Optimizing Dense Retrieval Model Training with Hard Negatives

Jingtao Zhan
BNRist, DCST, Tsinghua University
jingtaozhan@gmail.com

Jiaxin Mao
GSAI, Renmin University of China
maojiaxin@gmail.com

Yiqun Liu*
BNRist, DCST, Tsinghua University
yiqunliu@tsinghua.edu.cn

Jiafeng Guo
University of Chinese Academy of
Sciences, Institute of Computing
Technology, CAS
guojiafeng@ict.ac.cn

Min Zhang
BNRist, DCST, Tsinghua University
z-m@tsinghua.edu.cn

Shaoping Ma
BNRist, DCST, Tsinghua University
msp@tsinghua.edu.cn

ABSTRACT

Ranking has always been one of the top concerns in information retrieval researches. For decades, the lexical matching signal has dominated the ad-hoc retrieval process, but solely using this signal in retrieval may cause the vocabulary mismatch problem. In recent years, with the development of representation learning techniques, many researchers turn to Dense Retrieval (DR) models for better ranking performance. Although several existing DR models have already obtained promising results, their performance improvement heavily relies on the sampling of training examples. Many effective sampling strategies are not efficient enough for practical usage, and for most of them, there still lacks theoretical analysis in how and why performance improvement happens. To shed light on these research questions, we theoretically investigate different training strategies for DR models and try to explain why hard negative sampling performs better than random sampling. Through the analysis, we also find that there are many potential risks in static hard negative sampling, which is employed by many existing training methods. Therefore, we propose two training strategies named a Stable Training Algorithm for dense Retrieval (STAR) and a query-side training Algorithm for Directly Optimizing Ranking pErformance (ADORE), respectively. STAR improves the stability of DR training process by introducing random negatives. ADORE replaces the widely-adopted static hard negative sampling method with a dynamic one to directly optimize the ranking performance. Experimental results on two publicly available retrieval benchmark datasets show that either strategy gains significant improvements over existing competitive baselines and a combination of them leads to the best performance.

*Corresponding author

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**; *Information retrieval*; *Information retrieval query processing*.

KEYWORDS

Neural ranking, Dense retrieval, Representation learning

1 INTRODUCTION

Document ranking is essential for many IR related tasks, such as question answering [10] and Web search [1, 4]. An effective ranking algorithm will benefit many downstream tasks related to information access researches [11, 24]. Traditional algorithms such as BM25 [26] usually utilize exact term matching signals. Their capabilities are therefore limited to keyword matching and may fail if the query and document use different terms for the same meaning, which is known as the vocabulary mismatch problem. To better deduce the users' search intent and retrieve relevant items, the ranking algorithms are expected to conduct semantic matching between queries and documents [17], which is a challenging problem.

In recent years, with the development of deep learning [6, 20, 28], especially representation learning techniques [2], many researchers have turned to the Dense Retrieval (DR) model to solve the semantic matching problem [10, 15, 18, 22]. In essence, DR attempts to encode queries and documents into low-dimension embeddings to better abstract their semantic meanings. With the learned embeddings, document index can be constructed and the query embedding can be adopted to perform efficient similarity search for online ranking. Previous studies showed that DR models achieve promising results on many IR-related tasks [7, 10, 15].

However, there are some unsolved but essential problems related to DR's effectiveness and training efficiency[1]. Firstly, though

---
[1]We focus on the training efficiency because the efficiency in the inference process is guaranteed by the maximum inner product search algorithms[14, 27].

previous works achieved promising results using different training strategies, their empirical conclusions are sometimes different and even conflict with each other. For example, researchers have conflicting opinions on whether training with hard negatives outperforms that with random negatives [7, 12, 22, 29]. Secondly, many existing well-performed training methods are relatively inefficient and not so applicable for practical usage. For example, some previous works utilize computationally expensive methods, such as knowledge distillation [18] and periodic index refresh [10, 29] in the training process.

We believe that these problems are mainly caused by the lack of theoretical understanding of the training process for DR models. A fair comparison among the existing training strategies will help us design more effective and efficient optimization methods. Therefore, we theoretically formulate the DR training methods and investigate the relationship between training methods and the target optimization objectives.

Through the analysis, we find that random negative sampling aims to minimize the total pairwise errors. Thus, it has a critical problem that some difficult queries can easily dominate its training process and result in serious top-ranking performance loss. On the contrary, hard negative sampling minimizes the top-K pairwise errors. Therefore, it is more suitable for improving the ranking performance of top results, which is also the target of many popular IR systems, such as Web search engines.

Furthermore, we look into two kinds of hard negatives, namely static and dynamic ones. The static ones are employed by many existing training methods. These methods adopt a traditional retriever [8, 15, 22] or a warm-up DR model [10, 29] to pre-retrieve the top results as unchanging hard negatives during training. Theoretical analysis shows that their loss functions cannot guarantee performance improvement, and experiments show their training process is unstable. The dynamic ones are the real hard negatives and rely on the current model parameters. During training, they are dynamically changing because the model is updated iteratively. We show that they resolve the problems of static ones and thus are better for training DR models.

