Lexicon-Enhanced Self-Supervised Training for Multilingual Dense Retrieval

Houxing Ren[1*] **Linjun Shou**[2] **Jian Pei**[3] **Ning Wu**[2] **Ming Gong**[2] **Daxin Jiang**[2†]
[1]School of Computer Science and Engineering, Beihang University
[2]Microsoft STC Asia
[3]Duke University, Durham, NC, USA 27705
renhouxing@buaa.edu.cn {lisho,wuning,migon,djiang}@microsoft.com j.pei@duke.edu

Abstract

Recent multilingual pre-trained models have shown better performance in various multilingual tasks. However, these models perform poorly on multilingual retrieval tasks due to lacking multilingual training data. In this paper, we propose to mine and generate self-supervised training data based on a large-scale unlabeled corpus. We carefully design a mining method which combines the sparse and dense models to mine the relevance of unlabeled queries and passages. And we introduce a query generator to generate more queries in target languages for unlabeled passages. Through extensive experiments on Mr. TYDI dataset and an industrial dataset from a commercial search engine, we demonstrate that our method performs better than baselines based on various pre-trained multilingual models. Our method even achieves on-par performance with the supervised method on the latter dataset.

1 Introduction

Information Retrieval (IR) aims to retrieve relevant passages for a given query, which plays a critical role in many industry scenarios such as Open-Domain Question Answering (QA) (Lee et al., 2019) and Web Search (Nguyen et al., 2016). Traditionally, bag-of-words (BOW) retrieval systems such as TF-IDF and BM25 (Robertson and Zaragoza, 2009) were widely used, which mainly depend on keyword matching between queries and passages. With the development of large-scale pre-trained language models (PLMs) (Vaswani et al., 2017; Devlin et al., 2019) such as BERT, dense retrieval methods (Lee et al., 2019; Karpukhin et al., 2020) show quite effective performance. These methods usually employed a dual-encoder architecture to encode both queries and passages into dense embeddings and then perform approximate nearest neighbor searching (Johnson et al., 2021).

Recently, some works found that dense retrievers perform poorly in the zero-shot multilingual settings (Zhang et al., 2021b) due to the distributional shift. To boost the performance of dense retrievers, some previous methods for cross-domain retrieval can be directly adopted to unsupervised multilingual dense retrieval. There are two important kinds: 1) generating training data in target languages. For example, Kulshreshtha *et al.* applied self-training to generate labeled data and further proposed back-training (Kulshreshtha et al., 2021) to obtain more high-quality data. QGen (Ma et al., 2021a) proposed to use a query generator to generate in-domain queries. 2) leveraging sparse retrievers, which is more effective in the unsupervised setting, to enhance dense retrievers. For example, SPAR (Chen et al., 2021) proposed to distill knowledge from BM25 to the dense model and LaPraDoR (Xu et al., 2022) proposed to enhance the dense model by multiplying the similarity with the BM25 score.

However, there are three major problems when directly adopting these methods to multilingual dense retrieval. First, zero-shot multilingual query generators suffer from grammatical adjustment and accidental translation problems (Xue et al., 2021). As a result, zero-shot query generators only provide little help in bridging the gap among different languages. Second, hybrid dense and sparse models such as LaPraDoR and SPAR get high latency in the inference stage[1]. Finally, dense retrieval is different from other tasks, it not only needs positive query-passage pairs but also needs negative query-passage pairs (Xiong et al., 2021). However, previous methods such as the back-training focus on positive pairs and simply take the top passages of BM25 as negative passages.

Although training data in target languages is very expensive, unlabeled queries and passages can be

[*]Work done during internship at Microsoft STCA.
[†]Corresponding author.

[1]The latency of dense retriever on GPU is 32ms and the latency of BM25 on CPU is 36ms (Gao et al., 2021).

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 444–459
December 7-11, 2022

easily obtained from search engines such as *Google* and *Bing*. In this paper, we propose a novel method that augments data in target languages by combining sparse and dense models, namely LeSTM, which stands Lexicon-enhanced Self-supervised Training for Multilingual dense retrieval. First, as we mentioned above, sparse retrievers mainly depend on keyword matching between queries and passages and dense retrievers mainly depend on the language modeling ability of pre-trained models, which indicates the sparse and dense models perform retrieval in different aspects (Chen et al., 2021). In addition, the sparse–dense hybrid retriever is significantly better than both sparse and dense models (Zhang et al., 2021b; Ma et al., 2021b). Both can demonstrate that sparse and dense models notice different characteristics and are complementary. Therefore, we craft a lexicon-enhanced retrieval module to mine positive and negative passages for each unlabeled query in target languages, which leverages the retrieval results of both sparse and dense models. We treat passages that both sparse and dense models regard are relevant as positive passages, and passages that one model regards are relevant but the other regards are irrelevant as negative passages.

Furthermore, we employ a query generator to generate queries for passages in target languages due to the limited number of unlabeled queries. The query generation methods have been shown to significantly improve the performance of retrieval models in the monolingual setting (Kulshreshtha et al., 2021; Ma et al., 2021a). Considering the grammatical adjustment and accidental translation problems, we first use the mined positive query-passage pairs to train a query generator. Then, we use the trained model to generate more queries in target languages. Considering that there may exist more relevant passages to the generated queries, we use both sparse and dense retrievers to filter the generated samples. Finally, using only unlabeled data from target languages, LeSTM iteratively mines query passage pairs by the lexicon-enhanced retriever and generator, trains a new better retriever and query generator using these mined pairs, mines again for better query passage pairs, and repeats.

In summary, our contributions are as follows.

- To the best of our knowledge, our approach is the first attempt to combine sparse and dense retrievers to mine high-quality positive and negative query-passage pairs for the multilingual dense retriever.
- We propose to use a query generator to expand the unlabeled queries in target languages and an iterative training paradigm is introduced to further enhance the dense retriever and generator.
- Extensive experiments on two datasets show the effectiveness of our proposed approach. In particular, experiments demonstrate that our method is model-agnostic, they are effective on various pre-trained language models.

2 Related Work

Information Retrieval. Information retrieval aims to search relevant passages from a large corpus for a given query. Traditionally, researchers use bag-of-words (BOW) based methods such as TF-IDF and BM25 (Robertson and Zaragoza, 2009). These methods use a sparse vector to represent the text, so we call them sparse retrievers. Recently, some studies use neural networks to improve the sparse retriever. For example, DocTTTTTQuery (Nogueira et al., 2019) proposes to expand the document to narrow the vocabulary gap and DeepCT (Dai and Callan, 2020) generates a weight for each term to emphasize the import terms.

In contrast to sparse retrievers, dense retrievers usually encode both queries and passages into dense vectors whose lengths are much less than sparse vectors. There are two kinds of dense retrieval methods: 1) pre-training with unlabeled data and 2) fine-tuning with labeled data. For pre-training, ORQA (Lee et al., 2019) proposes Inverse Cloze Task (ICT) which aims to predict the context of a given sentence, and REALM (Guu et al., 2020) proposes to predict the masked text based on an end-to-end retriever-reader model. Furthermore, SEED (Lu et al., 2021), Condenser (Gao and Callan, 2021a), and coCondenser (Gao and Callan, 2021b) propose pre-training tasks to encode more information into the dense vectors. For fine-tuning, one major method is how to incorporate hard negative samples during training, including static sparse hard negative samples (Karpukhin et al., 2020; Luan et al., 2021) and dynamic dense hard negative samples (Xiong et al., 2021; Zhan et al., 2021). Another major method is training the retriever with a cross-attention encoder jointly, including extractive reader (Yang and Seo, 2020), generative reader (Izacard and Grave, 2021), and cross-encoder re-ranker (Qu et al., 2021; Ren et al.,

2021; Zhang et al., 2021a). In addition, some works trade time for performance by using multiple vectors to represent the passage (Khattab and Zaharia, 2020; Humeau et al., 2020; Tang et al., 2021; Zhang et al., 2022a).

