Empowering Dual-Encoder with Query Generator for Cross-Lingual Dense Retrieval

Houxing Ren1* Linjun Shou2 Ning Wu2 Ming Gong2 Daxin Jiang2†
1School of Computer Science and Engineering, Beihang University
2Microsoft STC Asia
renhouxing@buaa.edu.cn {lisho,wuning,migon,djiang}@microsoft.com

Abstract

In monolingual dense retrieval, lots of works focus on how to distill knowledge from cross-encoder re-ranker to dual-encoder retriever and these methods achieve better performance due to the effectiveness of cross-encoder re-ranker. However, we find that the performance of the cross-encoder re-ranker is heavily influenced by the number of training samples and the quality of negative samples, which is hard to obtain in the cross-lingual setting. In this paper, we propose to use a query generator as the teacher in the cross-lingual setting, which is less dependent on enough training samples and high-quality negative samples. In addition to traditional knowledge distillation, we further propose a novel enhancement method, which uses the query generator to help the dual-encoder align queries from different languages, but does not need any additional parallel sentences. The experimental results show that our method outperforms the state-of-the-art methods on two benchmark datasets.

1 Introduction

Information Retrieval (IR) aims to retrieve pieces of evidence for a given query. Traditional methods mainly use sparse retrieval systems such as BM25 (Robertson and Zaragoza, 2009), which 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, leveraging a cross-encoder re-ranker as the teacher model to distill knowledge to a dual-encoder has shown quite effective to boost the dual-encoder performance. Specifically, these methods first train a warm-up dual-encoder and a warm-up cross-encoder. Then, they perform knowledge distillation from the cross-encoder to the dual-encoder by KL-Divergence or specially designed methods. For example, RocketQAv2 (Qu et al., 2021) proposed dynamic distillation, and AR2 (Zhang et al., 2021) proposed adversarial training.

However, there are two major problems when scaling the method to the cross-lingual dense retrieval setting. Firstly, the cross-encoder typically requires large amounts of training data and high-quality negative samples due to the gap between pre-training (token-level task) and fine-tuning (sentence-level task), which are usually not satisfied in the cross-lingual setting (Asai et al., 2021a). Due to expensive labeling and lack of annotators in global languages, especially low-resource languages, the training data in cross-lingual are quite limited. Then with the limited training data, the dual-encoder is not good enough to provide...
high-quality negative samples to facilitate the cross-encoder. Secondly, the cross-lingual gaps between different languages have a detrimental effect on the performance of cross-lingual models. Although some cross-lingual pre-training methods such as InfoXLM (Chi et al., 2021) and LaBSE (Feng et al., 2022) have put lots of effort into this aspect by leveraging parallel corpus for better alignment between different languages, these parallel data are usually expensive to obtain and the language alignment could be damaged in the fine-tuning stage if without any constraint.

To solve these problems, we propose to employ a query generator in the cross-lingual setting, which uses the likelihood of a query against a passage to measure the relevance. On the one hand, the query generator can utilize pre-training knowledge with small training data in fine-tuning stage, because both of its pre-training and fine-tuning have a consistent generative objective. On the other hand, the query generation task is defined over all tokens from the query rather than just the [CLS] token in the cross-encoder, which has been demonstrated to be a more efficient training paradigm (Clark et al., 2020). As shown in Figure 1, with the number of training samples dropping, the performance of BERT-Large drops more sharply than T5-Base. Besides, the query generator is less sensitive to high-quality negative samples. As we can see, using BM25 as the retriever to mine negative samples for re-ranker training, the gap between cross-encoder and query generator is smaller than the gap using DPR as the retriever. Finally, the query generator can provide more training data by generation, which is precious in the cross-lingual setting. To sum up, the query generator is more effective than the cross-encoder in the cross-lingual setting.

Based on these findings, we propose a novel method, namely QuiCK, which stands Query generator improved dual-encoder by Cross-lingual Knowledge distillation. Firstly, at the passage level, we employ a query generator as the teacher to distill the relevant score between a query and a passage into the dual-encoder. Secondly, at the language level, we use the query generator to generate synonymous queries in other languages for each training sample and align their retrieved results by KL-Divergence. Considering the noise in the generated queries, we further propose a scheduled sampling method to achieve better performance.

The contributions of this paper are as follows:

- We propose a cross-lingual query generator as a teacher model to empower the cross-lingual dense retrieval model and a novel iterative training approach is leveraged for the joint optimizations of these two models.
- On top of the cross-lingual query generator, a novel cost-effective alignment method is further designed to boost the dense retrieval performance in low-resource languages, which does not require any additional expensive parallel corpus.
- Extensive experiments on two public cross-lingual retrieval datasets demonstrate the effectiveness of the proposed method.

2 Related Work

Retrieval. 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 such as DocTQuery (Nogueira et al., 2019) and DeepCT (Dai and Callan, 2020).

In contrast to sparse retrievers, dense retrievers usually employ a dual-encoder to encode both queries and passages into dense vectors whose lengths are much less than sparse vectors. These methods mainly focus on two aspects: pre-training (Lee et al., 2019; Guu et al., 2020; Lu et al., 2021; Gao and Callan, 2021a,b; Zhou et al., 2022) and fine-tuning methods, including negative sampling (Karpukhin et al., 2020; Luan et al., 2021; Xiong et al., 2021; Zhan et al., 2021) and multi-view representations (Khattab and Zaharia, 2020; Humeau et al., 2020; Tang et al., 2021; Zhang et al., 2022). Another fine-tuning method is jointly training the dual-encoder with a cross-encoder. For example, RDR (Yang and Seo, 2020) and FID-KD (Izacard and Grave, 2021) distill knowledge from a reader to the dual-encoder; RocketQA (Qu et al., 2021), PAIR (Ren et al., 2021a), RocketQA v2 (Ren et al., 2021b), and AR2 (Zhang et al., 2021) jointly train the dual-encoder with a cross-encoder to achieve better performance.