To improve the existing training strategies and help DR models gain better performance, we propose two methods to effectively train the DR models. Firstly, we propose a **S**table **T**raining **A**lgorithm for dense **R**etrieval (STAR) to stabilize the static hard negative sampling method. It utilizes static hard negatives to optimize the top-ranking performance and random negatives to stabilize the training. It reuses the document embeddings in the same batch to improve efficiency. Experimental results show that STAR outperforms competitive baselines with significant efficiency improvement. Secondly, we propose a method to train DR models with dynamic hard negative samples instead of static ones, named a query-side training **A**lgorithm for **D**irectly **O**ptimizing **R**anking p**E**rformance (ADORE). It adopts the document encoder trained by other methods and further trains the query encoder to directly optimize IR metrics. Therefore, it can be used to improve other methods by training a better query encoder. Experimental results show that ADORE significantly boosts the ranking performance. With the help of ADORE, some simple training methods can even match the existing competitive approaches in terms of effectiveness

and greatly outperform them in terms of efficiency. The combination of ADORE and STAR achieves the best performance and is still much more efficient than current popular training methods [10, 29]. We further investigate using index compression techniques [13] to greatly save computational resources for ADORE.

In summary, our contributions are as follows:

- We theoretically investigate the training process of DR models and compare different popular training strategies, including random sampling and hard negative sampling. Our analysis reveals that these strategies lead to different optimization targets and that hard negative sampling better optimizes the top-ranking performance. Experimental results verify our theoretical analysis results.
- We investigate one of the most popular training strategies for DR models that employ static hard negative samples. We theoretically and empirically demonstrate its potential risks in decreasing ranking performance.
- We propose two training strategies that employ hard negatives to optimize DR models. Experimental results on two popular retrieval benchmark datasets show that they achieve significant performance improvements. Their combination achieves the best retrieval performance[2].

2 RELATED WORK

Dense Retrieval represents queries and documents with embeddings. During the offline stage, it encodes documents and builds the document index. During the online stage, it encodes the input queries and performs similarity search [14]. Researchers mainly use negative sampling methods to train DR models except the recently proposed knowledge distillation method [18]. We introduce them in the following.

Several works utilized random negative sampling for training DR models. Huang et al. [12] used random negative samples to approximate the recall task. Karpukhin et al. [15] and Zhan et al. [33] adopted In-Batch training to use other queries' relevant documents in the same mini-batch as negatives, which we believe is approximately random negative sampling. Ding et al. [7] found it is beneficial to increase the number of random negatives in the mini-batch.

Some works applied hard-negative mining to train DR models. Gao et al. [8] and Karpukhin et al. [15] used BM25 top documents as hard negatives. Xiong et al. [29] and Guu et al. [10] used a warm-up DR model to pre-retrieve the top documents as hard negatives during training. They also periodically re-build the index and refresh the hard negatives, which greatly increases the computational cost. Since the hard negatives are fixed during the entire training process or for a few thousand training steps, we refer to these methods as static hard negative sampling.

However, some researchers found static hard negative sampling does not help or even harms the ranking performance. Luan et al. [22] found it brings no benefits for the open-domain QA task. Ding et al. [7] even found it significantly worsens the ranking performance.

---

[2]Code, trained models, and retrieval results are available at https://github.com/jingtaozhan/DRhard.

Although Xiong et al. [29] tried to theoretically analyze the training process of DR models based on convergence speed, we find their conclusions contradict the experimental results and will discuss it in Section 8.1. On the contrary, we present a more fundamental analysis based on optimization objectives, and the conclusions are well supported by the experiments.

3 TASK FORMULATION

Given a query q and a corpus C, the DR model with parameters θ is to find the relevant documents D^+. It encodes queries and documents separately into embeddings, denoted as $\vec{X}_{q;\theta}$ and $\vec{X}_{d;\theta}$, respectively. Then it uses the similarity function, often inner product, to perform efficient retrieval. Let $f(q, d)$ be the predicted relevance score. It equals to:

$$f(q, d) = \langle \vec{X}_{q;\theta}, \vec{X}_{d;\theta} \rangle \tag{1}$$

where $\langle , \rangle$ denotes the similarity function.

DR models are typically trained with pairwise loss where each training sample consists of a query, a negative document d^-, and a positive document d^+ [7, 29, 33]. For ease of explanation, we use the following pairwise loss function:

$$\mathcal{L}(d^+, d^-) = \mathbb{1}_{f(q,d^+) < f(q,d^-)} \tag{2}$$

where $\mathbb{1}_A$ is an indicator function, which is 1 if A holds and 0 otherwise. Therefore, we can establish the relationship between the ranking position of the positive document $\pi(d^+)$ and the training loss with respect to it as follows:

$$\pi(d^+) = \delta(d^+) + 1 + \sum_{d^- \in D^-} \mathcal{L}(d^+, d^-) \tag{3}$$

where $\delta(d^+)$ is the number of relevant documents ranked higher than d^+ and D^- is all the irrelevant documents, i.e., $C \backslash D^+$.