Cross-lingual (domain) Retrieval. These tasks aims to investigate the retrieval capabilities under cross-lingual (Zhang et al., 2021b; Asai et al., 2021a) or cross-domain (Thakur et al., 2021) setting. The methods for these tasks can be divided into two main categories: model transfer methods and data transfer methods.

The model transfer methods for cross-domain focus on pre-training sentence representation. For example, GTR (Ni et al., 2021) and CPT (Neelakantan et al., 2022) propose that scaling up the model size can significantly improve the performance of dense models. Contriever (Izacard et al., 2021) and LaPraDoR (Xu et al., 2022) propose to use contrastive learning to learn sentence aware representation. For cross-lingual, they focus on learning multilingual representations by pre-training (Conneau and Lample, 2019; Chi et al., 2021; Feng et al., 2022) such as mBERT (Devlin et al., 2019) and XLMR (Conneau et al., 2020).

The data transfer methods mainly focus on obtaining noisy training data in the target domain or target languages. For example, Back-training (Kulshreshtha et al., 2021) and QGen (Ma et al., 2021a) propose to use a query generator to generate in-domain queries. CORA (Asai et al., 2021b) leverages a generator to help mine retrieval training data and DR.DECR (Li et al., 2021) mines lots of parallel data to perform cross-lingual distillation.

3 Preliminaries

In this section, we give a brief review of dense retrieval and then present how to apply models to multilingual dense retrieval.

Overview. Given a query q and a corpus C, the retrieval task aims to find the relevant passages for the query from a large corpus. Usually, a dense retrieval model employs two dense encoders (*i.e.,* BERT) $E_Q(\cdot)$ and $E_P(\cdot)$. They encode queries and passages into dense embeddings, respectively. Then, the model uses a similarity function, often dot-product, to perform retrieval:

$$f(q, p) = E_Q(q) \cdot E_P(p), \tag{1}$$

where f denotes the similarity function, q and p denote the query and the passage, respectively. During the inference stage, we apply the passage encoder $E_P(\cdot)$ to all the passages and index them using FAISS (Johnson et al., 2021) which is an extremely efficient, open-source library for similarity search. Then given a query q, we derive its embedding by $\boldsymbol{v}_q = E_Q(q)$ and retrieve the top k passages with embeddings closest to $\boldsymbol{v}_q$.

Training. The training of retrieval is metric learning essentially. The goal is to narrow the distance between the query and the relevant passages (*a.k.a.,* positive passages) and widen the distance between the query and the irrelevant passages (*a.k.a.,* negative passages). Let $\{q_i, p_i^+, p_{i,0}^-, p_{i,1}^-, \ldots, p_{i,n}^-\}$ be the i-th training sample. It consists of one query, one positive passage, and n negative passages. Then we can employ a contrastive loss function, called InfoNCE (van den Oord et al., 2018), to optimize the model:

$$\mathcal{L} = -\log \frac{e^{f(q_i, p_i^+)}}{e^{f(q_i, p_i^+)} + \sum_{j=0}^{n} e^{f(q_i, p_{i,j}^-)}}. \tag{2}$$

In practice, we cannot use all passages in the corpus C as negative passages due to the limitation of resources. Therefore, a common practice is sampling a subset from the corpus C as negative samples, and many studies focus on which distribution the negative passages sampled from is better (Xiong et al., 2021; Qu et al., 2021).

Multilingual Setting. This setting aims to transfer knowledge from the source language to the target languages. In this setting, only labeled data from the source language is available. And the trained model will be directly evaluated on target languages. Note that the setting is different to *cross-lingual retrieval* whose queries and passages are in different languages. In this setting, the queries and the passages are in the same language and just training data from the source language (*e.g.,* English) is available.

4 Methodology

In this section, we present the proposed LeSTM. The overview is presented in Figure 1. We first present the augmentation method which combines sparse and dense retrievers. Then, we present how to use the mined data to train the query generator, generate new data, filter the generated samples, and fine-tune the dense retriever. Finally, we summarize the full training process.

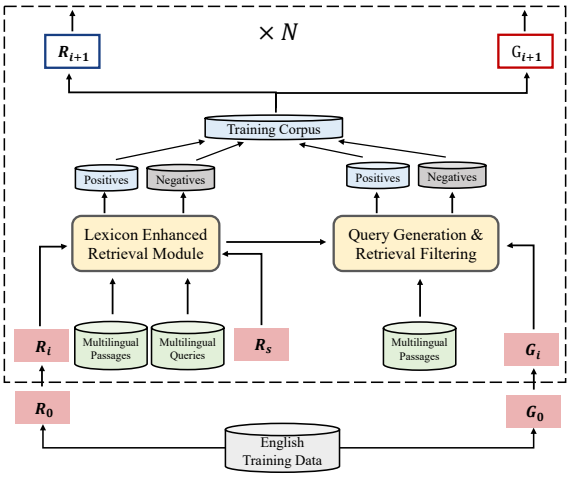


Figure 1: Overview of the training process.

Algorithm 1: Lexicon-enhanced Retrieval Module.

Input: One query q and candidate passages P.
Output: Positive passages and Negative passages.

Set L and S;
$\mathbb{L}_s$ & $\mathbb{S}_s \leftarrow$ Top-L and Top-S of sparse retriever;
$\mathbb{L}_d$ & $\mathbb{S}_d \leftarrow$ Top-L and Top-S of dense retriever;
$P^+ \leftarrow \varnothing$; $P^- \leftarrow \varnothing$;
for $p \in P$ **do**
  if $p \in \mathbb{S}_s$ & $p \in \mathbb{S}_d$ **then**
    $P^+ \leftarrow P^+ \cup \{p\}$
  end
  if $p \in \mathbb{S}_s$ & $p \notin \mathbb{L}_d$ **then**
    $P^- \leftarrow P^- \cup \{p\}$
  end
  if $p \notin \mathbb{L}_s$ & $p \in \mathbb{S}_d$ **then**
    $P^- \leftarrow P^- \cup \{p\}$
  end
end
Return P^+, P^-.

4.1 Lexicon-enhanced Retrieval Module

In target languages, we do not have the labeled training data, but we have unlabeled queries and passages. To effectively utilize the unlabeled queries and passages, we design a mining method shown in Algorithm 1.

This augmentation is based on the intuition that the sparse retriever and the dense retriever solve different problems and they are complementary. Specifically, the sparse retriever depends on word match, it is more effective than the dense retriever for words that do not appear in the training set. On the contrary, the dense retriever depends on neural networks, it is more effective than the sparse retriever for synonyms and semantics of the sentence. As a result, for a passage, if both of them regard it are relevant to the query, we then regard the passage as a positive passage. If one of them regards it as relevant but the other regards it as irrelevant, we regard the passage does not meet all conditions (*i.e.,* keyword match and semantic match), and the passage is a hard case. Although we cannot judge whether it is a negative passage, we think its relevance is weaker than the positive passage. As a result, we hope the score of the positive passage is higher than the hard case and then we regard the hard case as a hard negative passage.

