the target domain. Then, we multiply SPLADE-encoded sparse vectors by inverse document frequency weights to consider the importance of documents with low-frequency words. We conducted experiments using our method on datasets with a large vocabulary gap from a source domain. We show that our method outperforms the present state-of-the-art domain adaptation method. In addition, our method achieves state-of-the-art results, combined with BM25.

1 Introduction

Information retrieval (IR) systems are widely used nowadays. Most of them are based on lexical approaches like BM25 (Robertson and Walker, 1994). Because lexical approaches are based on bag-of-words (BoW), they suffer from *vocabulary mismatch*, where different words express the same notion. Recently, IR models with a pretrained masked language model (MLM), such as BERT (Devlin et al., 2019) have overcome this problem and outperformed BM25 (Nogueira et al., 2019; Karpukhin et al., 2020; Xiong et al., 2021; Formal et al., 2021).

In particular, SPLADE (Formal et al., 2021) is an effective model for practical use. SPLADE addresses vocabulary mismatch by expanding queries and documents through an MLM. Concretely, SPLADE encodes texts to sparse vectors using the logits of the MLM for each token of the texts. As a result, each element of these vectors corresponds to a word in the vocabulary of the MLM. In addition, the nonzero elements other than tokens appearing in the texts can be considered as query and document expansion. Because the encoded vectors are sparse, SPLADE can realize a fast search by utilizing inverted indexes and outperforms BM25, even when SPLADE is applied to out-of-domain datasets from a source domain of training data.

However, SPLADE still struggles with the exact matching of low-frequency words in the training data (Formal et al., 2022b). This problem is amplified for out-of-domain datasets. In addition, Thakur et al. (2021) discussed that large domain shifts in vocabulary and word frequencies deteriorate the performance of vector-based IR models. Furthermore, preparing massive supervision data for every dataset is impractical due to annotation costs. Thus, a method to address the domain shifts without supervision data is necessary.

Unsupervised domain adaptation (UDA) is an approach to overcome domain shift without supervision data. However, as discussed in Section 7, generated pseudo labeling (GPL) (Wang et al., 2021), a state-of-the-art UDA method using generated queries, cannot solve the problem of low-frequency words on some datasets.

In this paper, we propose a UDA method that fills the vocabulary and word-frequency gap between the source and target domains. Specifically, we use AdaLM (Yao et al., 2021), which is a domain adaptation method for an MLM through vocabulary expansion (Wang et al., 2019; Hong et al., 2021) and continual pre-training (Gururangan et al., 2020) on a domain-specific corpus. We expect AdaLM to

Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 752–765
November 20–23, 2022.

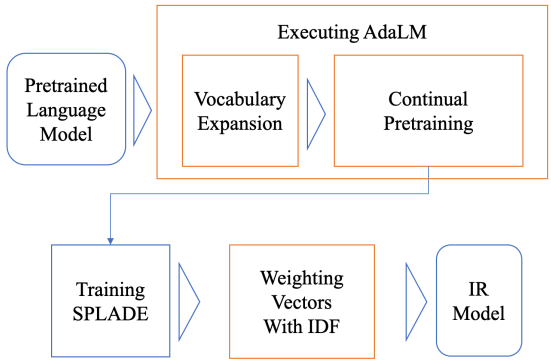

Figure 1: Outline of our method. Orange boxes indicate our proposal.

realize more accurate query and document expansions in the target domain. Furthermore, because SPLADE struggles with exact matching of low-frequency words in the training data, we weight such words by multiplying each element of the SPLADE-encoded sparse vectors by inverse document frequency (IDF) weights. We call our method Combination of AdaLM and IDF (CAI).

We apply CAI to SPLADE and conducted experiments with it on five IR datasets from biomedical and science domains in the BEIR benchmark (Thakur et al., 2021). We used these datasets because the five datasets have the largest vocabulary gap in BEIR from MS MARCO (Nguyen et al., 2016), the source dataset. The experimental results show that SPLADE with CAI outperforms SPLADE with GPL and achieves state-the-art results on average across all datasets by adding scores of BM25.

Finally, to confirm whether CAI can address the problem of exact matching words of low-frequency words in training data, we analyzed the weights of exact matching words, following the approach of Formal et al. (2022b). Our analysis confirms that SPLADE with CAI addresses the problem of the exact matching, whereas SPLADE with GPL cannot.

Our contributions can be summarized as follow:

- We present an unsupervised domain adaptation method, filling vocabulary and word-frequency gaps between the source and target domains. Furthermore, we show that our method performs well in sparse retrieval.
- We confirm that CAI outperforms GPL, the state-of-the-art domain adaptation method for IR, on datasets with large domain shifts from a source dataset.
- Our analysis shows that a factor in the success of CAI is addressing the problem of exact matching of low-frequency words.

2 Related Works

Thakur et al. (2021) showed that vector-based IR models based on a pretrained MLM deteriorate when applied to out-of-distribution datasets. They discussed that one of the causes of the deterioration of the IR performance was a large domain shift in vocabulary and word frequencies. Formal et al. (2022b) also found that IR models based on an MLM struggled with exact matching of low-frequency words in training data. This problem also leads to performance deterioration of MLM-based IR models on out-of-distribution datasets. MacAvaney et al. (2020) showed that a domain-specific MLM performed better than an MLM trained on a corpus of a general domain. However, no previous works showed that addressing vocabulary and word-frequency gaps can solve the problem of deterioration of IR performance for vector-based IR models.

Unsupervised domain adaptation (UDA) is a promising approach to solve the degradation due to domain shift without supervision data. MoDIR (Xin et al., 2022) adopts domain adversarial loss (Ganin et al., 2016) to allow a dense retrieval model to learn domain-invariant representations. Other approaches utilize generated queries. GenQ (Ma et al., 2021) generates queries from a document in an IR corpus with a generative model and then considers the pairs of generated queries and a document as relevant pairs. In addition to GenQ, GPL (Wang et al., 2021) uses documents retrieved by an IR model against a generated query as negative examples and adopts Margin-MSE loss (Hofstätter et al., 2020), which discerns how negative the retrieved documents are. GPL outperforms MoDIR, continual pretraining (Gururangan et al., 2020), and UDALM (Karouzos et al., 2021). However, these approaches target dense representations, and their effect on sparse representation is unclear. We present a more effective UDA method, especially for sparse representations.

3 Method

This paper proposes the UDA method to tackle the domain shifts in vocabulary and word frequency.

An outline of our method is illustrated in Figure 1. Our method consists of three parts: (1) executing AdaLM for domain adaptation of an MLM, (2) training SPLADE with supervised data, and (3) weighting sparse vectors encoded by SPLADE with IDF when searching. Our proposal parts are (1) and (3). We first introduce SPLADE as preliminary.