In practice, we cannot directly optimize over all the samples in a corpus since the cost is prohibitive. Therefore, an important question is what distribution should the negative documents be sampled from. Given a query q, different sampling strategies can be viewed as setting different weights $w(d^-)$ for each negative document d^-. Therefore, the general form of the learning objective is as follows:

$$\theta^* = \arg\min_\theta \sum_q \sum_{d^+ \in D^+} \sum_{d^- \in D^-} w(d^-) \cdot \mathcal{L}(d^+, d^-) \tag{4}$$

4 RANDOM VS. HARD NEGATIVES

This section provides a theoretical explanation of how hard negatives help optimize retrieval models.

4.1 Random Negative Sampling

We start with introducing the widely-used random negative sampling method [7, 12, 29]. It uniformly samples negatives from the entire corpus. Using the formulation in Section 3, we can see that

using random negatives is equivalent to minimizing $\pi(d^+)$ (or minimizing the total pairwise errors):

$$
\begin{aligned}
\theta^* &= \arg\min_\theta \sum_q \sum_{d^+ \in D^+} \sum_{d^- \in D^-} \mathcal{L}(d^+, d^-) \\
&= \arg\min_\theta \sum_q \sum_{d^+ \in D^+} \pi(d^+) - \delta(d^+) - 1 \\
&= \arg\min_\theta \sum_q \mathrm{const} + \sum_{d^+ \in D^+} \pi(d^+)
\end{aligned}
\tag{5}
$$

While minimizing the total pairwise errors seems reasonable, we argue that the above optimization objective has the following critical problem. The loss function with respect to a single query q is *unbounded*. $\pi(d^+)$ could be as large as the size of the whole corpus $|C|$. Therefore, the overall loss will be dominated by the queries with large $\pi(d^+)$, and the model can hardly focus on improving top-ranking performance. Section 8.1 will show that this problem leads to serious performance loss in practice.

Therefore, random negative sampling is sub-optimal for training DR models and it is necessary to investigate how to optimize the retrieval performance of DR models with other sampling strategies.

4.2 Hard Negative Sampling

Another sampling strategy is to sample top-K documents as negatives. This paper refers to them as hard negatives. The optimization process is formulated as follows.

$$
\begin{aligned}
\theta^* &= \arg\min_\theta \sum_q \sum_{d^+ \in D^+} \sum_{d^- \in D^-} \mathbb{1}_{\pi(d^-) \le K_q} \cdot \mathcal{L}(d^+, d^-) \\
&= \arg\min_\theta \sum_q \sum_{d^+ \in D^+} \min(\pi(d^+) - \delta(d^+) - 1, K)
\end{aligned}
\tag{6}
$$

where K is the number of hard negative documents, and K_q is the ranking position of the top K_{th} negative document in D^-, i.e., $\sum_{d^- \in D^-} \mathbb{1}_{\pi(d^-) \le K_q} = K$.

Comparing Eq. (6) with Eq. (5), we can see that hard negative sampling minimizes the top-K pairwise errors instead of the total pairwise errors and the loss w.r.t. a single query is bounded by K. In this sense, the adoption of hard negatives alleviates the unbounded loss problem of random negative sampling and leads to a more robust optimization in retrieval performance. Hard negative sampling emphasizes the top-ranking performance and disregards the lower-ranked pairs that hardly affect the user experience or evaluation metrics. Therefore, it is more in line with the truncated evaluation metrics.

The following theorem shows the relationship between Eq. (5) and (6) in terms of whether and when two sampling strategies lead to the same optimal parameter weights.

THEOREM 4.1. *Let θ_h^* be the optimal parameters for Eq. (6). If $\forall d^+, \pi(d^+) - \delta(d^+) - 1 \le K$, then θ_h^* is also the optimal parameters for Eq. (5).*

That is to say, if DR models can rank all relevant documents at high positions, using random negative sampling and hard negative sampling in training DR models will result in the same optimal parameters. However, DR models may not be effective for some training queries, especially those that require keyword match [8, 18, 33]. In this case, the random negative sampling strategy would

pay too much attention to the difficult queries, which, however, would not be reflected by the popular adopted truncated evaluation metrics.

5 STATIC VS. DYNAMIC HARD NEGATIVES

In Section 4, we have shown that hard negative sampling is more effective than random negative sampling. Since the real hard negatives are dynamically changing during training, we refer to them as dynamic hard negatives. However, the current main strategy is using a traditional retriever [8, 15, 22] or a warm-up DR model [10, 29] to pre-retrieve the top documents as fixed hard negatives during training. Thus, we refer to them as static hard negatives[3]. Next, we will theoretically discuss these two kinds of hard negatives.