In practice, because the score distributions of sparse retrievers and dense retrievers are different, we use the ranking position to measure the relevance between passages and queries. Then, we present our method as follows:

(1) We introduce two parameters to define relevant and irrelevant passages: S and L, *i.e.,* for a query, the retriever retrieves passages and ranks them with scores, if the ranking position of a passage is less than S, we regard the passage is relevant to the query and if the ranking position is greater than L, we regard the passage is irrelevant to the query.

(2) We retrieve L and S passages by both sparse and dense retrievers, respectively. We define the top-L passage set as $\mathbb{L}$ and the top-S passage set as $\mathbb{S}$, and use subscript s and d to denote sparse and dense retrievers, respectively.

(3) We traverse all passages in the corpus. For each passage, if it exists in both $\mathbb{S}_s$ and $\mathbb{S}_d$, we add it to the positive passage set; if it exists in one $\mathbb{S}$ but does not exist in another $\mathbb{L}$ (*a.k.a.,* exists in $\mathbb{S}_s$ but not exists in $\mathbb{L}_d$ or exists in $\mathbb{S}_d$ but does not exist in $\mathbb{L}_s$), we add it to the negative passage set.

(4) For each mined sample, we add random negative passages like DPR (Karpukhin et al., 2020): 1) random passages from the corpus; 2) positive passages of other queries (*a.k.a.,* in-batch negative). And we use our mined negative passage set to replace "top passages returned by BM25" in DPR.

To sum up, our mined training data includes a query, mined positive and negative passages, and random negative passages.

4.2 Query Generation Module

Due to the limited number of unlabeled queries, we leverage a query generator to generate more

Algorithm 2: The training algorithm.

Input: Labeled English training data and unlabeled queries and passages in target languages.

Construct index for sparse retriever;
Initialize the dense retriever and generator with pre-trained models;
Train the retriever and generator with English data;
Build ANN index for the retriever;
while *models has not converged* **do**
    Generating training data $\mathbb{D}_p$ with passage mining module;
    Generating training data $\mathbb{D}_g$ with query generation module;
    Fine-tune the generator and retriever with both $\mathbb{D}_p$ and $\mathbb{D}_g$;
    Refresh ANN index for the retriever.
end

queries for unlabeled passages in target languages. Note that the generated queries are in the same languages as the corresponding passage.

Specifically, for a trained generator, we randomly select some passages and leverage the fine-tuned query generator to generate queries for these passages. To tackle the noisy label problem introduced by the generator, we use both sparse and dense retrievers to filter the generated pairs. We retrieve the top-1 passage for each generated query with both sparse and dense retrievers and only accept pairs where the best passages from both sparse and dense retrievers are the corresponding passage. Finally, for each filtered sample, we select negative passages like DPR (Karpukhin et al., 2020): 1) random passages from the corpus; 2) top passages returned by sparse and dense retrievers (the passages returned by dense retriever are more effective (Xiong et al., 2021; Qu et al., 2021)); 3) positive passages of other queries.

To sum up, our generated training data includes one positive passage, generated query, random negative passages, and top passages returned by retrievers as hard negative passages.

4.3 Model Training

Previously, we introduced the lexicon-enhanced retrieval module and the query generation module. In this part, we present the full training process.

As shown in Algorithm 2, firstly, we train the warm-up dense retriever and query generator with data in the source language. We note that the input to the generator is the positive passage, and the label is the query. Secondly, we generate training data in target languages with the proposed two modules. Finally, we fine-tune the retriever and the generator with the generated data. Based on these steps, we can conduct iteratively generating and training procedures to improve the performance. Note that due to the grammatical adjustment and accidental translation problems in the zero-shot multilingual generator, we only use the lexicon-enhanced retrieval module to generate data in the first iteration.

Considering the query generator is more sensitive to the quality of data, we set $S = 1$ when generating data for the query generator.

5 Experiments

In this section, we construct experiments to demonstrate the effectiveness of the proposed method.

5.1 Experimental Setup

5.1.1 Dataset

Mr. TYDI. The Mr. TYDI dataset (Zhang et al., 2021b) is constructed from TYDI (Clark et al., 2020) dataset and can be viewed as the "open-retrieval" condition of the TYDI dataset. It is a multilingual dataset for monolingual retrieval in 11 languages. The detailed statistics of the Mr. TYDI dataset are presented in Appendix A.

DeepQA. An Q&A task dataset from one commercial Q&A system, with 18,000 labeled cases in three languages: English (En), German (De), French (Fr). Each case consists of two parts, *i.e.,* query and passage. The detailed statistics of the DeepQA dataset are presented in Appendix A.

5.1.2 Evaluation Metrics.

Following Mr. TYDI, we use MRR@100 and Recall@100 as evaluation metrics, where MRR denotes the mean of reciprocal rank across queries and Recall@k denotes the proportion of queries to which the top k retrieved passages contain positives. For DeepQA, due to the smaller size of the corpus (only 1,220,030 passages in the corpus, for comparison, the Mr. TYDI data has 58,043,326 passages, which is times that of DeepQA), we use MRR@10 and Recall@10 as metrics.

Table 1: Results on Mr. TYDI test set. The best results except supervised training are in bold. We copy the results of BM25, tuned BM25, and zero-shot mBERT from (Zhang et al., 2022b) and re-implement the zero-shot mBERT. $*$ denotes that our method significantly outperforms self-training at the level of 0.01. † denotes that our method significantly outperforms back-training at the level of 0.01.

Methods		MRR@100	Recall@100
Sparse Method	BM25	32.1	73.2
	(tuned)	33.3	75.8
mBERT	Zero-Shot	34.4	73.4
	(reimpl)	36.5	73.3
	Self-Training	37.2	78.5
	Back-Training	41.1	82.0
	LeSTM	**49.0***†	**83.6***†
	Supervised	54.6	87.0
XLM-R	Zero-Shot	30.4	74.3
	Self-Training	35.0	78.6
	Back-Training	29.6	77.5
	LeSTM	**47.2***†	**82.7***†
	Supervised	54.5	87.2

5.1.3 Implementation Details.

For the warm-up training stage, although Mr. TYDI proposed to use NQ (Kwiatkowski et al., 2019) as English training data, we follow Xinyu *et al.* (Zhang et al., 2022b) to use MS-MARCO as English training data. Xinyu *et al.* find that MS-MARCO is better than NQ for zero-shot multilingual retrieval. We have further constructed experiments on NQ in Appendix D.3.

For the iteratively training stage, both the retriever and the generator are scheduled to train with 500 mini-batches in each iteration. The document index is refreshed after each iteration of training. The hyper-parameters are shown in Appendix B.

All the experiments run on 8 NVIDIA Tesla A100 GPUs. The implementation code is based on HuggingFace Transformers (Wolf et al., 2020). For sentence embedding, we use the corresponding hidden state of the *[CLS] token* for mBERT (Devlin et al., 2019) and the average hidden states of all tokens for XLM-R (Conneau et al., 2020). For the generator, we leverage mBART (Liu et al., 2020) as the pre-trained model.

5.2 Results

5.2.1 Baselines

As we investigate retrieval in the multilingual setting, in this paper, the main baselines methods include BM25, and multilingual DPR with mBERT (Devlin et al., 2019), XLM-R (Conneau et al., 2020) as the multilingual pre-trained model. Furthermore, we compare our method with two state-of-the-art domain adaption methods: self-training (Yarowsky, 1995) and back-training (Kulshreshtha et al., 2021). Following Back-training, we train the models 3 iterations with 5 epochs per iteration. Then we present the results with the best MRR@100. In addition, we present the supervised performance as an upper limit reference. When constructing the supervised training data, we follow DPR (Karpukhin et al., 2020) to select three kinds of negative passages.