3.1 SPLADE (Preliminary)

SPLADE (Formal et al., 2021) is a supervised IR model. The model encodes queries and documents to sparse vectors using logits of an MLM and calculates relevance scores by dot products of sparse vectors of the queries and documents.

Let $\mathcal{V}$ denote the vocabulary of an MLM. We represent a text T as a sequence of $n+2$ tokens, $T = (t_0, t_1, t_2, \ldots, t_n, t_{n+1}) \in \mathcal{V}^{n+2}$, where t_0 represents the CLS token and t_{n+1} represents the SEP token. Each token is encoded to a d-dimensional vector, $\boldsymbol{t}_i \in \mathbb{R}^d$, using the MLM. We express the sequence of $n+2$ encoded tokens as $\boldsymbol{T} = (\boldsymbol{t}_0, \boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_n, \boldsymbol{t}_{n+1})$.

We express the process of SPLADE encoding T to a sparse vector $\boldsymbol{s} \in \mathbb{R}^{|\mathcal{V}|}$ as SPLADE(T). Formally, we say

$$\boldsymbol{s} = \text{SPLADE}(T). \tag{1}$$

Now, we explain the encoding process. First, the text T is encoded to $\boldsymbol{T}$. Then, SPLADE converts $\boldsymbol{t}_i \in \boldsymbol{T}$ to a sparse vector $\boldsymbol{s}_i \in \mathbb{R}^{|\mathcal{V}|}$ through the MLM layer. The formal expression is

$$\boldsymbol{s}_i = \boldsymbol{E} f(\boldsymbol{W}\boldsymbol{t}_i + \boldsymbol{b}) + \boldsymbol{c}. \tag{2}$$

Here, $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the embedding layer of the MLM. Then, $\boldsymbol{c} \in \mathbb{R}^{|\mathcal{V}|}$ is a bias term of the embedding layer. $\boldsymbol{W} \in \mathbb{R}^{l \times l}$ is a linear layer, and $\boldsymbol{b} \in \mathbb{R}^l$ is a bias term of the linear layer. $f()$ is an activation function with LayerNorm.

Then, we obtain a sparse vector $\boldsymbol{s}$ by max-pooling with log-saturation effect:

$$\boldsymbol{s} = \max_{0 \le i \le n+1} \log(\boldsymbol{1} + \text{ReLU}(\boldsymbol{s}_i)). \tag{3}$$

Here, the sparse vector $\boldsymbol{s}$ is likely to have nonzero elements other than $t_i \in T$. In this sense, SPLADE can be considered as a method using query and document expansion.

SPLADE infers a relevance score by inner product between sparse vectors of a query and document. We denote a tokenized query as $Q \in \mathcal{V}^l$ and a tokenized document as $D \in \mathcal{V}^m$. l and m are the lengths of Q and D, respectively. The relevance score of SPLADE, $S_{\text{SPL}}(Q, D)$, is formally

$$S_{\text{SPL}}(Q, D) = \text{SPLADE}(Q)^\top \text{SPLADE}(D). \tag{4}$$

Practically, reducing computational cost is another important point, especially when searching. Formal et al. (2021) replaced SPLADE(Q) with a bag-of-words (BoW) representation of a query. Formal et al. (2021) called this scoring **SPLADE-Doc**. This case gives no query expansion. Formally, the score of SPLADE-Doc, $S_{\text{SPL-D}}(Q, D)$, is

$$S_{\text{SPL-D}}(Q, D) = \sum_{t \in Q} \text{SPLADE}(D)_t. \tag{5}$$

Here, SPLADE$(D)_t$ is the element t of a sparse vector of D. Note that a token $t \in Q$ can indicate the element of the sparse vector of D.

To learn sparse representations, SPLADE adopts the FLOPS regularizer (Paria et al., 2020). We give the formal expression of the FLOPS regularizer in Appendix A.2.

3.2 Combination of AdaLM and IDF

This section explains our proposed CAI method more precisely.

3.2.1 Executing AdaLM

CAI is a method addressing the vocabulary and word-frequency gap between datasets without supervision data. We execute AdaLM before training SPLADE to fill this gap. AdaLM (Yao et al., 2021) is a UDA method for an MLM. It comprises vocabulary expansion and continual pretraining using the corpus of the target domain.

We use AdaLM based on two assumptions. One is that we can consider that SPLADE expands queries and documents because the sparse vector encoded by SPLADE has non-zero elements corresponding to tokens that do not appear in a query or document. Thus, continual pretraining should allow SPLADE to expand queries and documents more accurately. In addition, vocabulary expansion should amplify the effect of continual pretraining. The other is that Jang et al. (2021) showed that the larger the dimension of sparse vectors, the better sparse retrieval performed in MRR@10 in the source domain. Vocabulary expansion means increasing dimensions of sparse vectors for SPLADE. Thus, we expect that vocabulary expansion should improve IR performance even on out-of-domain datasets.

Algorithm 1 Procedure for vocabulary expansion

INPUT: Original vocabulary $\mathcal{V}_0$, a domain corpus C, incremental vocabulary size ΔV
OUTPUT: $\mathcal{V}_{\text{final}}$
Set iterating index $i = 0$
repeat
  $i = i + 1$
  $\mathcal{V}_i = \mathcal{V}_{i-1}$
  Set target vocabulary size $V_i = |\mathcal{V}_0| + i * \Delta V$
  Build WordPiece tokenizer $\mathcal{T}_i$ at the vocabulary size of V_i on C.
  Get vocabulary $\acute{\mathcal{V}}_i$ from $\mathcal{T}_i$
  Tokenize C by $\mathcal{T}_i$ and count tokens
  Sort $\acute{\mathcal{V}}_i$ by frequency
  Set new vocabulary $\mathcal{V}_i$ by adding words to $\mathcal{V}_0$ from frequent words until $|\acute{\mathcal{V}}_i| < V_i$ except for duplicate words and words consisting of only number of mark.
until $|\mathcal{V}_i| - |\mathcal{V}_{i-1}| < \Delta V$
return $\mathcal{V}_{\text{final}} = \mathcal{V}_i$

In the last part of this subsection, we explain details of how to execute AdaLM.

Vocabulary Expansion AdaLM first expands the vocabulary of an MLM for more effective continual pretraining. To expand the vocabulary, AdaLM first builds a domain-specific tokenizer with WordPiece (Schuster and Nakajima, 2012) at a target vocabulary size. Then, AdaLM adds new words obtained by the built tokenizer to the original tokenizer. The addition starts from the most frequent words and stops when the vocabulary size of the tokenizer reaches the target vocabulary size. We exclude tokens composed of only numbers and marks (e.g., !,?,",[,]) because these tokens are considered as noise. We repeat this procedure, increasing the target vocabulary by 3k. Finally, we stop this increment when the vocabulary size of the tokenizer cannot reach the target vocabulary size. We summarize this procedure in Algorithm 1.