5.1 Static Hard Negatives

We formulate static hard negative sampling as follows.

$$\theta^* = \arg\min_\theta \sum_q \sum_{d^+ \in D^+} \sum_{d^- \in D^-} \mathbb{1}_{d^- \in D_s^-} \cdot \mathcal{L}(d^+, d^-) \quad (7)$$

where D_s^- is a set of pre-retrieved hard negatives ($|D_s^-| \ll |C|$). To characterize D_s^-, we define its 'quality' $\phi(D_s^-)$ as the highest ranking position:

$$\phi(D_s^-) = \min_{d^- \in D_s^-} \pi(d^-) \in [1, |C| - |D_s^-| + 1] \quad (8)$$

We can see that $\phi(D_s^-)$ is very loosely bounded and theoretically can be very large. Though previous works [8, 10, 15, 22, 29] implicitly assumed that $\phi(D_s^-)$ is always very small during the entire training process, we argue that is not necessarily true. The training process is likely to drive $\phi(D_s^-)$ large instead of small because the gradient-based optimization constantly forces the DR model to predict small relevance scores for D_s^-. Therefore, D_s^- is likely to be ranked lower and lower during training, which, however, is invisible to the loss. Section 8.2 will verify that the DR model quickly ranks D_s^- very low.

Using the above notions, we discuss the ranking performance of an 'ideal' DR model whose training loss is minimum zero. We use mean reciprocal rank (MRR) to evaluate the ranking performance. The infimum of MRR is as follows:

$$\sum_q \sum_{d^+ \in D^+} \sum_{d^- \in D^-} \mathbb{1}_{d^- \in D_s^-} \cdot \mathcal{L}(d^+, d^-) = 0$$

$$\Rightarrow \inf(\text{MRR}) = E_q \frac{1}{\phi(D_s^-) - |D^+|} \geq \frac{1}{|C| - |D_s^-| + 1 - |D^+|} \approx \frac{1}{|C|} \quad (9)$$

where inf denotes the infimum.

Since $\phi(D_s^-)$ can be very large as discussed above, the infimum can be almost zero and thus the top-ranking performance is not theoretically guaranteed. In the worst case, MRR is approximately $\frac{1}{|C|}$. Considering that we are discussing an 'ideal' DR model whose training loss is zero, this result is unacceptable. Therefore, we believe it is risky to use static hard negative sampling.

---

[3]Some works [10, 29] periodically refresh the index and retrieve static hard negatives during training. Though we do not consider it in the theoretical analysis, we will explore it in our experiments and show its limited capability to resolve the problems analyzed in this section.

(a) Input: one relevant doc and multiple static hard negatives are sampled for each query.

(b) Reusing other document embeddings when computing pairwise loss.

Figure 1: Reusing strategy of STAR. The number of rows correspond to the batch size.

5.2 Dynamic Hard Negatives

This section will show the benefits of dynamic hard negatives compared with static ones. The dynamic hard negatives are actually the top-ranked irrelevant documents given the DR parameters at any training step. For ease of comparison, we formally define the dynamic hard negatives D_θ^- based on Eq. (6):

$$D_\theta^- = \{d^- : \mathbb{1}_{\pi(d^-) \leq K_q} = 1\} \quad (10)$$

During training, D_θ^- is dynamically changing because θ is constantly updated. In fact, the theoretical analysis in Section 4.2 is based on dynamic hard negatives and we have shown their benefits compared with random negatives.

Since we use $\phi(D_s^-)$ to represent the 'quality' of static hard negatives, we show the 'quality' of D_θ^- for comparison. Considering that D_θ^- always contains the top-ranked negatives, it is well bounded:

$$\phi(D_\theta^-) = \min_{d^- \in D_\theta^-} \pi(d^-) \in [1, |D^+| + 1] \quad (11)$$

Therefore, according to Eq. (9), MRR achieves the maximum one if the training loss achieves the minimum zero:

$$\sum_q \sum_{d^+ \in D^+} \sum_{d^- \in D^-} \mathbb{1}_{\pi(d^-) \leq K_q} \cdot \mathcal{L}(d^+, d^-) = 0$$

$$\Rightarrow \quad \phi(D_\theta^-) = |D^+| + 1 \quad \Rightarrow \quad \text{MRR} = 1 \quad (12)$$

In this sense, dynamic hard negative sampling is better than the static one.

6 TRAINING ALGORITHMS

In Section 4 and 5, we show that random negative sampling and static hard negative sampling have problems in terms of effectiveness and training stability, respectively. Thus, we propose two improved training strategies for better training DR models, namely STAR and ADORE.

6.1 STAR

According to our analysis in Section 5, static hard negative sampling is unstable. Therefore, we propose a **S**table **T**raining **A**lgorithm for dense **R**etrieval (STAR). STAR aims to employ static hard negatives to improve top-ranking performance and random negatives to stabilize the training process. Moreover, it reuses the document embeddings in the same batch to greatly improve efficiency.