5.2.2 Mr. TYDI

Table 1 shows the result on Mr. TYDI. The first group is the sparse retriever, *i.e.,* BM25 (Robertson and Zaragoza, 2009) and tuned BM25. For each pre-trained model, the first group is the multilingual pre-trained models which are only fine-tuned on MS-MARCO data. The second block is the multilingual pre-trained models which are fine-tuned on MS-MARCO data and data augmentation method. We conduct pair t-test (Hsu and Lachenbruch, 2014) between our method and other data augmentation method (self-training and back-training). The final block is the multilingual pre-trained models which are fine-tuned on Mr. TYDI dataset. Due to the limited space, we only present the average performance among all languages in Table 1 and present results for each language in Appendix E.

Based on the results, we have the following findings. Firstly, comparing the performance of domain adaption methods (the second block for each pre-trained model) and zero-shot performance, we can find that all domain adaption methods are effective. Secondly, comparing the three domain adaption methods, we can find that our method is better than the other methods. Finally, comparing our method and supervised dense retriever, we can find that the performance of our method is closed to the supervised performance on Recall@100, but is still worse than supervised performance with a clear edge on MRR@100. This indicates that the augmented data are noisy, for example, the mined passages are relevant to the queries but are not the best passages, and there may be more relevant passages for the queries. So, it is more helpful to Recall@100 but less helpful to MRR@100.

Table 2: Results on DeepQA test set. The best results except supervised training are in bold. $*$ denotes that our method significantly outperforms self-training at the level of 0.01. † denotes that our method significantly outperforms back-training at the level of 0.01.

(a) MRR@10

Methods	En	De	Fr	Avg
BM25	22.5	31.4	40.1	31.3
Zero-Shot	24.0	29.4	37.7	30.3
Self-Training	25.3	31.4	42.3	33.0
Back-Training	25.8	32.0	42.0	33.3
LeSTM	**27.2***†	**34.6***†	**43.0***†	**35.0***†
Supervised	23.0	33.9	39.7	32.2

(b) Recall@10

Methods	En	De	Fr	Avg
BM25	37.2	52.1	56.8	48.7
Zero-Shot	39.1	49.3	56.7	48.4
Self-Training	41.8	55.9	60.9	52.7
Back-Training	42.6	55.8	61.4	53.2
LeSTM	**44.2***†	**57.9***†	**62.9***†	**55.0***†
Supervised	38.8	62.8	63.0	54.7

Table 3: Performance of zero-shot dense retriever on DeepQA training set and test set.

	MRR@10	Recall@10
Training set	35.1	48.5
Test set	30.3	48.4

5.2.3 DeepQA

Due to the limited space, we only construct experiments on DeepQA based on mBERT. Table 2 presents the performance of all methods. As we can see, our method achieves the best performance among all the compared methods. It indicates that our method is effective for unsupervised multilingual dense retrieval.

In addition, we find that the unsupervised methods (*i.e.,* self-training and back-training) perform better than the supervised training on MRR@10 but worse on Recall@10. A possible reason is that the size of DeepQA is small and there is a large gap between the distributions of the training data and test data. To demonstrate that, we evaluate the performance of the zero-shot dense retriever on both training data and test data. As shown in Table 3, there is a large gap between the MRR@10 on the training set and the test set of DeepQA. That indicates the gap between the training set of the test set is large. The dense model trained on the training set may seriously suffer from the overfitting problem. These results also indicate that our method is even more effective than supervised training when the training data in target languages is limited.

Table 4: Ablation results based on mBERT. "LR" denotes the lexicon-enhanced retrieval module. "QG" denotes the query generation module.

Methods	MRR@100	Recall@100
LeSTM	**49.0**	**83.6**
w/o LR	46.9	81.6
w/o LR$_+$	37.9	77.9
w/o QG	48.1	83.2
w/o ALL	36.5	73.3

5.3 Ablation Study

In our method, we have incorporated two data augmentation modules, namely lexicon-enhanced retrieval, and query generation. Here, we would like to check how each module contributes to the final performance. We construct the ablation experiments on the Mr. TYDI data. We prepare four variants of our method that try all combinations:

- w/o LR denotes that the retriever does not be fine-tuned with data from the lexicon-enhanced retrieval module. But the generator also is fine-tuned with data from the lexicon-enhanced retrieval module.
- w/o LR$_+$ denotes that both the retriever and the generator do not be fine-tuned with data from the lexicon-enhanced retrieval module.
- w/o QG denotes that the retriever does not be fine-tuned with data from the query generation module.
- w/o ALL denotes without both the two modules, *a.k.a.,* zero-shot multilingual retrieval.

Table 4 presents all comparison results of the four variants. Due to the limited space, we present results for each language in Appendix E. As we can see, the performance rank can be given as follows w/o ALL < w/o QG < LeSTM. These results indicate that both the two augmentation modules are essential to improve performance. And we can find that the lexicon-enhanced retrieval module is more effective than the query generation module, because of w/o LR < w/o QG. In addition, we find that w/o LR > w/o LR$_+$, it denotes the zero-shot multilingual query generation suffers from lots of

Table 5: Effect of lexicon-enhanced retrieval module.

Methods	MRR@100	Recall@100
BM25	32.1	73.2
Zero-shot mBERT	36.5	73.3
LeSTM w/o QG	**47.5**	**82.5**
Sparse + Dense	46.6	81.1
Sparse × Dense	40.8	80.4
Double Dense Retrievers	44.5	81.9
w/o HN	42.5	80.4
w/ Sparse HN	43.3	79.2

problems and it also can demonstrate the effectiveness of the lexicon-enhanced retrieval module.

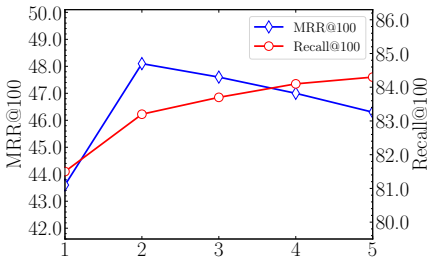
5.4 Method Analysis

5.4.1 Effect of Lexicon-enhanced Retrieval

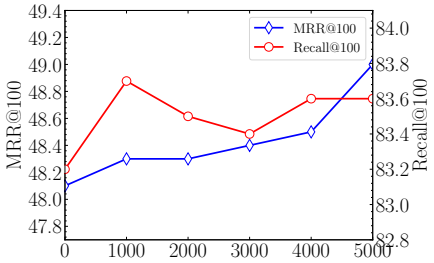
In our lexicon-enhanced retrieval module, we combine the results of the sparse and dense retrievers to mine new training data. To show the effectiveness of our mining method, we construct the five variants (for more conciseness, we use mBERT w/o QG + iterative refinement as the base model):

- Sparse + Dense combines results of sparse and dense retrievers by adding their scores.
- Sparse × Dense combines results of sparse and dense retrievers by multiplying their scores.
- Double Dense Retrievers mines positive and negative passages with results from two dense retrievers which are trained on different data (MS-MARCO and NQ).
- w/o Hard Negatives (HN) fine-tunes the model with mined positive passages and only in-batch negative passages.
- w/ Sparse Hard Negatives (HN) fine-tunes the model with mined positive passages, in-batch negative passages, and top passages returned by sparse retriever as negative passages.