Recently, with the development of cross-lingual pre-trained models (Conneau et al., 2020), researchers pay more attention to cross-lingual dense
Figure 2: Overview of different model architectures designed for retrieval or re-ranking.

retrieval (Asai et al., 2021a; Longpre et al., 2020). For example, CORA (Asai et al., 2021b) leverages a generator to help mine retrieval training data, Sentri (Sorokin et al., 2022) proposes a single encoder and self-training, and DR.DECR (Li et al., 2021) uses parallel queries and sentences to perform cross-lingual knowledge distillation.

Re-ranking. Re-ranking aims to reorder the retrieved passages as the relevant scores. Due to the small number of retrieved passages, re-ranking usually employs high-latency methods to obtain better performance, e.g., cross-encoder. Traditionally, the re-ranking task is heavily driven by manual feature engineering (Guo et al., 2016; Hui et al., 2018). With the development of pre-trained language models (e.g., BERT), researchers use the pre-trained models to perform re-ranking tasks (Nogueira and Cho, 2019; Li et al., 2020). In addition to cross-encoder, researchers also try to apply generator to re-ranking. For example, monoT5 (Nogueira et al., 2020) proposes a prompt-based method to re-rank passages with T5 (Raffel et al., 2020) and other studies (dos Santos et al., 2020; Zhuang et al., 2021; Lesota et al., 2021) propose to use the log-likelihood of the query against the passage as the relevance to perform the re-ranking task.

Recently, with the size of pre-trained models scaling up, the generative models show competitive zero-shot and few-shot ability. Researchers start to apply large generative models to zero-shot and few-shot re-ranking. For example, SGPT (Muenighoff, 2022) and UPR (Sachan et al., 2022) propose to use generative models to perform zero-shot re-ranking. P^3 Ranker (Hu et al., 2022) demonstrates that generative models achieve better performance in the few-shot setting. Note that all of these works are concurrent to our work. Instead of using a query generator as a re-ranker only, we propose to leverage the query generator as a teacher model to enhance the performance of the cross-lingual dual-encoder. In addition to the traditional knowledge distillation, we further propose a novel cost-effective alignment method to boost the dense retrieval performance in low-resource languages.

3 Preliminaries

In this section, we give a brief review of dense retrieval and re-ranking. The overviews of all methods are presented in Figure 2.

Dual-Encoder. Given a query \( q \) and a large 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 (e.g., 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:

\[
f_{de}(q,p) = E_Q(q) \cdot E_P(p),
\]

where \( q \) 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 \( v_q = E_Q(q) \) and retrieve the top \( k \) passages with embeddings closest to \( v_q \).

Cross-Encoder Re-ranker. Given a query \( q \) and top \( k \) retrieved passages \( C \), the re-ranking task aims to reorder the passages as the relevant scores. Due to the limited size of the corpus, the re-ranking task usually employs a cross-encoder to perform interaction between words across queries and passages at the same time. These methods also introduce a special token [SEP] to separate \( q \) and \( p \), and then the hidden state of the [CLS] token from the cross-encoder is fed into a fully-connected layer to output the relevant score:

\[
f_{ce}(q,p) = W \times E_C(q||p) + b,
\]

where “||” denotes concatenation with the [SEP] token. During the inference stage, we apply the
cross-encoder $E_C(\cdot)$ to all <q, p> pair and reorder the passages by the scores.

**Query Generator Re-ranker.** Similar to cross-encoder re-ranker, query generator re-ranker also aims to reorder the passages as the relevant scores. For the query generator re-ranker, we use the log-likelihood of the query against the passage to measure the relevance:

$$f_{qg}(q, p) = \log P(q|p) = \sum_{t=0} \log P(q_t|q_{<t}, p),$$

where $q_{<t}$ denotes the previous tokens before $q_t$. The rest of settings are the same as the cross-encoder re-ranker and are omitted here.

**Training.** The goal of retrieval and re-ranking is to enlarge the relevant score between the query and the relevant passages (a.k.a., positive passages) and lessen the relevant score between the query and the irrelevant passages (a.k.a., negative passages). Let $\{q_1, p_1^+, p_1^-, p_{i,0}, p_{i,1}, \ldots, p_{i,n}\}$ be the $i$-th training sample. It consists of a query, a positive passage, and $n$ negative passages. Then we can employ the contrastive loss function, called InfoNCE (van den Oord et al., 2018), to optimize the model:

$$L_R = -\log \frac{e^f(q_i, p_i^+)}{e^f(q_i, p_i^+) + \sum_{j=0}^n e^f(q_i, p_{i,j})},$$

where $f$ denotes the similarity function, e.g., $f_{dc}$ in Eq. (1), $f_{ce}$ in Eq. (2), or $f_{qg}$ in Eq. (3).

**Cross-lingual Retrieval.** In the cross-lingual information retrieval task, passages and queries are in different languages. In this paper, we consider the passages are in English and the queries are in non-English languages. A sample consists of three components: a query in a non-English language, a positive passage in English, and a span answer in English. Given a non-English query, the task aims to retrieve relevant passages in English to answer the query. If a retrieved passage contains the given span answer, it is regarded as a positive passage, otherwise, it is a negative passage.

### 4 Methodology

In this section, we present the proposed QuiCK. The overview of the proposed method is presented in Figure 3. We start with the training of the query generator, then present how to perform distillation and alignment training for the dual-encoder, and we finally discuss the entire training process.

4.1 Training of Query Generator

In our method, we employ mT5 (Xue et al., 2021) as the query generator. The query generator has two roles: teacher and generator. As a teacher, it aims to better re-rank the candidate passages with relevance and distill the knowledge to the dual-encoder. As a generator, it aims to generate synonymous queries in different languages.

**Input Format.** As we employ mT5, we design a prompt template for input sentences. Considering that most passages are long, we propose introducing the span answer as input to encourage the generator to focus on the same segment and generate parallel queries in different languages. As a result, we use “generate [language] query: answer: [span answer] content: [content]” as the template. For a specific sample, we fill the three placeholders with the language of the target query, the span answer, and the passage content, respectively.

**Training.** Considering the two roles of the query generator, the entire training process for the query generator contains two stages: query generation training and re-ranking training.