6.1.1 Employing Static Hard Negatives. STAR uses a warm-up model to retrieve the top documents for all training queries, which serve as the static hard negatives and will not be updated. We use them to approximate the real dynamic hard negatives. The inspiration comes from our pilot experiment where we find different well-trained DR models tend to recall the same set of documents but with different ranking order.

6.1.2 Employing Random Negatives. To stabilize the training process, STAR additionally introduces random negatives during training and optimizes the following objective:

$$\theta^* = \alpha \cdot L_r(\theta) + L_s(\theta) \quad (0 < \alpha \ll 1) \tag{13}$$

where $L_r(\theta)$ is the loss of random negative sampling defined as Eq. (5), $L_s(\theta)$ is the loss of static hard negative sampling defined as Eq. (7), and α is a hyper-parameter. If static hard negatives well approximate the dynamic ones, $L_s(\theta)$ tends to be large and dominates the training process. If not, $L_s(\theta)$ tends to be small and STAR approximately degenerates into a random negative sampling method, which is much better than the worst case of static hard negative sampling.

6.1.3 Improving Efficiency. STAR adopts a reusing strategy to improve efficiency. It does not explicitly add random negative documents to the input and instead reuse other documents in the same batch as approximately random negatives. Figure 1a shows one input batch of STAR, which is the same as static hard negative sampling method [29]. Each row has one query, its relevant document, and its static hard negative documents. Figure 1b shows how STAR computes pairwise loss for one query. The document embeddings in other rows are also utilized as negatives, which we believe can approximate the random negatives. The reusing strategy could also be regarded as an improved version of In-Batch training strategy [15, 33].

6.1.4 Loss Function. STAR adopts RankNet pairwise loss [3]. Given a query q, let d^+ and d^- be a relevant document and a negative document. $f(q, d)$ is the relevance score predicted by the DR model. The loss function $\mathcal{L}_R$ is formulated as follows:

$$\mathcal{L}(d^+, d^-) = \log(1 + e^{f(q,d^-) - f(q,d^+)}) \tag{14}$$

6.2 ADORE

Our previous analysis in Section 4 and 5 shows great potential of utilizing dynamic hard negatives for training DR models. Therefore, this section presents a query-side training **A**lgorithm for **D**irectly **O**ptimizing **R**anking p**E**rformance (ADORE). It utilizes dynamic hard negatives and LambdaLoss [3] to directly optimize ranking performance. It adopts a pre-trained document encoder and trains the query encoder. We show the training process of ADORE in Figure 2b and the common negative sampling method in Figure 2a for comparison.

6.2.1 Employing Dynamic Hard Negatives. ADORE resolves the difficulty to acquire dynamic hard negatives as follows. Before training, ADORE pre-computes the document embeddings with a pre-trained document encoder and builds the document index. They are fixed throughout the entire training process. At each training iteration, it encodes a batch of queries and uses the embeddings to retrieve

(a) Negative sampling method.   **(b) ADORE.**

Figure 2: The flow chart of negative sampling training method and our proposed ADORE. Batch size is set to one.

the corresponding top documents, which are the real dynamic hard negatives. ADORE utilizes them to train the DR model. To the best of our knowledge, ADORE is the first DR training method to employ dynamic hard negatives. Its effectiveness is guaranteed by the theoretical analysis in Section 4.2 and 5.2.

6.2.2 Directly Optimizing IR Metrics. Unlike previous methods, ADORE performs retrieval at each step and thus can utilize the listwise approach [19] to directly optimize the ranking performance. We use LambdaLoss [3] to derive a weight $w(d^-)$ for each training pair to better optimize IR metrics. Given a query q, a positive document d^+, and a negative document d^-, let $\Delta \mathcal{M}$ be the size of the change in target IR metric given by swapping the ranking positions of d^+ and d^-. Through multiplying $\Delta \mathcal{M}$ to RankNet loss, Burges [3] empirically showed that it could directly optimize IR metrics. Therefore, ADORE employs the following loss functions:

$$\mathcal{L}(d^+, d^-) = \Delta \mathcal{M} \cdot \log(1 + e^{f(q,d^-) - f(q,d^+)}) \tag{15}$$

According to Qin et al. [25], when the training set is large enough, this method will yield the optimal model in terms of the corresponding metric.

6.2.3 End-to-end Training. Document index is often compressed in practice to save computational resources. Different compression techniques may lead to different optimal DR parameters and hence should be considered during training. Though previous methods ignore this information, ADORE can well perform end-to-end training with the actual document index used in inference as shown in Figure 2b. In this regard, ADORE alleviates the discrepancy between training and inference and can achieve better ranking performance. We will empirically verify this in section 8.3.3.

6.3 Combining STAR and ADORE

Both ADORE and STAR have their own advantages. ADORE directly optimizes the ranking performance while STAR cannot. STAR optimizes both the query encoder and the document encoder while ADORE only optimizes the query encoder. We combine the two

strategies by using STAR to train the document encoder and using ADORE to further train the query encoder.