After adding words, AdaLM initializes the embeddings of these words. To obtain the embeddings, AdaLM tokenizes the added words to subwords by the original tokenizer, takes the average of embeddings of subwords, and then sets the averaged embeddings as initial vectors of newly added words.

Continual Pretraining Continual pretraining (Gururangan et al., 2020) is also a UDA method for an MLM. This method is straightforward; it further trains an MLM on a domain-specific corpus. Following BERT, we randomly mask 15% of tokens with a special token like [MASK] and let the model predict the original token.

3.2.2 Weighting Sparse Vectors with IDF

After training SPLADE, we multiply the SPLADE-encoded sparse vectors by IDF weights. Formal et al. (2022b) noted that SPLADE struggles with the exact matching of low-frequency words in the training data. In addition, the problem is amplified on out-of-domain datasets. Thus, we expect sparse vectors weighted with IDF to match the low-frequency words.

Now, we denote the number of documents in a target dataset as N and documents including token t as N_t. We express the IDF weight vector $\boldsymbol{w}^{\text{IDF}} \in \mathbb{R}^{|\mathcal{V}|}$ by the following equation:

$$\boldsymbol{w}_t^{\text{IDF}} = \begin{cases} \log \frac{N}{N_t} & \text{if } N_t \neq 0 \\ 1 & \text{otherwise} \end{cases}. \tag{6}$$

When $N_t = 0$, we set the weight as 1 so that the weight inferred by SPLADE does not change.

We can express the weighted sparse vector $\hat{\boldsymbol{s}} \in \mathbb{R}^{\mathcal{V}}$ by the following equation:

$$\hat{\boldsymbol{s}} = \boldsymbol{w}^{\text{IDF}} \odot \boldsymbol{s}. \tag{7}$$

where $\odot$ denotes the Hadamard product. Note that we apply the weighting only for document vectors.

3.3 Combination with Lexical Approach

Finally, we discuss the combination of our method with the lexical approach, which is an approach to enhance IR performance further. Previous works showed that lexical approaches and IR models based on an MLM are complementary (Luan et al., 2021; Gao et al., 2021). In addition, several works (Ma et al., 2021; Xu et al., 2022; Formal et al., 2022a) showed that simply adding or multiplying the scores of the lexical approach and an IR model based on an MLM improved the IR performance. Following these works, we also experimented with the adding case using BM25 for the lexical approach. We refer to this approach as **Hybrid**. Now, we denote a score of BM25 between a query Q and a document D as $S_{\text{BM25}}(Q, D)$. Formally, for both $S_{\text{SPL}}(Q, D)$ and $S_{\text{SPL-D}}(Q, D)$, the scores of Hybrid, $S_{\text{H-SPL}}(Q, D)$ and $S_{\text{H-SPL-D}}(Q, D)$, are

$$S_{\text{H-SPL}}(Q, D) = S_{\text{BM25}}(Q, D) + S_{\text{SPL}}(Q, D), \quad (8)$$

$$S_{\text{H-SPL-D}}(Q, D) = S_{\text{BM25}}(Q, D) + S_{\text{SPL-D}}(Q, D). \quad (9)$$

4 Experimental Setup

We confirm the effectiveness of our proposed CAI through experimental results. First, we introduce baselines. They show us how effective CAI is. Second, we explain IR datasets and domain corpora. The last subsection gives details of the implementation.[1]

4.1 Baselines

To measure the effectiveness of our approach, we compared it with other IR models. First, we chose dense retrieval (Karpukhin et al., 2020; Xiong et al., 2021), Cross Encoder (Nogueira et al., 2019; MacAvaney et al., 2019) and LaPraDoR (Xu et al., 2022).

Dense retrieval converts queries and documents into dense vectors and calculates relevance scores by the inner product or cosine similarity of dense vectors. Following Reimers and Gurevych (2019), we used average pooling to obtain dense vectors and cosine similarity for calculating relevance scores.

Cross Encoder[2] lets an MLM infer relevance scores by inputting texts composed from concatenations of queries and documents. This method achieved the best performance in a study by Thakur et al. (2021). We explain Cross Encoder formally in Appendix A.1.

LaPraDoR adopts a kind of hybrid approach by multiplying the score of BM25 and dense retrieval. To the best of our knowledge, this approach showed the state-of-the-art result on the average performance of five benchmark datasets mentioned in the next subsection.

We use **BM25** (Robertson and Walker, 1994) and **docT5query** (Nogueira et al., 2019) as models using BoW representations for queries like SPLADE-Doc. BM25 is still a strong baseline (Thakur et al., 2021). DocT5query expands documents using a generative model in addition to BM25.

Note that we did not apply domain adaptation for these baselines. We quote the results of docT5query and Cross Encoder from Thakur et al. (2021).

As the baseline of another UDA method, we used **GPL** (Wang et al., 2021), a state-of-the-art UDA method for dense retrieval. We experimented by applying GPL to SPLADE and our dense retrieval model[3]. When we applied CAI for comparison with GPL in dense retrieval, we used weighted average pooling with IDF weights.

4.2 Datasets and Evaluation Measures

This study used part of BEIR (Thakur et al., 2021). BEIR is a benchmark dataset in a zero-shot case, where no supervision data are available in the target datasets. Following the setting of BEIR, we used MS MARCO (Nguyen et al., 2016) as a source domain dataset where massive supervision data are available. This means that all supervised IR models were trained using MS MARCO. We measured IR performance by nDCG@10 as BEIR. For datasets of target domains, we chose BioASK (B-ASK) (Tsatsaronis et al., 2015), NFCorpus (NFC) (Boteva et al., 2016), and TREC-COVID (T-COV) (Voorhees et al., 2021) from the biomedical domain and SCIDOCS (SDOCS) (Cohan et al., 2020) and SciFact (SFact) (Wadden et al., 2020) from science domain because they have the largest vocabulary gap from the source domain. We show the vocabulary gap in Appendix D.

We built domain-specific corpora of the biomedical and science domains for domain adaptation of an MLM. We align the domains to the target datasets. For the biomedical domain, we extracted abstracts from the latest collection of PubMed[4]. We removed abstracts with less than 128 words from the corpus, following PubmedBERT (Gu et al., 2021). The corpus size was approximately 17 GB. For the science domain, we used the abstracts of the S2ORC (Lo et al., 2020) corpus. We also excluded abstracts with less than 128 words from the corpus. The corpus size was approximately 7.3 GB. The resulting size of $\mathcal{V}_{final}$ was 71,694 words in

[1] Our code is available at https://github.com/meshidenn/CAI.git.