Firstly, we train the generator with the generation task, which takes the positive passage as input and aims to generate the query. The task can be for-
mulated as maximizing the conditional probability:
\[
\hat{q} = \arg \max_{q} P(q|p,a) = \arg \max_{q} \prod_{t=0} \left[ P(q_{t}|p,a,q_{<t}) \right], \tag{5}
\]
where \(q_t\) is the \(t\)-th token of the generated query, \(a\) denotes the span answer, and \(q_{<t}\) represents the previous decoded tokens. Then we can employ cross-entropy loss to optimize the model:
\[
\mathcal{L}_{QG} = \frac{1}{T} \sum_{t=0}^{T} -\log P(q_{t}|p,a,q_{<t}), \tag{6}
\]
where \(T\) denotes the number of the query tokens.

Secondly, we train the generator with the re-ranking task, which takes a query and a passage as input and outputs the relevant score of the two sentences. The detailed training process is introduced in Section 3 and is omitted here.

4.2 Distillation for Dual-Encoder
We then present how to distill knowledge from query generator to dual-encoder. Similar to previous methods (Ren et al., 2021b; Zhang et al., 2021), we employ KL-Divergence to perform distillation. Formally, given a query \(q\) and a candidate passage set \(C_q = \{p_i\}_{1 \leq i \leq n}\) which is retrieved by the dual-encoder, we compute relevant scores by query generator and dual-encoder, respectively. After that, we normalize the scores by softmax and compute the KL-Divergence as the loss:
\[
s_{qg}(q,p) = \frac{\exp(f_{qg}(q,p))}{\sum_{p' \in C_q} \exp(f_{qg}(q,p'))},
\]
\[
s_{de}(q,p) = \frac{\exp(f_{de}(q,p))}{\sum_{p' \in C_q} \exp(f_{de}(q,p'))}, \tag{7}
\]
\[
\mathcal{L}_D = \sum_{p \in C_q} s_{qg}(q,p) \frac{s_{qg}(q,p)}{s_{de}(q,p)},
\]
where \(f_{qg}\) and \(f_{de}\) denote the relevant score given by the query generator and the dual-encoder which are presented in Section 3.

4.3 Alignment for Dual-Encoder
Alignment is a common topic in the cross-lingual setting, which can help the model better handle sentences in different languages. Previous works (Zheng et al., 2021; Yang et al., 2022) usually use parallel data or translated data to perform alignment training among different languages. Here, we propose a novel method to align queries in different languages for cross-lingual retrieval, which does not need any parallel data. The core idea of our method is to leverage the query generator to generate synonymous queries in other languages to form parallel cases.

Generation. For each case in the training set, we generate a query in each target language (a.k.a., if there are seven target languages, we generate seven queries for the case). Then we use the confidence of the generator to filter the generated queries. Specifically, we set filter thresholds to accept 50% of generated queries.

Scheduled Sampling. In this work, we select a generated query to form a pair-wise case with the source query. Considering the semantics of generated queries, we carefully design a scheduled sampling method to replace the random sampling. For a generated query \(q'\), we first use the dual-encoder to retrieve passages for the source query \(q\) and generated query \(q'\), respectively, namely \(C_q\) and \(C'_{q'}\). Then we calculate a coefficient for the generated query \(q'\) as
\[
\epsilon' = \frac{|C_q \cap C'_{q'}|}{\max(|C_q|, |C'_{q'}|)}, \tag{8}
\]
\[
\epsilon' = \begin{cases} 
\epsilon' & \text{if } \epsilon' \geq T, \\
0 & \text{if } \epsilon' < T,
\end{cases}
\]
where threshold \(T\) is a hyper-parameter and \(| \cdot |\) denotes the size of the set. The basic idea is that the larger the union of retrieved passages, the more likely the queries are to be synonymous. When sampling the generated query, we first calculate coefficients \(\{\epsilon'_1, \ldots, \epsilon'_m\}\) for all generated queries \(\{q'_1, \ldots, q'_m\}\), then normalize them as the final sampling probability \(p\):
\[
p_i = \frac{\epsilon'_i}{\sum_{j=0}^{m} \epsilon'_j}, \tag{9}
\]
where \(m\) denotes the number of generated queries.

During the training stage, for each training case, we sample a generated query to form the pair-case with the source query \(q\) based on the probabilities.

Alignment Training. After sampling a generated query, we present the how to align the source query and the generated query. Different to previous works (Zheng et al., 2021), we employ asymmetric KL-Divergence rather than symmetric KL-Divergence due to the different quality of the source
Algorithm 1: The training algorithm.


1. Initialize $R$ and $G$ with pre-trained model;
2. Train the warm-up $R$ with Eq. (4) on $D$;
3. Train the warm-up $G$ with Eq. (6) on $D$;
4. Generate queries for each sample in $D$;
5. Build ANN index for $R$;
6. Retrieve relevant passages on corpus $C$;
7. Fine-tune the $G$ with Eq. (4) on $D$ and retrieved negative passages.
8. while models has not converged do
   9.     Fine-tune the $R$ with Eq. (11) on $D$ and retrieved passages;
   10.    Refresh ANN index for $R$;
   11.    Retrieve relevant passages on corpus $C$;
   12.    Fine-tune the $G$ with Eq. (4) on $D$ and retrieved negative passages.
end

Query and the generated query:

$$L_A = \sum_{p \in C_q \cup C'_q} c'_s \frac{\delta_{de}(q, p)}{\delta_{de}(q', p)} s_{de}(q, p)$$

where $q$ denotes the query, $C_q$ denotes the set of retrieved passages, superscript “r” denotes the generated case, and $c'$ is the coefficient of the generated query. Note that $\delta_{de}$ in Eq. (10) are normalized across $C_q \cup C'_q$ instead of $C_q$ or $C'_q$ in Eq. (7).

4.4 Training of Dual-Encoder

As shown in Figure 3, we combine the distillation loss and the alignment loss as final loss:

$$L = L_D + L'_D + \alpha \times L_A$$

where $L_D$ denotes the distillation loss for the source queries, $L'_D$ denotes the distillation loss for the generated queries, $L_A$ denotes the alignment loss, and $\alpha$ is a hyper-parameter to balance the loss.