Table 5 presents all comparison results of the five variants. Based on the results, we have the following findings. Firstly, our mining method is more effective than the hybrid results of sparse and dense models. It demonstrates that our method can effectively leverage the knowledge of both sparse and dense retrievers. Secondly, mining data with sparse and dense retrievers are more effective than two dense retrievers. It demonstrates that sparse and dense retrievers have noticed different characteristics of retrieval. Finally, mined negatives are more effective than sparse negatives. It demonstrates that negatives are important in dense retrieval tasks and our methods can provide more effective negatives.



(a) S in Algorithm 1.



(b) The number of generated queries.

Figure 2: Parameter sensitivity.

5.4.2 Effect of Parameters

In our method, we introduce two parameters in the lexicon-enhanced retrieval module to define relevant and irrelevant passages: S and L. And the number of generated queries also influences the final performance. Here, we tune the S and L based on mBERT w/o QG. We vary S in the set $\{1, 2, 3, 4, 5\}$. And for more conciseness, we set $L = S \times 10$. In addition, we tune the number of generated queries based on mBERT. We vary the number of generated queries per language in the set $\{1000, 2000, 3000, 4000, 5000\}$.

Figure 2(a) presents the effect of the parameter S. We can observe that $S = 1$ leads to poor performance on both MRR@100 and Recall@100. Because the method with little S mines few samples and leads to the overfitting problem. When we set $S > 2$, it leads to better Recall@100 but poorer MRR@100. A possible reason is that large S leads to more noisy samples. As we mentioned above, noisy samples are helpful to Recall@100 but harmful to MRR@100.

Figure 2(b) presents the effect of the number of generated queries. As we can see, the large number of generated queries improves the MRR@100 but cannot improve the Recall@100. A possible reason

is that the generated queries mainly focus on a few kinds (*e.g.,* when or where something happened). They are helpful to MRR@100 for these kinds of queries but less helpful to both Recall@100 and MRR@100 for other kinds of queries.

6 Conclusion

In this paper, we propose a novel augmentation method that combines sparse and dense retrievers for multilingual retrieval. We firstly designed a passage mining method based on the results of both sparse and dense retrievers. After that, we utilized the mined data to train a query generation model and generate more training data. Extensive experimental results show that the proposed method outperforms the baselines, and can significantly improve the state-of-the-art performance. Currently, we directly utilize a large number of unlabeled queries in target languages. As future work, we will investigate how to augment training data without any unlabeled queries in target languages.

7 Limitations

The limitations are summarized as follows.

- The method needs unlabeled queries. For seriously rare languages, there are no unlabeled queries in search engines and we cannot perform our passage mining method in this condition. Although our query generation module can alleviate this problem, the zero-shot query generator suffers from grammatical adjustment and accidental translation problems and can only provide limited help.
- The method performs inconsistently on the two metrics (MRR@100 and Recall@100). Due to the quality of augmented data, we need to set some threshold to filter the augmented data, where different parameters lead to optimal performance on different metrics.
- The sparse retriever is fixed during training. The fixed sparse retriever leads to the rapid convergence of the dense retriever. We believe that if both sparse and dense retrievers can be improved in the iterative process, the dense retriever may achieve better performance.

Acknowledgments

Jian Pei's research is supported in part by the NSERC Discovery Grant program. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Akari Asai, Jungo Kasai, Jonathan H. Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. 2021a. XOR QA: cross-lingual open-retrieval question answering. In *NAACL-HLT*, pages 547–564. Association for Computational Linguistics.

Akari Asai, Xinyan Yu, Jungo Kasai, and Hanna Hajishirzi. 2021b. One question answering model for many languages with cross-lingual dense passage retrieval. In *NeurIPS*, pages 7547–7560.

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.

Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2021. Infoxlm: An information-theoretic framework for cross-lingual language model pre-training. In *NAACL-HLT*, pages 3576–3588. Association for Computational Linguistics.

Jonathan H. Clark, Jennimaria Palomaki, Vitaly Nikolaev, Eunsol Choi, Dan Garrette, Michael Collins, and Tom Kwiatkowski. 2020. Tydi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Trans. Assoc. Comput. Linguistics*, 8:454–470.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *ACL*, pages 8440–8451. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *NeurIPS*, pages 7057–7067.

Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *SIGIR*, pages 1533–1536. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *ACL (1)*, pages 878–891. Association for Computational Linguistics.

Luyu Gao and Jamie Callan. 2021a. Condenser: a pre-training architecture for dense retrieval. In *EMNLP (1)*, pages 981–993. Association for Computational Linguistics.

Luyu Gao and Jamie Callan. 2021b. Unsupervised corpus aware language model pre-training for dense passage retrieval. *CoRR*, abs/2108.05540.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: revisit exact lexical match in information retrieval with contextualized inverted list. In *NAACL-HLT*, pages 3030–3042. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Henry Hsu and Peter A Lachenbruch. 2014. Paired t test. *Wiley StatsRef: statistics reference online*.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *ICLR*. OpenReview.net.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Towards unsupervised dense information retrieval with contrastive learning. *CoRR*, abs/2112.09118.

Gautier Izacard and Edouard Grave. 2021. Distilling knowledge from reader to retriever for question answering. In *ICLR*. OpenReview.net.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547.

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

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *SIGIR*, pages 39–48. ACM.

Devang Kulshreshtha, Robert Belfer, Iulian Vlad Serban, and Siva Reddy. 2021. Back-training excels self-training at unsupervised domain adaptation of question generation and passage retrieval. In *EMNLP (1)*, pages 7064–7078. Association for Computational Linguistics.

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*, 7:452–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *ACL (1)*, pages 6086–6096. Association for Computational Linguistics.

Yulong Li, Martin Franz, Md. Arafat Sultan, Bhavani Iyer, Young-Suk Lee, and Avirup Sil. 2021. Learning cross-lingual IR from an english retriever. *CoRR*, abs/2112.08185.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Trans. Assoc. Comput. Linguistics*, 8:726–742.

Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tie-Yan Liu, and Arnold Overwijk. 2021. Less is more: Pretrain a strong siamese encoder for dense text retrieval using a weak decoder. In *EMNLP (1)*, pages 2780–2791. Association for Computational Linguistics.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Trans. Assoc. Comput. Linguistics*, 9:329–345.

Ji Ma, Ivan Korotkov, Yinfei Yang, Keith B. Hall, and Ryan T. McDonald. 2021a. Zero-shot neural passage retrieval via domain-targeted synthetic question generation. In *EACL*, pages 1075–1088. Association for Computational Linguistics.

Xueguang Ma, Kai Sun, Ronak Pradeep, and Jimmy Lin. 2021b. A replication study of dense passage retriever. *CoRR*, abs/2104.05740.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and code embeddings by contrastive pre-training. *CoRR*, abs/2201.10005.

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 *CoCo@NIPS*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2021. Large dual encoders are generalizable retrievers. *CoRR*, abs/2112.07899.

Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to doctttttquery. *Online preprint*.

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*, pages 5835–5847. Association for Computational Linguistics.

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 (1)*, pages 2825–2835. Association for Computational Linguistics.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Hongyin Tang, Xingwu Sun, Beihong Jin, Jingang Wang, Fuzheng Zhang, and Wei Wu. 2021. Improving document representations by generating pseudo query embeddings for dense retrieval. In *ACL/IJCNLP (1)*, pages 5054–5064. Association for Computational Linguistics.

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*, abs/2104.08663.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

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 *NIPS*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP (Demos)*, pages 38–45. 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 *ICLR*. OpenReview.net.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2022. Laprador: Unsupervised pretrained dense retriever for zero-shot text retrieval. *CoRR*, abs/2203.06169.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *NAACL-HLT*, pages 483–498. Association for Computational Linguistics.