[2] The actual model is cross-encoder/ms-marco-MiniLM-L-6-v2.

[3] GPL used Margin-MSE as a loss function. The teacher model of Margin-MSE was cross-encoder/ms-marco-MiniLM-L-6-v2. When applying GPL to our dense retrieval model trained on MS MARCO, Negative examples were sampled from the top-50 results of the dense retrieval model and sentence-transformers/msmarco-MiniLM-L-6-v3. When applying GPL to SPLADE, we replaced our dense retrieval model with SPLADE.

[4] https://pubmed.ncbi.nlm.nih.gov/

the biomedical domain and 62,783 in the science domain.

4.3 Details of Model Training

To train SPLADE and dense retrieval, we used Margin-MSE as a loss function[5]. Negative documents used in Margin-MSE were retrieved by BM25 or other retrieval methods as hard negative samples[6]. The loss of SPLADE[7] was the sum of Margin-MSE and FLOPS regularizers. The regularization weight of FLOPS for the query side λ_Q and document side λ_D were set as $\lambda_Q = 0.08$ and $\lambda_D = 0.1$, respectively, following Formal et al. (2021). Note that SPLADE-Doc was only used when searching, not training. We trained SPLADE and dense retrieval on one NVIDIA A100 40 GB GPU.

For continual pretraining, we began from bert-base-uncased[8] and conducted training on eight NVIDIA A100 40 GB. We set the batch size to 32 per device and trained one epoch.

For GPL, we generated queries for each document with docT5query (Nogueira et al., 2019)[9] using top-k and nucleus sampling (top-k: 25; top-p: 0.95). Following Wang et al. (2021), we sampled three queries per document and limited the size of the target IR corpus to 1M to reduce the computational cost when generating queries.

We give other parameters related to training models in Appendix C.

5 Results

This section compares our method with baselines. We first show the results of our approach and other IR methods. Next, we present the results of CAI and GPL as a comparison of UDA methods.

5.1 Comparison with other IR Methods

Table 1 lists the results of our method and other IR models. First, SPLADE with CAI outperformed SPLADE on all datasets. In addition, our approach showed comparable performance with Cross Encoder on the average of nDCG@10 for all datasets. These results illustrate that our method effectively fills the vocabulary and word-frequency gap for IR. Note that SPLADE can realize faster retrieval than Cross Encoder because SPLADE only has to encode queries, not concatenations of queries and documents.

Next, SPLADE-Doc with CAI scored best on four of five datasets in other methods using BoW representations of queries. In addition, SPLADE-Doc with CAI outperformed SPLADE on all datasets. This result suggests that our approach performs quite well for BoW representations and is as fast as BM25 when searching [10].

Finally, Hybrid-SPLADE with CAI achieved the best on the average of nDCG@10 for all datasets and outperformed LaPraDor. However, on some datasets, LaPraDor scored higher. This implies that sparse retrieval and dense retrieval learn different aspects of IR. It seems necessary in future work to research a more effective method of utilizing the complementarity of dense and sparse representations.

5.2 Comparison of Unsupervised Domain Adaptation Methods

Next, we compare CAI with GPL, a state-of-the-art UDA method. Table 2 shows the results of comparing CAI and GPL on dense retrieval, SPLADE, and SPLADE-Doc. For all IR models in Table 2, our method outperformed GPL. This result shows that our method is suitable for the domain shift in vocabulary and word frequencies. Focusing on the performance difference, it was large for SPLADE and SPLADE-Doc but small for dense retrieval. This result suggests that our approach is more effective for sparse retrieval. Note that GPL deteriorates the performance of SPLADE-Doc. Our approach seems more robust for query representation in SPLADE than GPL.

6 Ablation with AdaLM for Confirming Assumption

We conducted ablation studies for AdaLM to confirm the assumptions presented in Section 3.2.

[5]We introduce the formal expression of Margin-MSE in Appendix A.1. We used cross-encoder/ms-Marco-MiniLM-L-6-v2 as the teacher model for Margin-MSE.

[6]We used negative documents distributed by sentence transformers website https://huggingface.co/datasets/sentence-transformers/msmarco-hard-negatives/resolve/main/msmarco-hard-negatives.jsonl.gz

[7]We introduce the formal expression of SPLADE loss in AppeindixA

[8]https://huggingface.co/bert-base-uncased

[9]Actual model is BeIR/query-gen-msmarco-t5-base-v1

[10]We also checked the sparseness of SPLADE with CAI on the NFCorpus. The average of nonzero elements of SPLADE with CAI is 291.7, though the average document length is 175.5 with the pyserini analyzer. We consider this number to be sufficiently sparse to utilize an inverted index.

Table 1: Evaluation of our methods and other IR models by nDCG@10. The best results are in **bold**. The best results in the same category are in *italics*.

	Biomedical			Science		
	B-ASK	NFC	T-COV	SDOCS	SFact	Ave
Dense	0.377	0.301	0.716	0.144	0.571	0.422
SPLADE	0.503	0.336	0.627	0.155	0.691	0.462
Cross Encoder	0.523	0.350	*0.757*	*0.166*	0.688	*0.497*
SPLADE with CAI (Ours)	*0.544*	*0.353*	0.719	0.161	*0.708*	*0.497*
Bag-of-words representations of queries						
BM25	0.515	0.335	0.581	0.148	0.674	0.451
docT5query	0.431	0.328	*0.713*	*0.162*	0.675	0.462
SPLADE-Doc	0.488	0.323	0.539	0.147	0.678	0.435
SPLADE-Doc with CAI (Ours)	*0.551*	*0.342*	0.633	*0.162*	*0.715*	*0.480*
Hybrid with Lexical Approach						
LaPraDor	0.511	0.347	**0.779**	**0.185**	0.697	0.504
Hybrid-SPLADE-Doc with CAI (Ours)	0.567	0.347	0.680	0.162	0.714	0.494
Hybrid-SPLADE with CAI (Ours)	**0.573**	**0.357**	0.756	0.165	**0.716**	**0.514**

Table 2: Evaluation of our methods and GPL by nDCG@10. The best results are in **bold**. The best results in the same category are in *italics*.