7 EXPERIMENTAL SETTINGS

7.1 Datasets

We conduct experiments on the TREC 2019 Deep Learning (DL) Track [4]. The Track focuses on ad-hoc retrieval and consists of the passage retrieval and document retrieval tasks. The passage retrieval task has a corpus of 8.8 million passages with 503 thousand training queries, 7 thousand development queries, and 43 test queries. The document retrieval task has a corpus of 3.2 million documents with 367 thousand training queries, 5 thousand development queries, and 43 test queries. We use the official metrics to evaluate the top-ranking performance, such as MRR@10 and NDCG@10. Besides, R@100 is adopted to evaluate the recall power.

7.2 Baselines

7.2.1 Sparse Retrieval. We list several representative results according to the TREC overview paper [4] and runs on MS MARCO [1] leaderboard, such as BM25 [31], the best traditional retrieval method, BERT weighted BM25 (DeepCT) [5].

7.2.2 Dense Retrieval. The DR baselines include several popular training methods. For random negative sampling baselines, we present Rand Neg [12] and In-Batch Neg [15, 33]. The former randomly samples negatives from the entire corpus, and the latter uses other queries' relevant documents in the same batch as negative documents. For static hard negative sampling baselines, we present BM25 Neg [8] and ANCE [29]. BM25 Neg uses the BM25 top candidates as the negative documents. ANCE uses a warm-up model to retrieve static hard negatives. Every $10k$ training steps, it encodes the entire corpus, rebuilds the document index, and refreshes the static hard negatives with the current DR model parameters. For knowledge distillation baseline, we present TCT-ColBERT [18] which uses ColBERT [16] as the teacher model.

7.2.3 Cascade IR. Although this paper focuses on the retrievers, we employ cascade systems for further comparison. We report the performances of the best LeToR model and the BERT model [23], which use BM25 as the first-stage retriever.

7.3 Implementation Details

All DR models use the RoBERTa$_{base}$ [20] model as the encoder. The output embedding of the "[CLS]" token is used as the representation of the input text. We use the inner product to compute the relevance score and adopt the Faiss library [14] to perform the efficient similarity search. Documents are truncated to a maximum of 120 tokens and 512 tokens for the passage and document tasks, respectively. The top-200 documents are used as the hard negatives.

The implementation details for DR baselines are as follows. In-Batch Neg and Rand Neg models are trained on passage task with Lamb optimizer [32], batch size of 256, and learning rate of 2×10^{-4}. We find LambdaLoss [3] cannot bring additional performance gains and hence use the RankNet loss. The trained models are directly evaluated on the document task because this produces better retrieval performance [29, 30]. We use the open-sourced BM25 Neg model and ANCE model [29]. They are re-evaluated so

Figure 3: Distribution of pairwise errors per query on MARCO Dev Passage dataset for Rand Neg model. The histogram shows the proportion of total queries, and the line chart shows the proportion of total pairwise errors.

we can perform the significance test. Minor performance variances are observed compared with the originally reported values. Since TCT-ColBERT [18] is not open-sourced, we borrow the results from its original paper and do not perform the significance test on it.

STAR uses the BM25 Neg model as the warm-up model, which is the same as ANCE and hence their results are directly comparable. It uses Lamb optimizer, batch size of 256, and learning rate of 1×10^{-4} on passage task. It uses AdamW optimizer [21], batch size of 60, and learning rate of 2×10^{-6} on document task.

ADORE uses AdamW optimizer, learning rate of 5×10^{-6}, and batch size of 32 on both tasks. $\mathcal{M}$ in Eq. (15) is MRR@200 and MRR@10 on passage and document tasks, respectively. ADORE can improve a trained DR model by further training its query encoder. For example, ADORE+Rand Neg means it uses the document encoder trained by Rand Neg and further trains the query encoder.

8 EXPERIMENTAL RESULTS

We conduct experiments to verify our theoretical analysis and the effectiveness of our proposed methods. Specifically, this section studies the following research questions:

- **RQ1:** How do random negatives and hard negatives affect optimization objectives?
- **RQ2:** How does the static hard negative sampling method perform in practice?
- **RQ3:** How effective and efficient are our proposed training methods?

8.1 Random vs. Hard Negatives

This section compares random negative sampling and hard negative sampling to answer **RQ1**.

In Section 4, we theoretically show that random negative sampling faces a critical problem that some difficult queries may dominate the training process. To verify whether this phenomenon really exists, we plot the distribution of pairwise errors per query in Figure 3 by evaluating the trained Rand Neg model on MARCO Dev Passage dataset. The figure shows that 0.2% difficult queries (pairwise errors $\geq 10^5$) contribute surprisingly 60% of the total pairwise errors. Therefore, the problem is very serious in practice.

To investigate whether hard negative sampling alleviates the above problem, we evaluate the total pairwise errors and top-K (200)

Table 1: Total pairwise errors and top-K pairwise errors of different DR models. Best results are marked bold. K equals to 200.