Sohee Yang and Minjoon Seo. 2020. Is retriever merely an approximator of reader? *CoRR*, abs/2010.10999.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, pages 189–196. Morgan Kaufmann Publishers / ACL.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *SIGIR*, pages 1503–1512. ACM.

Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2021a. Adversarial retriever-ranker for dense text retrieval. *CoRR*, abs/2110.03611.

Shunyu Zhang, Yaobo Liang, Ming Gong, Daxin Jiang, and Nan Duan. 2022a. Multi-view document representation learning for open-domain dense retrieval. In *ACL (1)*, pages 5990–6000. Association for Computational Linguistics.

Xinyu Zhang, Xueguang Ma, Peng Shi, and Jimmy Lin. 2021b. Mr. tydi: A multi-lingual benchmark for dense retrieval. *CoRR*, abs/2108.08787.

Xinyu Zhang, Kelechi Ogueji, Xueguang Ma, and Jimmy Lin. 2022b. Towards best practices for training multilingual dense retrieval models. *CoRR*, abs/2204.02363.

Appendix

A Dataset Statistics

Mr.TYDI. Mr. TYDI is a multilingual retrieval benchmark based on the TYDI dataset. Mr. TYDI covers 11 languages. The corpus for each language is drawn from Wikipedia, and the query and judgments are prepared by native speakers of that language. Table 12 presents statistics of the Mr. TYDI dataset, copied from the original paper.

DeepQA. DeepQA is a Q&A dataset from one commercial Q&A system, with 18,000 labeled cases in three languages. Each case consists of two parts, *i.e.,* query and passage. The following briefly describes how the data is collected. Firstly, for each query, the top 10 relevant documents returned by the search engine are selected to form <query, url> pairs; Then passages are further extracted from these documents to form <query, url, passage> triples; These <query, passage> pairs are sampled and sent to crowd sourcing judges. Specifically, each <query, passage> pair is required to get judged by three judges. Those cases with more than 2/3 positive labels will get positive labels, otherwise negative. The detailed statistics of the DeepQA dataset are presented in Table 6.

Table 6: Statistics for DeepQA: number of queries (# Q), judgments (# J), and the number of passages.

L	Train		Test		Corpus Size
	# Q	# J	# Q	# J	
En	4,437	6,022	1,703	1,978	741,840
De	2,401	3,977	910	1,023	385,440
Fr	2,976	4,000	936	1,000	92,750
All	9,814	13,999	3,549	4,001	1,220,030

B Hyper-parameters

We have analyzed the parameters of our method in Section 5.4.2. Here, we present the other hyper-parameters of our method in Table 7, most of them follow Back-training (Kulshreshtha et al., 2021) and DPR (Karpukhin et al., 2020).

C Efficiency Report

We list the time cost of training and inference in Table 8. The evaluation is made with 8 NVIDIA A100 GPUs. The number of iterations is set as 3.

Table 7: Hyper-parameters.

	Parameters	Value
	Max Query Length	32
	Max Passage Length	128
Train warm-up retriever	Learning Rate	1e-5
	Batch Size	128
	Negative Size	255
	Optimizer	AdamW
	Scheduler	Linear
	Warmup Proportion	0.1
	Training Epoch	3
Train warm-up generator	Learning Rate	1e-5
	Batch Size	64
	Optimizer	AdamW
	Scheduler	Linear
	Warmup Proportion	0.1
	Training Epoch	1
Iteraively training of retriever	Learning Rate	1e-6
	Batch Size	128
	Negative Size	255
	Optimizer	AdamW
	Scheduler	Linear
	Warmup Proportion	0.1
	Training Epoch	5
	S in Algorithm 1	2
	L in Algorithm 1	20
	# of Generated Queries	5000
	# of Iteration	3
Iteraively training of generator	Learning Rate	1e-5
	Batch Size	64
	Optimizer	AdamW
	Scheduler	Linear
	Warmup Proportion	0.1
	Training Epoch	5
	# of Iteration	3

Table 8: Efficiency Report.

Training	Warm-up	0.5h
	Per Iteration	0.2h
	Index Refresh	1.7h
	Generate Queries	0.5h
	Overall	8.9h
Inference	Build Index	1.7h
	Query Encoding	40ns
	Dense Retrieval	2ms

D Additional Experiments

D.1 Effect of Query Filter

In our query generation module, we use the dense retriever to filter the generated queries. Here, we analyze the effectiveness of filtering based on mBERT. We present the result for w/o Queries Filter and w/o Queries Filter & Hard Negative passages in Table 10. As we can see, filtering generated queries can lead to better performance. We also find that w/o QF & HN > LeSTM > w/o QF on Re-

Table 9: A filtered out generated query.

Passage: Baada ya uhuru wa Fiji (1970) kutoka kwa Waingereza, yalifuata mapinduzi ya kijeshi yaliyotokea mwaka 1987, hali iliyosababishwa na wakazi wa Fiji kulaumu serikali yao kutawaliwa na watu wa kabila la Indofijian au Wahindi. **Translation:** After Fiji's independence (1970) from the British, it followed a military coup in 1987, a situation in which Fijians blamed their government for being ruled by Indofijian or Indian people.
Generated query: Kwa upi uhuru wa Fiji ulifanyika mwaka gani? **Translation:** In what year did Fiji's independence take place?
Top-1 passage: Fiji ilijipatia uhuru wake kutoka katika utawala wa kikoloni wa Uingereza tarehe 10 Oktoba 1970. **Translation:** Fiji gained its independence from British colonial rule on October 10, 1970.

call@100. It denotes that the generated queries are relevant to the passages but the top passages returned by retrievers may be more relevant. We present an example of a query that is filtered out in Table 9. As we can see, although the generated query can be answered by the passage, the main statement of the passage is not intended to answer the generated query and there is a more relevant passage to answer the generated query. As a result, these samples (w/o hard negative passages) are helpful to Recall@100 but harmful to MRR@100.

D.2 Visualization of the Training Procedure

Our method employs iteratively training to improve the performance. Here, we report the iterative performance of our method in Figure 3. To better show the effectiveness of our method, we set the number of iterations as 5 in the experiments. As we can see, the performance of our method increases with iteration and it holds steady when the model convergences. It shows that the distribution of mined data is similar to the distribution of real data, so the model does not suffer from the overfitting problem. In the end, the MRR@100 is improved by approximately 12%, and Recall@100 is improved by approximately 10%. It demonstrates the effectiveness of the mined data.

D.3 Effect of English Data

In this section, we test the influence of different English data. As Xinyu *et al.* (Zhang et al., 2022b) said, MS-MARCO (Nguyen et al., 2016) has a larger dataset size than NQ (Kwiatkowski et al., 2019), and the data size is a more critical factor. To

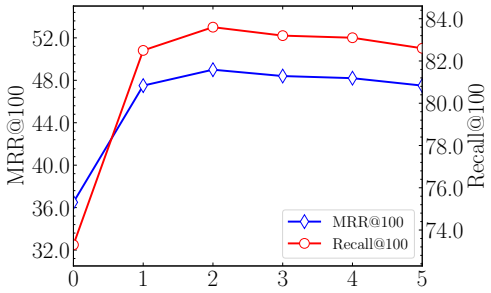


Figure 3: Iterative performance for the proposed LeSTM based on mBERT.

test the effectiveness of our methods, we construct experiments on both MS-MARCO and NQ. Note that the only difference is the English data and we directly use the tuned parameters based on MS-MARCO for all experiments. Table 11 presents the performance of two English data based on mBERT. Note that our re-implement mBERT based on NQ is better than the Mr. TYDI paper (Zhang et al., 2021b)[2]. Because we share the parameters of the query encoder and the passage encoder, the trick leads to better performance.