	Biomedical			Science		
	B-ASK	NFC	T-COV	SDOCS	SFact	Ave
Dense Retrieval						
Original	0.377	0.301	0.716	0.144	0.571	0.422
GPL	*0.420*	0.325	0.723	*0.162*	*0.654*	0.457
CAI	0.411	*0.329*	**0.760**	0.148	0.648	*0.459*
SPLADE						
Original	0.503	0.336	0.627	0.155	0.691	0.462
GPL	0.513	0.319	0.708	**0.171**	0.676	0.477
CAI	*0.544*	**0.353**	*0.719*	0.161	*0.708*	**0.497**
SPLADE-Doc						
Original	0.488	0.323	0.539	0.147	0.678	0.435
GPL	0.491	0.305	0.562	0.153	0.649	0.432
CAI	**0.551**	*0.342*	*0.633*	*0.162*	**0.715**	*0.480*

Table 3: Ablation study using AdaLM by nDCG@10. We use SPLADE as a base model. The best results are in **bold**.

	Biomedical	Science	All
SPLADE	0.489	0.423	0.462
Ablation to AdaLM			
Continual Pretraining	0.509	0.426	0.476
Vocabulary Expansion	0.493	0.416	0.462
Scratch Models	0.000	**0.446**	0.178
AdaLM	**0.528**	0.426	**0.491**

Precisely, we considered the case of only applying vocabulary expansion or continual pretraining. In addition, we also used models trained on a domain-specific corpus from scratch. By comparing AdaLM with continual pretraining, we confirm whether IR performance is further enhanced by expanding the vocabulary. In addition, we observe the effect of vocabulary size based on the results achieved by vocabulary expansion and the scratch models. As scratch models, we used PubMedBERT [11] (Gu et al., 2021) for the biomedical domain and SciBERT [12] (Beltagy et al., 2019) for the science domain.

Table 3 lists the result of the ablation study using SPLADE. First, continual pretraining improved IR performance over the original SPLADE. In addition, SPLADE with AdaLM outperformed continual pretraining. These results support that vocabulary expansion enhances the effect of continual pretraining.

However, expanding vocabulary cannot improve the IR performance on average. In addition, the scratch model of the science domain outperformed AdaLM. Note that the vocabulary size of scratch is the same with the original BERT. These results show that larger dimensions themselves cannot help improve IR performance when no supervision data are available and that accurate query and document expansion is more important.

By contrast, the scratch model of the biomedical domain failed to learn SPLADE. Thus, scratch models on the domain-specific corpus may not learn SPLADE, and AdaLM seems a more stable method than the scratch models.

We further analyzed the effect of vocabulary size

[11] microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract

[12] https://huggingface.co/allenai/scibert_scivocab_uncased

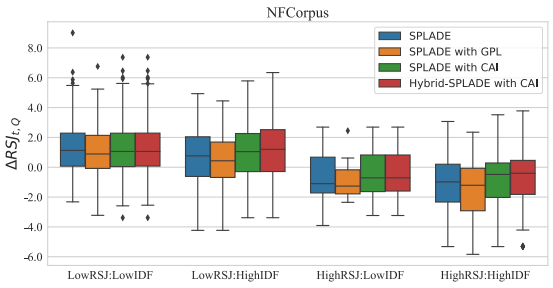

Figure 2: $\Delta\text{RSJ}_{t,Q}$ of SPLADE, SPLADE with GPL, SPLADE with CAI, and Hybrid-SPLADE with CAI.

Table 4: An example of top-ranked documents for a query including a HighRSJ and HighIDF word. The example is from NFCorpus. The top-ranked document by SPLADE with CAI is correct. The HighRSJ and HighIDF words appearing in the query and document are labeled in **bold**.

Query	**Phytate**s for the Treatment of Cancer
Top-ranked documents	
SPLADE with CAI	Dietary suppression of colonic cancer. Fiber or **phytate**? The incidence of colonic cancer differs widely ...
SPLADE	Phytochemicals for breast cancer prevention by targeting aromatase. Aromatase is a cytochrome P450 enzyme ...

on AdaLM in Section E. The result suggests that AdaLM with a larger vocabulary size tended to perform better in nDCG@10 for SPLADE.

7 Analysis for Weight of Words

This section shows whether our method can solve the problem of exact matching of low-frequency words.

Formal et al. (2022b) analyzed IR models with an MLM, using Robertson Spärck Jones (RSJ) weight (Robertson and Spärck Jones, 1994). RSJ weight measures how a token can distinguish relevant from non-relevant documents in an IR corpus. This weight also indicates an ideal weight in terms of lexical matching. We denote RSJ weight as $\text{RSJ}_{t,Q}$ for a tokenized query $Q \in \mathcal{V}^l$ and a token $t \in Q$. To infer the RSJ weight of the IR models, Formal et al. (2022b) replaced relevant documents with top-K documents retrieved by the model. We express the inferred RSJ weight as $\widehat{\text{RSJ}}_{t,Q}$. We set $\text{K} = 100$, following the authors. We give the formal expression of $\text{RSJ}_{t,Q}$ and $\widehat{\text{RSJ}}_{t,Q}$ in Appendix B.

Following Formal et al. (2022b), we take the difference between $\text{RSJ}_{t,Q}$ and $\widehat{\text{RSJ}}_{t,Q}$, i.e.,

$$\Delta\text{RSJ}_{t,Q} = \text{RSJ}_{t,Q} - \widehat{\text{RSJ}}_{t,Q}, \qquad (10)$$

as an indicator to measure the gap between the ideal RSJ weight and RSJ weight of the models. If $\Delta\text{RSJ}_{t,Q} > 0$, an IR model overestimates the weights of tokens. Conversely, if $\Delta\text{RSJ}_{t,Q} < 0$, an IR model underestimates the weight of the tokens. For analysis, we also divide all tokens into HighRSJ and LowRSJ at the 75-percentile. Furthermore, we split all tokens into groups of HighIDF and LowIDF at the median IDF of all tokens in queries. This analysis is conducted on the NFCorpus. The tokenizer used is the analyzer of pyserini [13], which processes porter stemming and removes some stopwords.

Figure 2 shows the $\Delta\text{RSJ}_{t,Q}$ of SPLADE, SPLADE with GPL, SPLADE with CAI, and Hybrid SPLADE with CAI. First, SPLADE with CAI underestimates the RSJ weight less than SPLADE in the groups of HighRSJ and HighIDF. In addition, Hybrid-SPLADE with CAI is closer to $\Delta\text{RSJ}_{t,Q} = 0$ than SPLADE with CAI on the same groups. This result suggests that our approach solves the problem of term matching for rare words. By contrast, SPLADE with GPL shows lower $\Delta\text{RSJ}_{t,Q}$ than SPLADE. GPL seems to accelerate the problem of term matching for low-frequency words. As a result, GPL may lead to lower IR performance than SPLADE, as shown in Table 2.