Models	Total Errors	Top-K Errors
In-Batch Neg	679	43.2
Rand Neg	**659**	39.3
BM25 Neg	2432	46.4
ANCE	1448	37.3
STAR	1128	**35.8**
ADORE+In-Batch Neg	**649**	37.7
ADORE+Rand Neg	736	36.8
ADORE+BM25 Neg	1840	40.0
ADORE+ANCE	1345	36.5
ADORE+STAR	1037	**34.4**

(a) Overlap with the real dynamic hard negatives. (b) The MRR@10 at each step.

Figure 4: The performance of static hard negative sampling on MARCO Train Doc dataset. X-axes is the training steps in thousands. Dynamic/static/random separately denote dynamic hard/static hard/random negative sampling.

pairwise errors for different models and show the results in Table 1. As we expect, random negative sampling methods, namely Rand Neg and In-Batch Neg, well minimize the total pairwise errors but cannot effectively minimize the top-K pairwise errors. The hard negative sampling method, STAR, well minimizes the top-K pairwise errors. The static hard negative sampling methods, namely BM25 Neg and ANCE, achieve compromised top-ranking performance. ADORE effectively improves each model's top-ranking performance and performs best using the document encoder trained by STAR. Therefore, the results convincingly show that hard negative sampling can better optimize top-ranking performance.

Note that our experiments contradict previous theoretical analyses. Xiong et al. [29] argued that random and hard negative sampling share the same optimization objective and that the latter's advantage lies in quicker convergence. However, Table 1 shows that random negative sampling well converges and outperforms hard negative sampling in terms of total pairwise errors. Figure 4b, which will be introduced in Section 8.2, shows that random negative sampling decreases the top-ranking performance using a relatively good initialization. Therefore, random and hard negative sampling optimizes different targets, and the benefits of hard negatives are not about convergence speed.

Figure 5: The ablation study for ADORE. ADORE\L uses RankNet loss instead of LambdaLoss. Results are MRR@10 values on MARCO Dev Passage dataset.

Figure 6: The t-SNE plot of query and document representations for ADORE. The QID is 1129237 and is from TREC DL Doc test set.

8.2 Static vs. Dynamic Hard Negatives

This section compares static hard negatives and dynamic hard negatives to answer **RQ2**. In Section 5.1, we argue that the DR model may quickly rank the static hard negatives very low and thus they cannot approximate the dynamic ones. We also analyze that static hard negative sampling does not necessarily optimize ranking performance. This section aims to verify them.

Our experimental settings are as follows. Compared with our theoretical analysis, we further explore the popular periodic index refresh approach by iteratively retrieving static hard negatives every 5k training steps [10, 29]. The DR model is initialized with a trained BM25 Neg model. We fix the document embeddings and retrieve documents for training queries at each step to acquire the dynamic hard negatives and evaluate the ranking performance. Figure 4a shows the overlap between static hard negatives and the real dynamic hard negatives. Figure 4b presents the ranking performance, which also uses random negative sampling and dynamic hard negative sampling for comparison.

According to Figure 4b, the static hard negatives are quickly ranked very low by the DR model and can hardly approximate the dynamic ones as we expect. Moreover, the overlap is always less than 70% because the static hard negatives cannot consider the random noise introduced by dropout during training. In this regard, the dynamic hard negatives cannot be entirely replaced by static ones.

Figure 4b also supports our analysis by showing the unstableness of static hard negative sampling. The ranking performance fluctuates wildly and periodically. In most time, it significantly underperforms random negative sampling. On the contrary, dynamic hard negative sampling steadily improves the ranking performance.

Table 2: Results on TREC 2019 Deep Learning Track. We perform the significance test on DR models except for TCT-ColBERT. We use paired t-test with p-value threshold of 0.01 on dev dataset and 0.05 on TREC test dataset. *indicate significant improvements over In-Batch Neg, Rand Neg and BM25 Neg. †indicate significant improvements over ANCE. ‡indicate significant improvements over ADORE (In-Batch Neg), ADORE (Rand Neg) and ADORE (BM25 Neg). Best results of DR models are marked bold. Results not available or not applicable are marked as 'n/a'.