As we can see, the gap between the performance of our method based on MS-MARCO data and NQ data is smaller than other methods. And the Recall@100 on NQ data is even higher than Recall@100 on MS-MACRO data. A possible reason is that NQ is closer to Mr. TYDI, both of them contain relatively well-formed queries posed against Wikipedia. The zero-shot performance of English data in Mr. TYDI data (the Recall@100 of English is 75.1 for the model trained on MS-MARCO data and 78.3 for the model trained on NQ data) can demonstrate that. So, the mined data of dense retriever trained by NQ is more effective than MS-MARCO, and more effective data in target languages can lead to better performance.

E Detailed Results

Due to the limited space, we only present average performance in the experiment section. Here, we present the performance of each language. First, we present the detailed performance of both our method and baselines in Table 13. Second, we present the detailed performance of ablation results in Table 14. Finally, we present the detailed performance of variants of the passage mining module in Table 15. As we can see, our method performs better in most languages in these settings.

[2]https://github.com/castorini/mr.tydi

Table 10: Effect of query filter. "QF" denotes query filter and "HN" denotes hard negative passages which are top passages returned by sparse and dense retrievers.

(a) MRR@100

Methods	Ar	Bn	En	Fi	Id	Ja	Ko	Ru	Sw	Te	Th	Avg
LeSTM	**58.5**	**49.5**	**37.3**	**45.6**	**51.3**	**38.6**	**43.6**	**46.2**	**48.6**	**66.5**	**53.1**	**49.0**
w/o QF	53.5	47.1	33.1	39.9	47.5	36.6	36.2	44.0	44.2	27.6	42.0	41.1
w/o QF & HN	58.1	48.6	37.0	45.2	50.6	38.4	**43.6**	46.0	47.7	54.6	53.0	47.5

(b) Recall@100

Methods	Ar	Bn	En	Fi	Id	Ja	Ko	Ru	Sw	Te	Th	Avg
LeSTM	85.9	89.2	77.6	83.1	86.4	77.0	74.9	81.8	83.4	95.6	85.3	83.6
w/o QF	83.7	91.4	74.5	79.3	85.4	76.7	69.5	82.1	81.6	91.9	81.9	81.6
w/o QF & HN	**86.2**	**90.1**	**79.1**	**83.4**	**87.3**	**77.5**	**76.8**	**83.3**	**82.9**	**96.1**	**86.7**	**84.5**

Table 11: Performance comparison for different English data based on mBERT.

(a) MRR@100

Methods	Ar	Bn	En	Fi	Id	Ja	Ko	Ru	Sw	Te	Th	Avg
MS-MARCO												
Zero-Shot	47.7	41.5	32.4	34.9	41.9	30.9	30.8	35.6	40.4	34.8	30.2	36.5
Self-Training	45.4	40.5	28.5	33.8	40.2	32.1	32.1	34.1	43.8	42.9	36.2	37.2
Back-Training	49.0	48.7	31.7	38.9	44.5	34.5	34.8	38.3	46.0	46.7	38.9	41.1
LeSTM	**58.5**	49.5	37.3	**45.6**	**51.3**	38.6	**43.6**	**46.2**	**48.6**	66.5	**53.1**	**49.0**
NQ												
Zero-Shot	30.0	38.7	30.6	25.6	30.7	30.6	23.4	29.6	28.1	24.1	22.6	28.5
Self-Training	32.2	41.5	27.9	27.3	33.5	30.4	24.2	30.4	33.3	36.5	28.0	31.4
Back-Training	35.4	42.3	29.7	28.5	33.2	31.5	27.7	32.8	40.8	24.7	30.5	32.5
LeSTM	54.8	**56.6**	**41.0**	42.3	51.2	**41.6**	38.9	46.1	46.2	**66.7**	49.1	48.6

(b) Recall@100

Methods	Ar	Bn	En	Fi	Id	Ja	Ko	Ru	Sw	Te	Th	Avg
MS-MARCO												
Zero-Shot	80.6	78.8	75.1	74.7	79.3	67.8	65.5	73.1	70.4	77.1	63.7	73.3
Self-Training	82.3	86.9	74.4	78.2	81.3	72.7	67.0	77.6	78.7	91.0	73.6	78.5
Back-Training	84.6	90.1	76.5	81.4	84.4	76.2	73.6	82.2	80.8	89.6	83.1	82.0
LeSTM	**85.9**	89.2	77.6	83.1	86.4	77.0	74.9	81.8	83.4	95.6	85.3	83.5
NQ												
Zero-Shot	70.3	80.2	78.3	69.2	76.4	74.3	60.9	72.7	65.7	69.9	59.9	70.7
Self-Training	78.9	86.5	79.1	77.7	83.3	79.0	69.6	77.9	76.7	90.5	74.8	79.5
Back-Training	81.7	88.7	80.1	80.0	86.0	80.8	73.5	81.9	83.4	93.6	83.0	83.0
LeSTM	**85.9**	**91.9**	**83.9**	**84.3**	**88.7**	**83.2**	**78.4**	**85.5**	**82.9**	**95.7**	**85.7**	**86.0**

Table 12: Statistics for Mr. TYDI: number of queries (# Q), judgments (# J), and the number of passages.

L	Train		Dev		Test		Corpus Size
	# Q	# J	# Q	# J	# Q	# J	
Ar	12,377	12,377	3,115	3,115	1,081	1,257	2,106,586
Bn	1,713	1,719	440	443	111	130	304,059
En	3,547	3,547	878	878	744	935	32,907,100
Fi	6,561	6,561	1,738	1,738	1,254	1,451	1,908,757
Id	4,902	4,902	1,224	1,224	829	961	1,469,399
Ja	3,697	3,697	928	928	720	923	7,0000,027
Ko	1,295	1,317	303	307	421	492	1,496,126
Ru	5,366	5,366	1,375	1,375	955	1,168	9,597,504
Sw	2,072	2,401	526	623	670	743	136,689
Te	3,880	3,880	983	983	646	664	548,224
Th	3,319	3,360	807	817	1,190	1,368	568,855
All	48,729	49,127	12,317	12,431	8,661	10,092	58,043,326

Table 13: Detail results on Mr. TYDI test set.

(a) MRR@100

Methods		Ar	Bn	En	Fi	Id	Ja	Ko	Ru	Sw	Te	Th	Avg
Sparse Retriever	BM25	36.8	41.8	14.0	28.4	37.6	21.1	28.5	31.3	38.9	34.3	40.1	32.1
	(tuned)	36.7	41.3	15.1	28.8	38.2	21.7	28.1	32.9	39.6	42.4	41.7	33.3
mBERT	Zero-Shot	44.4	38.3	31.5	30.6	37.8	31.4	29.7	33.7	36.9	36.3	28.2	34.4
	(reimpl)	47.7	41.5	32.4	34.9	41.9	30.9	30.8	35.6	40.4	34.8	30.2	36.5
	Self-Training	45.4	40.5	28.5	33.8	40.2	32.1	32.1	34.1	43.8	42.9	36.2	37.2
	Back-Training	49.0	48.7	31.7	38.9	44.5	34.5	34.8	38.3	46.0	46.7	38.9	41.1
	LeSTM	**58.5**	**49.5**	**37.3**	**45.6**	**51.3**	**38.6**	**43.6**	**46.2**	**48.6**	**66.5**	**53.1**	**49.0**
	Supervised	64.2	55.2	45.4	52.7	53.8	43.1	42.8	46.4	58.8	83.3	54.5	54.6
XLM-R	Zero-Shot	31.7	40.9	18.1	30.3	31.7	22.9	31.7	27.0	24.4	36.0	39.5	30.4
	Self-Training	37.8	41.6	21.6	31.6	36.0	25.6	29.4	28.1	40.5	54.6	38.4	35.0
	Back-Training	32.0	43.0	20.2	28.7	31.3	25.1	30.4	29.0	20.2	27.5	38.8	29.6
	LeSTM	**54.0**	**51.7**	**33.0**	**43.9**	**49.9**	**34.8**	**40.3**	**41.4**	**45.1**	**70.1**	**54.9**	**47.2**
	Supervised	62.9	61.5	43.0	51.6	53.6	39.9	41.2	44.0	57.6	83.3	60.9	54.5