8 Case Study

Finally, we confirm the case where SPLADE with CAI improves the IR performance by matching important and rare words, i.e., HighRSJ and HighIDF words. Table 4 shows a pair of a query including a HighRSJ and HighIDF word and top-1 documents in NFCorpus retrieved by SPLADE with CAI and SPLADE. In the example query, phytate is a HighRSJ and HighIDF word. SPLADE with CAI ranks a correct document, including phytate, at the top. However, the top-ranked document by SPLADE does not include phytate and is incorrect. The document frequency of phytate is bottom 2% in MS MARCO. This example supports that SPLADE with CAI successfully matches rare words in training data and can rank a correct document higher.

[13] https://github.com/castorini/pyserini

9 Conclusion

This paper presented an effective unsupervised domain adaptation method, CAI. We showed that the combination of SPLADE with CAI and the lexical approach gave a state-of-the-art performance on datasets with a large vocabulary and word-frequency gap. In addition, CAI outperformed GPL and was robust enough to show high accuracy even when BoW representations were used for query expression. Finally, our analysis showed that SPLADE with CAI addressed the problem of the exact matching of low-frequency words in training data. We believe that CAI works on smaller MLMs by distilling AdaLM because Yao et al. (2021) showed that a distilled AdaLM achieved higher performance than BERT on NLP tasks and Formal et al. (2021) showed that the results of SPLADE initialized with DistilBERT-base[14] was competitive on MS MARCO with other IR models initialized with BERT.

Integrating sparse and dense retrieval is a promising way to enhance IR performance further on out-of-domain datasets. Future work will integrate them to reveal the factors contributing to IR.

Acknowledgements

We are grateful to the anonymous reviewers, Masahiro Kaneko, Sakae Mizuki and Ayana Niwa who commented our paper. This paper is based on results obtained from a project, JPNP18002, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Alexander Bondarenko, Maik Fröbe, Meriem Beloucif, Lukas Gienapp, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. 2020. Overview of Touché 2020: Argument Retrieval: Extended Abstract. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings*, page 384–395. Springer-Verlag.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In *Advances in Information Retrieval*, pages 716–722.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022a. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. *arXiv Preprint*, arXiv:2205.04733.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2022b. Match Your Words! A Study of Lexical Matching in Neural Information Retrieval. In *Advances in Information Retrieval*, pages 120–127, Cham. Springer International Publishing.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17(59):1–35.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement Lexical Retrieval Model with Semantic Residual Embeddings. In *Advances in Information Retrieval*, pages 146–160. Springer International Publishing.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *ACM Trans. Comput. Healthcare*, 3(1).

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining:

[14] https://huggingface.co/distilbert-base-uncased

Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

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 *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 1265–1268. Association for Computing Machinery.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. *arXiv Preprint*, arXiv:2010.02666.

Jimin Hong, TaeHee Kim, Hyesu Lim, and Jaegul Choo. 2021. AVocaDo: Strategy for Adapting Vocabulary to Downstream Domain. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4692–4700, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kyoung-Rok Jang, Junmo Kang, Giwon Hong, Sung-Hyon Myaeng, Joohee Park, Taewon Yoon, and Heecheol Seo. 2021. Ultra-high dimensional sparse representations with binarization for efficient text retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1016–1029, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Constantinos Karouzos, Georgios Paraskevopoulos, and Alexandros Potamianos. 2021. UDALM: Unsupervised Domain Adaptation through Language Modeling. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2579–2590, Online. Association for Computational Linguistics.

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)*, pages 6769–6781, Online. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur 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. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Markus Leippold and Thomas Diggelmann. 2020. Climate-FEVER: A Dataset for Verification of Real-World Climate Claims. In *NeurIPS 2020 Workshop on Tackling Climate Change with Machine Learning*.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The Semantic Scholar Open Research Corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

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*, 9:329–345.

Ji Ma, Ivan Korotkov, Yinfei Yang, Keith Hall, and Ryan McDonald. 2021. Zero-shot Neural Passage Retrieval via Domain-targeted Synthetic Question Generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1075–1088, Online. Association for Computational Linguistics.

Sean MacAvaney, Arman Cohan, and Nazli Goharian. 2020. SLEDGE-Z: A Zero-Shot baseline for COVID-19 literature search. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4171–4179, Online. Association for Computational Linguistics.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 1101–1104. Association for Computing Machinery.

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 *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1941–1942. International World Wide Web Conferences Steering Committee.

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 *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona,*

Spain, December 9, 2016, volume 1773 of *CEUR Workshop Proceedings*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *arXiv Preprint*, arXiv:1904.08375.

Biswajit Paria, Chih-Kuan Yeh, Ian E.H. Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing FLOPs to Learn Efficient Sparse Representations. In *International Conference on Learning Representations*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

S. E. Robertson and S. Walker. 1994. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, page 232–241, Berlin, Heidelberg. Springer-Verlag.

S.E. Robertson and K. Spärck Jones. 1994. Simple, proven approaches to text retrieval. Technical Report UCAM-CL-TR-356, University of Cambridge, Computer Laboratory.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

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)*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16:138.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. TREC-COVID: Constructing a Pandemic Information Retrieval Test Collection. *SIGIR Forum*, 54(1).

Ellen M Voorhees. 2004. Overview of the TREC 2004 Robust Retrieval Track. In *TREC*.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018. Retrieval of the Best Counterargument without Prior Topic Knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251, Melbourne, Australia. Association for Computational Linguistics.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online. Association for Computational Linguistics.

Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, and Dong Yu. 2019. Improving Pre-Trained Multilingual Model with Vocabulary Expansion. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 316–327, Hong Kong, China. Association for Computational Linguistics.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2021. GPL: Generative Pseudo Labeling for Unsupervised Domain Adaptation of Dense Retrieval. *arXiv Preprint*, arXiv:2112.07577.

Ji Xin, Chenyan Xiong, Ashwin Srinivasan, Ankita Sharma, Damien Jose, and Paul Bennett. 2022. Zero-Shot Dense Retrieval with Momentum Adversarial Domain Invariant Representations. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4008–4020, Dublin, Ireland. Association for Computational Linguistics.

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 *International Conference on Learning Representations*.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2022. LaPraDoR: Unsupervised Pretrained Dense Retriever for Zero-Shot Text Retrieval. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3557–3569, Dublin, Ireland. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Yunzhi Yao, Shaohan Huang, Wenhui Wang, Li Dong, and Furu Wei. 2021. Adapt-and-Distill: Developing Small, Fast and Effective Pretrained Language Models for Domains. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 460–470, Online. Association for Computational Linguistics.