5.1 Experimental Setup

Datasets. We evaluate the proposed method on two public cross-lingual retrieval datasets: XOR-Retrieve (Asai et al., 2021a) and MKQA (Longpre et al., 2020). The detailed descriptions of the two datasets are presented in Appendix A.

Evaluation Metrics. Following previous works (Asai et al., 2021a; Sorokin et al., 2022), we use R@2kt and R@5kt as evaluation metrics for the XOR-Retrieve dataset and R@2kt as evaluation metrics for the MKQA dataset. The metrics measure the proportion of queries to which the top k retrieved tokens contain the span answer, which is fairer with different passage sizes.

Implementation Details. For the warm-up training stage, we follow XOR-Retrieve to first train the model on NQ (Kwiatkowski et al., 2019) data and then fine-tune the model with XOR-Retrieve data. For the iteratively training stage, we generate seven queries for each case (because the XOR-Retrieve data contains seven languages). We set the number of retrieved passages as 100, the number of iterations as 5, threshold $T$ in Eq. (8) as 0.3 and coefficient $\alpha$ in Eq. (11) as 0.5. The detailed hyper-parameters are shown in Appendix C. And we conduct more experiments to analyze the parameter sensitivity in Appendix D.

All the experiments run on 8 NVIDIA Tesla A100 GPUs. The implementation code is based on HuggingFace Transformers (Wolf et al., 2020). For the dual-encoder, we use XLM-R Base (Conneau et al., 2020) as the pre-trained model and use the average hidden states of all tokens to represent the sentence. For the query generator, we leverage mT5 Base (Xue et al., 2021) as the pre-trained model, which has almost the same number of parameters as a large cross-encoder.

5.2 Results

Baselines. We compare the proposed QuiCK with previous state-of-the-art methods, including mDPR, DPR+MT (Asai et al., 2021a), Sentri (Sorokin et al., 2022), DR.DECR (Li et al., 2021). Note that Sentri introduces a shared encoder with large size, DR.DECR introduces parallel queries and parallel corpus, but our method only utilizes an encoder with base size, XOR-Retrieve and NQ training data. For more fairly comparison, we also report their ablation results. Here, “Bi-Encoder” denotes two unshared encoders with base size. “KDxor” de-
Table 1: Comparison results on XOR-Retrieve dev set. The best results are in bold. “∗” denotes the results are copied from the source paper. Results unavailable are left blank.

<table>
<thead>
<tr>
<th>Methods</th>
<th>R@2kt</th>
<th>R@5kt</th>
<th>Avg</th>
<th>R@2kt</th>
<th>R@5kt</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ar</td>
<td>Bn</td>
<td>Fi</td>
<td>Ja</td>
<td>Ko</td>
<td>Ru</td>
</tr>
<tr>
<td>mDPR∗</td>
<td>38.8</td>
<td>48.4</td>
<td>52.5</td>
<td>26.6</td>
<td>44.2</td>
<td>33.3</td>
</tr>
<tr>
<td>DPR + MT∗</td>
<td>43.4</td>
<td>53.9</td>
<td>55.1</td>
<td>40.2</td>
<td>50.5</td>
<td>30.8</td>
</tr>
<tr>
<td>Sentri∗</td>
<td>47.6</td>
<td>48.1</td>
<td>53.1</td>
<td>46.6</td>
<td>49.6</td>
<td>44.3</td>
</tr>
<tr>
<td>w/ Bi-Encoder</td>
<td>47.8</td>
<td>39.1</td>
<td>48.9</td>
<td>51.2</td>
<td>40.2</td>
<td>41.2</td>
</tr>
<tr>
<td>DR.DECR∗</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>w/o KD&lt;sub&gt;XR&lt;/sub&gt;</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>w/o KD&lt;sub&gt;PC&lt;/sub&gt;</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>QuiCK</td>
<td>52.8</td>
<td>70.1</td>
<td>62.2</td>
<td>54.8</td>
<td>62.8</td>
<td>57.8</td>
</tr>
<tr>
<td>QuiCK w/ LaBSE</td>
<td>67.3</td>
<td>78.9</td>
<td>65.9</td>
<td>59.8</td>
<td>66.3</td>
<td>63.7</td>
</tr>
</tbody>
</table>

In addition, we show the results on XOR-Retrieve test set in Table 2, which is copied from the leaderboard on June 15, 2022. As we can see, our method achieves the top position on the leaderboard of XOR-Retrieve.

**XOR-Retrieve.** Table 1 shows the results on XOR-Retrieve dev set. The proposed QuiCK outperforms mDPR, DPR+MT, and Sentri with a clear edge in almost all languages. Although QuiCK does not introduce any parallel corpus, it also outperforms DR.DECR w/o KD<sub>XR</sub>. Finally, QuiCK based on LaBSE outperforms all baselines, especially DR.DECR w/o KD<sub>XR</sub>, and even outperforms DR.DECR which utilizes both parallel queries and parallel corpus. Note that knowledge distillation with parallel corpus in DR.DECR is designed for cross-lingual dense retrieval, but LaBSE is a general pre-trained model for all cross-lingual tasks. These results show the effectiveness of the proposed QuiCK. Our method combines two methods in dense retrieval and cross-lingual tasks, namely distillation and alignment. We further analyze the contribution of each component in Section 5.3.

In addition, we show the results on XOR-Retrieve test set in Table 2, which is copied from the leaderboard on June 15, 2022. As we can see, our method achieves the top position on the leaderboard of XOR-Retrieve.

**MKQA.** Furthermore, we evaluate the zero-shot performance of our method on the MKQA test set. Following previous works (Sorokin et al., 2022), we directly evaluate the dual-encoder training on XOR-Retrieve data and report the performance of unseen languages on MKQA. As shown in Table 3, our method outperforms all baselines and even performs better than Sentri. Note that Sentri uses a shared encoder with large size. The comparison between Sentri and Sentri w/ Bi-Encoder shows that the large encoder has better transfer ability. Finally, the proposed QuiCK w/ LaBSE outperforms all baselines with a clear edge. It shows the better transfer ability of our methods.