(b) Recall@100

Methods		Ar	Bn	En	Fi	Id	Ja	Ko	Ru	Sw	Te	Th	Avg
Sparse Retriever	BM25	79.3	86.9	53.7	71.9	84.3	64.5	61.9	64.8	76.4	75.8	85.3	73.2
	(tuned)	80.0	87.4	55.4	72.5	84.6	65.6	79.7	66.0	76.4	81.3	85.3	75.8
mBERT	Zero-Shot	79.9	82.0	75.8	69.3	75.8	73.8	64.5	72.8	68.6	79.7	64.8	73.4
	(reimpl)	80.6	78.8	75.1	74.7	79.3	67.8	65.5	73.1	70.4	77.1	63.7	73.3
	Self-Training	82.3	86.9	74.4	78.2	81.3	72.7	67.0	77.6	78.7	91.0	73.6	78.5
	Back-Training	84.6	**90.1**	76.5	81.4	84.4	76.2	73.6	**82.2**	80.8	89.6	83.1	82.0
	LeSTM	**85.9**	89.2	**77.6**	**83.1**	**86.4**	**77.0**	**74.9**	81.8	**83.4**	**95.6**	**85.3**	**83.6**
	Supervised	90.2	92.3	84.2	85.8	87.7	82.1	78.8	84.7	85.9	96.2	88.7	87.0
XLM-R	Zero-Shot	76.3	84.2	68.8	74.6	82.3	66.7	67.9	68.9	60.3	81.2	86.5	74.3
	Self-Training	78.9	85.1	68.8	78.6	84.0	70.2	66.8	71.6	78.2	93.6	88.4	78.6
	Back-Training	78.9	**88.3**	72.4	77.3	84.5	73.1	71.1	75.2	59.7	80.6	91.1	77.5
	LeSTM	**85.0**	86.0	**76.2**	**82.6**	**86.4**	**74.6**	**73.6**	**77.8**	**80.2**	**95.4**	**92.1**	**82.7**
	Supervised	89.3	92.8	83.1	86.3	89.8	80.1	78.7	82.6	87.0	96.7	92.6	87.2

Table 14: Ablation results based on mBERT.

(a) MRR@100

Methods	Ar	Bn	En	Fi	Id	Ja	Ko	Ru	Sw	Te	Th	Avg
All	58.5	49.5	**37.3**	**45.6**	**51.3**	**38.6**	**43.6**	**46.2**	**48.6**	66.5	**53.1**	**49.0**
w/o LR	55.7	**49.6**	35.9	43.0	49.7	38.1	40.1	43.5	46.9	**66.9**	45.9	46.9
w/o LR$_+$	49.1	45.5	32.4	36.4	43.4	32.0	32.4	39.0	41.9	45.9	39.0	39.7
w/o QG	**58.8**	49.4	37.0	45.1	51.0	38.0	42.7	45.5	47.2	62.7	51.4	48.1
w/o LR + QG	47.7	41.5	32.4	34.9	41.9	30.9	30.8	35.6	40.4	34.8	30.2	36.5

(b) Recall@100

Methods	Ar	Bn	En	Fi	Id	Ja	Ko	Ru	Sw	Te	Th	Avg
All	**85.9**	**89.2**	77.6	83.1	86.4	**77.0**	**74.9**	81.8	**83.4**	**95.6**	**85.3**	**83.6**
w/o LR	84.0	87.4	76.0	80.4	85.1	76.2	70.7	80.7	81.7	93.8	81.6	81.6
w/o LR$_+$	80.8	85.1	73.7	76.8	82.5	71.1	65.7	77.7	77.5	89.5	76.6	77.9
w/o QG	85.7	88.3	**78.6**	**83.3**	**86.9**	76.2	74.4	81.7	80.8	95.4	84.3	83.2
w/o LR + QG	80.6	78.8	75.1	74.7	79.3	67.8	65.5	73.1	70.4	77.1	63.7	73.3

Table 15: Effect of lexicon-enhanced retrieval module.

(a) MRR@100

Methods	Ar	Bn	En	Fi	Id	Ja	Ko	Ru	Sw	Te	Th	Avg
Sparse	36.8	41.8	14.0	28.4	37.6	21.1	28.5	31.3	38.9	34.3	40.1	32.1
Dense	47.7	41.5	32.4	34.9	41.9	30.9	30.8	35.6	40.4	34.8	30.2	36.5
LeSTM w/o QG	**58.5**	50.8	**37.0**	**45.0**	**52.1**	**38.2**	**41.7**	**45.0**	46.6	**59.9**	48.2	**47.5**
Sparse + Dense	57.1	**54.1**	34.3	43.6	50.2	37.4	38.6	44.3	**47.7**	55.7	**49.9**	46.6
Sparse $\times$ Dense	48.5	48.3	25.7	37.1	44.0	31.2	34.6	39.0	42.4	49.8	47.8	40.8
Double Dense Retrievers	52.5	52.5	36.4	40.6	48.2	37.9	38.1	40.8	45.1	56.4	40.7	44.5
w/o HN	52.2	47.4	32.6	39.9	46.7	35.3	36.6	39.8	45.5	53.2	38.5	42.5
w/ Sparse HN	54.3	45.6	34.8	41.5	49.3	35.7	37.1	41.5	46.1	50.9	39.9	43.3

(b) Recall@100

Methods	Ar	Bn	En	Fi	Id	Ja	Ko	Ru	Sw	Te	Th	Avg
Sparse	79.3	86.9	53.7	71.9	84.3	64.5	61.9	64.8	76.4	75.8	85.3	73.2
Dense	80.6	78.8	75.1	74.7	79.3	67.8	65.5	73.1	70.4	77.1	63.7	73.3
LeSTM w/o QG	**85.9**	89.2	78.9	**82.5**	**85.8**	75.2	73.6	**81.3**	**79.3**	**94.0**	**81.9**	**82.5**
Sparse + Dense	85.1	**89.6**	77.0	79.8	**85.8**	**77.4**	**75.5**	79.3	78.3	85.8	78.7	81.1
Sparse $\times$ Dense	84.5	**89.6**	75.3	78.9	85.0	76.1	75.1	78.6	76.5	85.7	78.7	80.4
Double Dense Retrievers	84.5	88.7	**79.5**	81.7	85.5	77.1	73.9	79.7	**79.3**	92.2	78.2	81.9
w/o HN	84.8	86.9	77.0	80.6	83.0	74.2	72.4	80.2	77.5	92.2	75.6	80.4
w/ Sparse HN	84.2	83.3	77.4	80.2	83.8	71.8	70.3	78.1	77.4	89.9	74.7	79.2