A Loss Function and Regulalizer

This section shows the formal expression of margin mean squared error (Margin-MSE), $\mathcal{L}_{\text{Margin-MSE}}$, and FLOPS regularizer, $\mathcal{L}^{\text{FLOPS}}$. The loss function of training SPLADE, $\mathcal{L}_{\text{SPL}}$, is

$$\mathcal{L}_{\text{SPL}} = \mathcal{L}_{\text{Margin-MSE}} + \lambda_Q \mathcal{L}^{Q}_{\text{FLOPS}} + \lambda_D \mathcal{L}^{D}_{\text{FLOPS}}, \tag{11}$$

where $\mathcal{L}^{Q}_{\text{FLOPS}}$ is a FLOPS regualizer of a query side and $\mathcal{L}^{D}_{\text{FLOPS}}$ is a document side

A.1 Margin Mean Squared Error

Margin-MSE (Hofstätter et al., 2020) can be used to distill knowledge from Cross Encoder. Cross Encoder inferrs relevance scores by inputting concatenated queries and documents to an MLM. Now, we denote a tokenized query as $Q \in \mathcal{V}^l$ and tokenized document as $D \in \mathcal{V}^m$. l and m are the lengths of Q and D, respectively. We express the concatenated text as $C_{Q,D} = [\text{CLS}; Q; \text{SEP}; D; \text{SEP}] \in \mathcal{V}^{m+l+3}$ and the process encoding the CLS token to a d-dimensinal vector as $\text{BERT}(C_{Q,D})_{\text{CLS}}$. The inferred score is calculated by the dot product of the vector of the CLS token and linear layer $\boldsymbol{W}_{\text{CLS}} \in \mathbb{R}^{d \times d}$ and a bias term of the layer $\boldsymbol{b}_{\text{CLS}} \in \mathbb{R}^d$. Then, the score $S_{\text{CE}}(Q, D)$ is

$$S_{\text{CE}}(Q, D) = \boldsymbol{W}^{\top}_{\text{CLS}} \text{BERT}(C_{Q,D})_{\text{CLS}} + \boldsymbol{b}_{\text{CLS}}. \tag{12}$$

Now, we assume a batch of size B. Then we express a query in the batch as Q_i, a positive document to the query as D^+_i, and a negative document as D^-_i. The difference of score δ_i between D^+_i and D^-_i by Cross Encoder is

$$\delta_i = S_{\text{CE}}(Q_i, D^+_i) - S_{\text{CE}}(Q_i, D^-_i). \tag{13}$$

Next, we express a target model for training as M and a score inferred by the model between Q and D as $S_M(Q, D)$. The difference of scores between D^+_i and D^-_i by M is

$$\hat{\delta}_i = S_M(Q_i, D^+_i) - S_M(Q_i, D^-_i). \tag{14}$$

Finally, we can express Margin-MSE by the following equation:

$$\mathcal{L}_{\text{Margin-MSE}} = \frac{1}{B} \sum_{i=1}^{B} (\delta_i - \hat{\delta}_i)^2. \tag{15}$$

A.2 FLOPS Regularizer

FLOPS (Paria et al., 2020) regularizer induces sparseness to encoded vectors by neural models. We denote a query matrix as $\boldsymbol{Q} \in \mathbb{R}^{B \times l}$, which consists of l-dimensional vectors of queries with batsh size B. In the same way, we denote a document matrix as $\boldsymbol{D} \in \mathbb{R}^{B \times l}$. The formal expressions of FLOPS loss are

$$\mathcal{L}^{Q}_{\text{FLOPS}} = \sum_{j=1}^{l} \left(\frac{1}{B} \sum_{i=1}^{B} |\boldsymbol{Q}_{i,j}| \right)^2 \tag{16}$$

$$\mathcal{L}^{D}_{\text{FLOPS}} = \sum_{j=1}^{l} \left(\frac{1}{B} \sum_{i=1}^{B} |\boldsymbol{D}_{i,j}| \right)^2. \tag{17}$$

B Robertson Spärck Jones Weight

Formal et al. (2022b) analyzed IR models with an MLM, using Robertson Spärck Jones (RSJ) weight (Robertson and Spärck Jones, 1994). RSJ weight measures how a token can distinguish relevant from non-relevant documents in an IR corpus. The weight is inferred by pairs of a query Q and correct documents. We denote a token of the query as t. Formally, the RSJ weight is

$$\text{RSJ}_{t,Q} = \log \frac{p(t|R_Q) p(\neg t|\neg R_Q)}{p(\neg t|R_Q) p(t|\neg R_Q)}. \tag{18}$$

R_Q is a set of relevant documents for a query Q. $p(t|R_Q)$ is the probability that relevant documents have token t. $p(t|\neg R_Q)$ is the probability that non-relevant documents have a token t. Lastly, $p(\neg t|R_Q) = 1 - p(t|R_Q)$ and $p(\neg t|\neg R_Q) = 1 - p(t|\neg R_Q)$.

To investigate the RSJ weight of IR models, the authors proposed the following modification:

$$\widehat{\text{RSJ}}_{t,Q} = \log \frac{p(t|\hat{R}^K_Q) p(\neg t|\neg \hat{R}^K_Q)}{p(\neg t|\hat{R}^K_Q) p(t|\neg \hat{R}^K_Q)}. \tag{19}$$

Table 5: Hyper-parameters of dense retrieval

Batch size	64
Max document length	300
Learning rate	2e-5
Epoch	30
Warmup steps	1000

Table 6: Hyper-parameters of SPLADE

Batch size	40
Max document length	256
Learning rate	2e-5
Epoch	30
Warmup steps	1000

Table 7: Hyper-parameters when using GPL

Batch size	24
Max document length	350
Learning rate	2e-5
Training steps	140000
Warmup steps	1000

$\hat{R}_Q^K$ represents the top-K documents retrieved for the query Q by an IR model. $p(t|\hat{R}_Q^K)$ is the probability that the top-K documents retrieved by the IR model include the token t. $p(t|\neg\hat{R}_Q^K)$ is the probability that top-K documents not retrieved by the IR model include the token t. Lastly, $p(\neg t|\hat{R}_Q^K) = 1 - p(t|\hat{R}_Q^K)$ and $p(\neg t|\neg\hat{R}_Q^K) = 1 - p(t|\neg\hat{R}_Q^K)$.

C HyperParameters

We give show hyperparameters for training the models in Tables 5, 6, and 7.