### 5.3 Methods Analysis

**Ablation Study.** Here, we check how each component contributes to the final performance. We construct the ablation experiments on XOR-Retrieve data. We prepare four variants of our method:

<table>
<thead>
<tr>
<th>Methods</th>
<th>R@2kt</th>
<th>R@5kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAAMA</td>
<td>52.8</td>
<td>59.9</td>
</tr>
<tr>
<td>Sentri</td>
<td>52.7</td>
<td>61.0</td>
</tr>
<tr>
<td>CCP</td>
<td>54.8</td>
<td>63.0</td>
</tr>
<tr>
<td>Sentri 2.0</td>
<td>58.5</td>
<td>64.6</td>
</tr>
<tr>
<td>DR.DECR</td>
<td>63.0</td>
<td>70.3</td>
</tr>
<tr>
<td>QuiCK w/ LaBSE</td>
<td>65.6</td>
<td>72.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>R@2kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORA∗</td>
<td>41.1</td>
</tr>
<tr>
<td>BM25 + MT∗</td>
<td>42.0</td>
</tr>
<tr>
<td>Sentri∗ w/ Bi-Encoder∗</td>
<td>53.3</td>
</tr>
<tr>
<td>QuiCK</td>
<td>53.4</td>
</tr>
<tr>
<td>QuiCK w/ LaBSE</td>
<td>60.3</td>
</tr>
</tbody>
</table>

In Table 2, the average performance of 20 unseen languages in MKQA test set. “∗” denotes the results are copied from the Sentri paper.
Table 4: Ablation results on XOR-Retrieve dev set.

<table>
<thead>
<tr>
<th>Methods</th>
<th>R@2kt</th>
<th>R@5kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuiCK</td>
<td>61.3</td>
<td>68.9</td>
</tr>
<tr>
<td>w/o Sampling</td>
<td>59.5</td>
<td>67.5</td>
</tr>
<tr>
<td>w/o Alignment</td>
<td>59.9</td>
<td>67.1</td>
</tr>
<tr>
<td>w/o Generation</td>
<td>58.8</td>
<td>65.9</td>
</tr>
<tr>
<td>w/o All</td>
<td>41.5</td>
<td>53.4</td>
</tr>
</tbody>
</table>

Table 5: Effect of alignment based on different pre-trained languages models.

<table>
<thead>
<tr>
<th>Methods</th>
<th>XLM-R</th>
<th>LabSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@2kt</td>
<td>R@5kt</td>
</tr>
<tr>
<td>QuiCK</td>
<td>61.3</td>
<td>68.9</td>
</tr>
<tr>
<td>w/o Alignment</td>
<td>59.9</td>
<td>67.1</td>
</tr>
</tbody>
</table>

(1) w/o Sampling denotes without the scheduled sampling but keep the threshold $T$ for $c'$, a.k.a., if $c' \geq T$, then $c' = 1$, otherwise $c' = 0$; (2) w/o Alignment denotes without $L_A$ in Eq. (11); (3) w/o Generation denotes without $L'_D$ and $L_A$ in Eq. (11); (4) w/o All denotes without the enhanced training, a.k.a., the warm-up dual-encoder.

Table 4 presents all comparison results of the four variants. As we can see, the performance rank of R@5kt can be given as: w/o All < w/o Generation < w/o Alignment < w/o Sampling < QuiCK. These results indicate that all components are essential to improve performance. And we can find the margin between w/o Alignment and w/o Sampling is small, it denotes that the generated queries are noisy and demonstrate the effectiveness of our schedule sampling strategy.

Effect of Alignment. As we mentioned in Section 1, the alignment established in the pre-training stage may be damaged without any constraint in the fine-tuning stage. Here, we construct experiments on both XLM-R and LabSE to analyze the effectiveness of the proposed alignment training. As shown in Table 5, the proposed alignment training is effective based on the two models. It indicates that the alignment constraint in the fine-tuning stage is effective for models which pre-trained with parallel corpus. And we find that the gains of alignment training based on XLM-R are larger than LabSE, which shows that the alignment constraint is more effective for models which do not pre-trained with parallel corpus.

Cross-Encoder versus Query Generator. Here, we analyze the re-ranking ability of cross-encoder and query generator. Here, we use the warm-up dual-encoder to retrieve passages, vary the number of candidate passages, and then evaluate the re-ranked result. As shown in Figure 4, when we use the top-100 candidate passages, the performance of the cross-encoder and generator is almost the same. But as the number of candidate passages increases, especially when it surpasses 500, the gap between the performance of the cross-encoder and query generator gradually becomes larger. It shows low generalization performance of the cross-encoder when there are not enough training samples.

Visualization of the Training Procedure. We visualize the performance changes of R@2kt during the iteratively training on XOR-Retrieve dev set. Here, “QG” denotes Query Generator and “CE” denotes Cross Encoder.
improved by approximately 20%. Finally, comparing the performance of the cross-encoder and the query generator, we can find that there are approximately 6% gaps for both teachers and students. It shows the effectiveness of our method.

6 Conclusion

In this paper, we showed that the cross-encoder performs poorly when there are not sufficient training samples which are hard to obtain in the cross-lingual setting. Then we proposed a novel method that utilizes the query generator to improve the dual-encoder. We firstly proposed to use a query generator as the teacher. After that, we proposed a novel alignment method for cross-lingual retrieval which does not need any parallel corpus. Extensive experimental results show that the proposed method outperforms the baselines and significantly improves the state-of-the-art performance. Currently, our method depends on training data in all target languages. As future work, we will investigate how to perform the proposed method for zero-shot cross-lingual dense retrieval.

7 Limitations

The limitations are summarized as follows.

- The method depends on training data in all target languages. Intuitively, the method can be directly applied to the zero-shot cross-lingual dense retrieval if we only take the passage as input for the query generator, but the query generator performs poorly in the zero-shot setting. As future work, novel pre-training tasks for cross-lingual generation can be considered.

- The method does not investigate how to effectively train the query generator for the re-ranking task, just directly applies the training method for the cross-encoder re-ranker. We believe the potential of query generators for re-ranking is strong and designing a special re-ranking training method for query generators such as token-level supervision may be interesting for future work.

- The method requires large GPU sources. The final model approximately costs 12 hours on 8 NVIDIA Tesla A100 GPUs. Although researchers who do not have enough GPU sources can use the “gradient accumulation” technique to reduce GPU memory consumption, they also need to pay more time.

- This work does not consider the inconsistency between different countries (e.g., law and religion), which leads to inconsistent positive passages for synonymous queries in different languages (e.g., the legal age of marriage varies from country to country). Because we find that most of the queries in XOR-Retrieve contain the target country such as “Mikä on yleisin kissa laji Suomessa?” (translation: What is the most common cat breed in Finland?).

References


Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In ICLR. OpenReview.net.


Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Ari-
vazhagan, 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 cor-
pus aware language model pre-training for dense pas-

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce
Croft. 2016. A deep relevance matching model for
ad-hoc retrieval. In CIKM, pages 55–64. ACM.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasu-
pat, and Ming-Wei Chang. 2020. REALM: retrieval-
augmented language model pre-training. CoRR,

Xiaomeng Hu, Shi Yu, Chenyan Xiong, Zhenghao
Liu, Zhiyuan Liu, and Ge Yu. 2022. P3 \textsuperscript{3} ranker:
Mitigating the gaps between pre-training and rank-
ing fine-tuning with prompt-based learning and pre-
finetuning. CoRR, abs/2205.01886.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard
IR model for ad-hoc retrieval. In WSDM, pages 279–
287. ACM.

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

Gautier Izacard and Edouard Grave. 2021. Distilling
knowledge from reader to retriever for question an-
swering. In ICLR. OpenReview.net.

Billion-scale similarity search with gpus. IEEE

Vladimir Karpukhin, Barlas Ogunz, 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: Effi-
cient and effective passage search via contextualized
late interaction over BERT. In SIGIR, pages 39–48.
ACM.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Red-
field, Michael Collins, Ankur P. Parikh, Chris Alberti,
Danielle Epstein, Illia Polosukhin, Jacob Devlin, Ken-
ton Lee, Kristina Toutanova, Llion Jones, Matthew
Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob
Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natu-
ral questions: a benchmark for question answering
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.

Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus An-
tonius Grasserbauer, Carsten Eickhoff, and Markus
Schedl. 2021. A modern perspective on query likeli-
hood with deep generative retrieval models. In ICTIR,
pages 185–195. ACM.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He,
and Yingfei Sun. 2020. PARADE: passage represen-
tation aggregation for document reranking. CoRR,
abs/2008.09093.

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.

Shayne Longpre, Yi Lu, and Joachim Daiber. 2020.
MKQA: A linguistically diverse benchmark for mul-

Shuqi Lu, Di He, Chenyan Xiong, Guolun 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
Association for Computational Linguistics.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and
Michael Collins. 2021. Sparse, dense, and atten-
tional representations for text retrieval. Trans. Assoc.

Niklas Muennighoff. 2022. SGPT: GPT sentence

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and
Jimmy Lin. 2020. Document ranking with a pre-
trained sequence-to-sequence model. In EMNLP
(Findings), volume EMNLP 2020 of Findings of ACL,
pages 708–718. Association for Computational
Linguistics.

From doc2query to doctttttquery. Online preprint.

Rodrigo Frassetto Nogueira and Kyunghyun Cho.
2019. Passage re-ranking with BERT. CoRR, abs/1901.04085.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruyi Yang
Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and
Haifeng Wang. 2021. Rocketqa: An optimized train-
ing approach to dense passage retrieval for open-
domain question answering. In NAACL-HLT, pages
Appendix

A Dataset

**XOR-Retrieve.** XOR-Retrieve dataset is a cross-lingual retrieval dataset which aims to retrieve relevant passages from the English corpus for non-English queries. XOR-Retrieve data contains queries in seven typologically diverse languages: Arabic (Ar), Bengali (Bn), Finnish (Fi), Japanese (Ja), Korean (Ko), Russian (Ru), and Telugu (Te). The statistics are presented in Table 6.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ar</td>
<td>2,574</td>
<td>350</td>
<td>137</td>
</tr>
<tr>
<td>Bn</td>
<td>2,582</td>
<td>312</td>
<td>128</td>
</tr>
<tr>
<td>Fi</td>
<td>2,088</td>
<td>360</td>
<td>530</td>
</tr>
<tr>
<td>Ja</td>
<td>2,288</td>
<td>296</td>
<td>449</td>
</tr>
<tr>
<td>Ko</td>
<td>2,469</td>
<td>299</td>
<td>646</td>
</tr>
<tr>
<td>Ru</td>
<td>1,941</td>
<td>366</td>
<td>235</td>
</tr>
<tr>
<td>Te</td>
<td>1,308</td>
<td>238</td>
<td>374</td>
</tr>
</tbody>
</table>

**Corpus size** 18,003,200

**MKQA.** MKQA dataset is a translated dataset of 10,000 query-answer pairs from NQ to 26 different languages, and the dataset is only used for evaluation. In our experiments, since we measure the R@2kt score, we filter the samples which do not have span answers. Then, we get 6,620 parallel queries in each language. Finally, we directly evaluate the dual-encoder trained on XOR-Retrieve data and use the same corpus with XOR-Retrieve in the experiments. Note that Arabic (Ar), English (En), Finnish (Fi), Japanese (Ja), Korean (Ko), and Russian (Ru) are seen in the training stage and we only report the performance of the rest 20 languages. As a result, it has a total of 132,400 samples.

B Efficiency Report

We list the time cost of training and inference in Table 7 which is made with 8 NVIDIA A100 GPUs.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warm-up</td>
<td>3h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Per Iteration</td>
<td>1h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dual-Encoder</td>
<td>0.3h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index Refresh</td>
<td>0.35h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>11.5h</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Build Index</th>
<th>Query Encoding</th>
<th>Dense Retrieval</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.35h</td>
<td>40ms</td>
<td>2ms</td>
</tr>
</tbody>
</table>

C Hyper-parameters

We present all hyper-parameters in Table 8.