D Vocabulary Gap from MS MARCO

Following Thakur et al. (2021), we calculated weighted Jaccard similarity $J(A, B)$ between a source dataset A and target dataset B in BEIR [15]. $J(A, B)$ is calculated by the following equation:

$$J(A, B) = \frac{\sum_t \min(A_t, B_t)}{\sum_t \max(A_t, B_t)}. \quad (20)$$

Here, A_t is the normalized frequency of word t in a source dataset, and B_t is in a target dataset. We used MS MARCO as a source dataset. Table 8 lists the results. We can observe that the five datasets we chose for our experiment were the most dissimilar to MS MARCO.

[15]The target datasets were ArguAna (Wachsmuth et al., 2018), BioASK, Climate-FEVER (Leippold and Diggelmann, 2020), DBPedia-Entity (Hasibi et al., 2017), FEVER (Thorne et al., 2018), FiQA (Maia et al., 2018), HotpotQA (Yang et al., 2018), Natural Question (Kwiatkowski et al., 2019), NFCorpus, Quora, Robust04 (Voorhees, 2004), SCIDOCS, SciFact, TREC-COVID, and Touché-2020 (Bondarenko et al., 2020)

Table 8: Weighted Jaccard similarity between a target dataset in BEIR and MS MARCO

Dataset	$J(S, T)$
Natural Question	0.523
Robust04	0.475
Touché-2020	0.410
FiQA	0.407
Quora	0.395
ArguAna	0.385
Climate-FEVER	0.384
FEVER	0.384
HotpotQA	0.342
DBPedia-Entity	0.334
SCIDOCS	0.327
BioASK	0.317
TREC-COVID	0.315
NFCorpus	0.285
SciFact	0.273

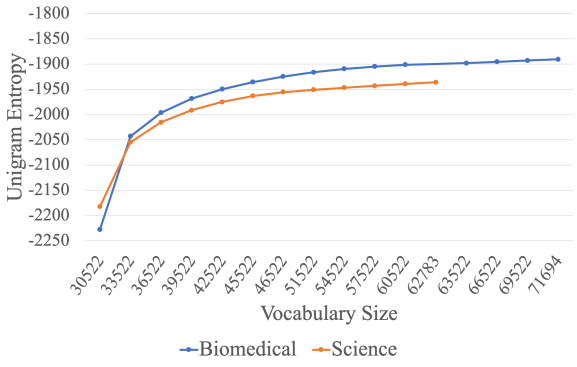

Figure 3: Unigram entropy of each vocabulary size on each domain corpus.

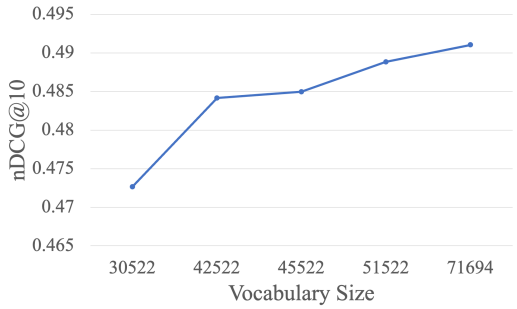

Figure 4: Performance variation with vocabulary size for SPLADE with AdaLM. Performance is measured by average nDCG@10 all datasets.

E Effect of Vocabulary Size

To confirm the effect of vocabulary size, we experimented with the case of smaller vocabulary sizes of AdaLM. To save the computational cost, we selected several vocabulary sizes, using unigram entropy criteria $I(C)$ of MLM training corpus C,

as by Yao et al. (2021). For a tokenizer with vocabulary $\mathcal{V}$, we calculated unigram probability $p(x)$ by counting the occurrence of each sub-word x in the corpus. Then, the unigram entropy $I(\boldsymbol{x})$ of each text sequence $\boldsymbol{x} = (x_1, x_2, .., x_L)$ can be calculated by following equation:

$$I(\boldsymbol{x}) = \sum_{i=1}^{L} \log(p(x_i)). \tag{21}$$

Now, we can describe the unigram entropy of the corpus $I(C)$ as

$$I(C) = \sum_{\boldsymbol{x} \in C} I(\boldsymbol{x}). \tag{22}$$

As mentioned in Section 3.2.1, we increment vocabulary size from the original BERT tokenizer. We denote the vocabulary of a tokenizer in a step as $\mathcal{V}_i$ and the unigram entropy of the tokenizer as $I_i(D)$.

We prepare three additional stopping critera of vocabulary expansion vocabulary. The first is $\frac{I_i(D) - I_{i-1}(D)}{I_{i-1}(D)} < \epsilon_1$. We set $\epsilon_1 = 0.01$, as used by Yao et al. (2021). The resulting vocabulary size was 42,522. Next, $I_i(D) - I_{i-1}(D)$ is largest in the first increment as shown in Figure 3. Thus, the next stopping criterion is $I_i(D) - I_{i-1}(D) < \epsilon_2(I_1(D) - I_0(D))$. We set the coefficient $\epsilon_2 = 0.1$ and $\epsilon_2 = 0.05$. As a result, the vocabulary sizes were 45,522 and 51,522, respectively.

We present the results of SPLADE with AdaLM on the average of nDCG@10 for all datasets in Figure 4. The figure shows the trend that the model of large vocabulary size performed better in nDCG@10.

F Interaction between In-Domain Supervision Data and CAI

We experimented in the case where in-domain supervision data were available to observe the effect of CAI with supervision data.

We trained SPLADE and SPLADE with CAI used in our main experiment further on NFCorpus because NFCorpus has the most training pairs of a query and a relevant document in the all target datasets. The loss function for the training was MultipleNegativeRankingLoss[16] (Henderson et al., 2017). Negative examples were sampled from BM25 and two dense retrieval models. One was the same with the model mentioned in Section 4. The other was trained on NFCorpus further from the first with negative examples from BM25. We did not use Margin-MSE loss in this experiment because SPLADE models trained with Margin-MSE [17] loss on NFCorpus degraded the performance. We changed λ_Q, λ_D, and batch size from the settings of Section 4. We set the batch size at 32. We used $\lambda_Q = 0.0006$ and $\lambda_D = 0.0008$, following Formal et al. (2021).

Table 9: Experimental results with and without supervision data of NFCorpus.

	nDCG@10
SPLADE	
Without Supervision	0.336
With Supervision	0.339
SPLADE with CAI	
Without Supervision	0.353
With Supervision	0.377

Table 9 shows the results. SPLADE with supervision data of NFCorpus certainly improved nDCG@10 over the case without supervision. However, the improvement of the performance was limited. In contrast, SPLADE with CAI and supervision data showed a larger improvement. Thus, adapting MLM to the target domain is also important for SPLADE when supervision data are available.

[16] https://www.sbert.net/docs/package_reference/losses.html#multiplenegativesrankingloss

[17] The model of cross encoder is cross-encoder/ms-marco-MiniLM-L-6-v2.