D Additional Experiments

D.1 Parameter Sensitivity

In this section, we tune the parameters of the proposed method to analyze parameter sensitivity. We vary both the threshold \( T \) (Eq. (8)) and the coefficient \( \alpha \) (Eq. (11)) in the set \{0.1, 0.3, 0.5, 0.7, 0.9\}. We report the tuning results with both R@2kt and R@5kt on the XOR-Retrieve dev set in Figure 6. As we can see, \( T = 0.3 \) and \( \alpha = 0.5 \) lead to the optimal R@5kt which is the ordering basis on
D.2 Effect of The Number of Candidates

Here, we investigate the effect of the number of candidates which is demonstrated to have a significant effect on the final performance. As shown in Figure 7, a large number of candidates leads to better performance. And when the number surpasses 32, the improvement gradually slows down. The results indicate that 32 candidates can better represent the whole corpus.

D.3 Effect of Model Size

Following Sentri (Sorokin et al., 2022), we employ a shared encoder (i.e., the parameters of the query encoder and the passage encoder are the same) with large size as the dual-encoder and evaluate our method. As shown in Table 9, QuiCK with XLM-R Large achieves a significant performance improvement, which further demonstrates the effectiveness of the proposed QuiCK.

D.4 Effect of Scheduled Sampling

Previously, we demonstrate the effectiveness of the scheduled sampling by ablation study. Here, we present two generated samples to qualitatively analyze the scheduled sampling. As shown in Table 10, for the first case, the generated queries in other languages have the same semantics as the query from the source language. The sample is effective in alignment training and is helpful to achieve better performance. For the second case, the generated query in Finnish is relevant to the query from the source language but not synonymous. The sample is harmful to the model training. These samples indicate that the scheduled sampling is necessary for alignment training. In this way, we can reduce the impact of the cases which do not have the same semantics and further achieve better performance.

D.5 Effect of Span Answer

In our method, we employ the span answer to encourage the query generator to generate synonymous queries. Here, we conduct experiments to evaluate the effect of the span answer. Specially, we use another template: “generate [language] query: [content]” where we only need to fill two placeholders with the language of the target query and the leaderboards.

In addition, we find that the optimal R@2kt and R@5kt are led by different thresholds $T$. Because the two metrics have different sensitivities to data quality, low-quality data is helpful to R@5kt but harmful to R@2kt. As a result, a small threshold $T$ leads to more low-quality alignment training data and further leads to higher R@5kt but lower R@2kt. On the contrary, the optimal R@2kt and R@5kt are led by the same coefficient $\alpha$.

Overall, our model is relatively stable when varying the two parameters, and consistently better than Sentri and Dr.DECR w/o KDPC.

Table 9: Effect of model size.

<table>
<thead>
<tr>
<th></th>
<th>XLM-R Base</th>
<th>XLM-R Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>R@2kt</td>
<td>R@5kt</td>
<td>R@2kt</td>
</tr>
<tr>
<td>Ar</td>
<td>52.8</td>
<td>64.4</td>
</tr>
<tr>
<td>Bn</td>
<td>70.1</td>
<td>77.0</td>
</tr>
<tr>
<td>Fi</td>
<td>62.2</td>
<td>67.0</td>
</tr>
<tr>
<td>Ja</td>
<td>54.8</td>
<td>56.4</td>
</tr>
<tr>
<td>Ko</td>
<td>62.8</td>
<td>67.4</td>
</tr>
<tr>
<td>Ru</td>
<td>57.8</td>
<td>66.7</td>
</tr>
<tr>
<td>Te</td>
<td>70.6</td>
<td>79.4</td>
</tr>
<tr>
<td>Avg</td>
<td>61.3</td>
<td>68.4</td>
</tr>
</tbody>
</table>
Table 10: Two generated examples. The span answers are in bold.

**Passage:** Johanna Maria Magdalena "Magda" Goebbels (née Ritschel; 11 November 1901 – 1 May 1945) was the wife of Nazi Germany's Propaganda Minister Joseph Goebbels. A prominent member of the Nazi Party, she was a close ally, companion and political supporter of Adolf Hitler. Some historians refer to her as the unofficial “First Lady” of Nazi Germany, while others give that title to Emmy Göring.

**Source Query (Ja):** ヨハンナ・マリア・マクダレナ・ゲッベルスは何歳で死去した？
**Translation:** At what age did Johanna Maria McDalena Goebbels die?

**Generated Query (Ru):** В каком возрасте умерла Магда Геббельс?
**Translation:** At what age did Magda Goebbels die?

**Generated Query (Fi):** Minä vuonna Magda Goebbels kuoli?
**Translation:** In what year did Magda Goebbels die?

**Passage:** Charles V (24 February 1500 – 21 September 1558) was ruler of both the Holy Roman Empire from 1519 and the Spanish Empire (as Charles V of Spain) from 1516, as well as of the lands of the former Duchy of Burgundy from 1506. He stepped down from these and other positions by a series of abdications between 1554 and 1556. Through inheritance, he brought together under his rule extensive territories in western, central, and southern Europe, and the Spanish vicerealties in the Americas and Asia.

**Source Query (Ko):** 신성 로마 제국 카를 5세 재위 기간은 얼마나 되나요?
**Translation:** How long was the reign of Charles V of the Holy Roman Empire?

**Generated Query (Ru):** Сколько лет правил Карл V?
**Translation:** How many years did Charles V rule?

**Generated Query (Fi):** Minä vuonna Charles V hallitsi Rooman valtakuntaa?
**Translation:** In what year did Charles V rule the Roman Empire?

As input leads to better performance on re-ranking tasks for the query generator. On the other hand, both the two query generator is better than the cross-encoder when re-ranking top-1000 retrieved passages, it shows the effectiveness of the query generator in the cross-lingual setting.

In addition, we show queries generated by the two query generators in Table 11. As we can see, for the query generator that does not take the span answer as input, the generated queries can be answered by the passage, but they focus on different segments of the passage and they are not synonymous. On contrary, for the query generator that takes the span answer as input, generated queries can be answered by the passage and they are synonymous. It shows that taking the span answer as input can effectively encourage the generator to generate synonymous queries.

### E Detailed Results

Due to the limited space, we only present average performance for some experiments in Section 5. Here, we present the detailed performance in all languages of these experiments. Firstly, we present the detailed performance of all methods on the MKQA test set in Table 13. Secondly, we present the detailed performance of ablation results in Table 14. Finally, we present the detailed performance for evaluating the effect of alignment in Table 15.
Table 12: Performance comparison of different re-rankers on XOR-Retrieve dev set. The best results are in bold. Here, "QG" denotes query generator.

<table>
<thead>
<tr>
<th>Methods</th>
<th>R@2kt</th>
<th></th>
<th>R@5kt</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ar</td>
<td>Bn</td>
<td>Fi</td>
<td>Ja</td>
</tr>
<tr>
<td>Dual-Encoder</td>
<td>35.3</td>
<td>43.1</td>
<td>50.3</td>
<td>35.7</td>
</tr>
<tr>
<td>Re-ranking top-100 retrieved passages</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QG w/ Answer</td>
<td>51.1</td>
<td>57.2</td>
<td>53.5</td>
<td>43.2</td>
</tr>
<tr>
<td>QG w/o Answer</td>
<td>4.89</td>
<td>55.6</td>
<td>54.1</td>
<td>41.9</td>
</tr>
<tr>
<td>Cross-Encoder</td>
<td>49.2</td>
<td>53.6</td>
<td>57.6</td>
<td>41.1</td>
</tr>
</tbody>
</table>

Table 13: Detailed performance on MKQA test set. "∗" denotes that the results are copied from the Sentri paper.

| Methods                        | Da | De | Es | Fr | Hu | It | Km | Ms | Ni | No | Pt | Sv | Th | Tr | Vi | Zh-cn | Zh-hk | Zh-tw | Avg |
|--------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|-------|-------|-----|
| CORA                          | 44.5| 44.6| 45.3| 44.8| 27.3| 39.1| 44.2| 22.2| 44.3| 47.3|    |    |    |    |       |       |       |     |
| BM25 + MT                     | 44.1| 43.3| 44.9| 42.5| 36.9| 39.3| 40.1| 31.3| 42.5| 46.5|    |    |    |    |       |       |       |     |
| Sentri                        | 57.6| 56.5| 55.9| 55.1| 47.9| 51.8| 54.3| 43.9| 56.0| 56.3|    |    |    |    |       |       |       |     |
| w/ Bi-Encoder                 | 30.0| 47.8| 48.7| 47.4| 37.7| 43.4| 41.8| 37.8| 49.5| 47.3|    |    |    |    |       |       |       |     |
| QuiCK                         | 58.3| 56.4| 55.2| 55.5| 44.7| 52.4| 52.3| 42.0| 56.9| 57.5|    |    |    |    |       |       |       |     |
| QuiCK w/ LaBSE                | 63.3| 61.8| 62.2| 62.4| 56.1| 58.9| 60.6| 53.0| 64.2| 63.0|    |    |    |    |       |       |       |     |
| No                             | 48.3| 44.8| 40.8| 43.6| 45.0| 34.8| 33.9| 33.5| 41.5| 41.0|    |    |    |    |       |       |       |     |
| w/o Alignment                 | 43.3| 46.5| 45.7| 49.7| 46.5| 42.5| 43.5| 37.5| 37.5| 41.2|    |    |    |    |       |       |       |     |
| w/o Generation                | 56.5| 55.8| 54.8| 56.9| 55.3| 53.0| 54.4| 50.2| 50.7| 49.4|    |    |    |    |       |       |       |     |
| w/o All                       | 49.1| 47.0| 47.7| 50.0| 46.5| 45.6| 47.3| 42.6| 41.5| 45.3|    |    |    |    |       |       |       |     |

Table 14: Detailed performance for ablation study on XOR-Retrieve dev set.

<table>
<thead>
<tr>
<th>Methods</th>
<th>R@2kt</th>
<th></th>
<th>R@5kt</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ar</td>
<td>Bn</td>
<td>Fi</td>
<td>Ja</td>
</tr>
<tr>
<td>QuiCK</td>
<td>52.8</td>
<td>70.1</td>
<td>60.2</td>
<td>54.8</td>
</tr>
<tr>
<td>w/o Sampling</td>
<td>53.1</td>
<td>68.4</td>
<td>60.2</td>
<td>50.2</td>
</tr>
<tr>
<td>w/o Alignment</td>
<td>51.8</td>
<td>68.1</td>
<td>60.8</td>
<td>51.0</td>
</tr>
<tr>
<td>w/o Generation</td>
<td>55.0</td>
<td>68.1</td>
<td>59.6</td>
<td>47.3</td>
</tr>
<tr>
<td>w/o All</td>
<td>35.3</td>
<td>43.1</td>
<td>50.3</td>
<td>35.7</td>
</tr>
</tbody>
</table>

Table 15: Detailed performance of alignment based on different pre-trained languages models.

<table>
<thead>
<tr>
<th>Methods</th>
<th>R@2kt</th>
<th></th>
<th>R@5kt</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ar</td>
<td>Bn</td>
<td>Fi</td>
<td>Ja</td>
</tr>
<tr>
<td>XLM-R</td>
<td>52.8</td>
<td>70.1</td>
<td>60.2</td>
<td>54.8</td>
</tr>
<tr>
<td>w/o Alignment</td>
<td>51.8</td>
<td>68.1</td>
<td>60.8</td>
<td>51.0</td>
</tr>
<tr>
<td>LaBSE</td>
<td>67.3</td>
<td>78.9</td>
<td>65.9</td>
<td>59.8</td>
</tr>
<tr>
<td>w/o Alignment</td>
<td>65.7</td>
<td>78.3</td>
<td>65.0</td>
<td>58.9</td>
</tr>
</tbody>
</table>

3121