Models	MARCO Dev Passage		TREC DL Passage		MARCO Dev Doc		TREC DL Doc	
	MRR@10	R@100	NDCG@10	R@100	MRR@100	R@100	NDCG@10	R@100
Cascade IR								
Best TREC Trad LeToR	n/a	n/a	0.556	n/a	n/a	n/a	0.561	n/a
BERT Reranker [23]	0.365	n/a	0.742	n/a	0.413	n/a	0.646	n/a
Sparse Retrieval								
BM25 [31]	0.187	0.670	0.506	0.453	0.279	0.807	0.519	0.395
Best TREC Trad Retrieval	n/a	n/a	0.554	n/a	n/a	n/a	0.549	n/a
DeepCT [5]	0.243	0.760	n/a	n/a	n/a	n/a	0.554	n/a
DR: Distillation								
TCT-ColBERT [18]	0.335	n/a	0.670	n/a	n/a	n/a	n/a	n/a
DR: Negative Sampling								
In-Batch Neg [9]	0.264	0.837	0.583	0.463	0.320	0.864	0.544	0.295
Rand Neg [12]	0.301	0.853	0.612	0.464	0.330	0.859	0.572	0.284
BM25 Neg [8]	0.309	0.813	0.607	0.362	0.316	0.794	0.539	0.223
ANCE [29]	$0.338^{*\ddagger}$	0.862^{*}	0.654	0.445	$0.377^{*\ddagger}$	0.894^{*}	0.610	0.273
DR: Ours								
STAR	$0.340^{*\ddagger}$	0.867^{*}	0.642	$0.467^{\dagger}$	$0.390^{*\dagger\ddagger}$	$0.913^{*\dagger\ddagger}$	0.605	0.313
ADORE+In-Batch Neg	0.316	0.860	0.658^{*}	$0.471^{\dagger}$	0.362^{*}	0.884^{*}	0.580	$0.315^{\dagger}$
ADORE+Rand Neg	0.326^{*}	0.865^{*}	0.661^{*}	$0.472^{\dagger}$	0.361^{*}	0.885^{*}	0.585	$0.298^{\dagger}$
ADORE+BM25 Neg	0.329^{*}	0.846	0.661	0.431	0.352^{*}	0.872	0.610	0.293
ADORE+ANCE	$0.341^{*\ddagger}$	0.866^{*}	$0.675^{*\dagger}$	$0.454^{\dagger}$	$0.390^{*\dagger\ddagger}$	$0.902^{*\dagger\ddagger}$	$\mathbf{0.634}^{*\dagger}$	0.292
ADORE+STAR	$\mathbf{0.347}^{*\dagger\ddagger}$	$\mathbf{0.876}^{*\dagger\ddagger}$	$\mathbf{0.683}^{*}$	$\mathbf{0.473}^{\dagger}$	$\mathbf{0.405}^{*\dagger\ddagger}$	$\mathbf{0.919}^{*\dagger\ddagger}$	0.628^{*}	$\mathbf{0.317}^{\dagger}$

Though carefully tuning the hyper parameters may alleviate the above problems and several works achieved promising results using this method [10, 29], the next two sections will show that our proposed methods can better optimize the ranking performance with great efficiency gain.

8.3 Effectiveness

This section investigates the effectiveness of our proposed STAR and ADORE to answer **RQ3**. We conduct experiments on passage retrieval and document retrieval tasks and show the ranking performance in Table 2. We discuss the results[4] in the following.

8.3.1 Baselines. Random negative sampling can effectively train DR models compared with sparse retrieval and the LeToR methods. Rand Neg outperforms BM25, DeepCT, and LeToR even by a large margin on some metrics. Static hard negative sampling does not necessarily lead to performance improvements compared with random negative sampling. It improves the top-ranking performance but may harm the recall capability. Specifically, BM25 Neg model almost underperforms Rand Neg in terms of every metric. ANCE outperforms Rand Neg on MRR@10 but underperforms it in terms of R@100 on the test sets.

8.3.2 STAR. STAR significantly outperforms random negative sampling, especially on top-ranking performance. For example, it outperforms Rand Neg by 13% and 18% on dev passage and dev document sets in terms of MRR@10 and MRR@100, respectively. The results are consistent with our original expectation to improve top-ranking performance.

STAR also significantly outperforms the static hard negative sampling method, especially on recall capability. For example, it outperforms BM25 Neg and ANCE by 40% and 15% on TREC DL Doc set in terms of R@100, respectively. Such achievement is very meaningful considering that ANCE iteratively re-builds the index and then updates the static hard negatives while STAR only retrieves once. It demonstrates that STAR better optimizes ranking performance through stabilizing the training process.

STAR outperforms the knowledge distillation approach on the large dev set but underperforms it on the small testing set. Therefore, the necessity of knowledge distillation remains further explored since it is usually more computationally expensive than directly training the models.

8.3.3 ADORE. ADORE greatly improves all DR models' performance by further training the query encoders. For example, ADORE improves In-Batch Neg's top-ranking performance by 20% and 22% separately on dev passage and dev doc datasets. It also improves BM25 Neg's recall performance by 19% and 31% separately on

---

[4]Although DR models significantly underperform BM25 on TREC DL Doc in terms of R@100, it may be caused by many unlabeled relevant documents [29].

Table 3: ADORE+BM25 Neg's ranking performance when using different document indexes for training and evaluation. Results are MRR@10 values on MARCO Dev Passage. Row and column names are PQ [13] values, which denote the number of compressed subvectors. A smaller PQ value corresponds to more compression. '-' refers to no compression.

Train \ Test	24	48	96	-
24	**0.247**	0.280	0.282	0.324
48	0.244	**0.283**	0.285	0.327
96	0.243	0.278	**0.291**	0.326
-	0.243	0.278	0.288	**0.329**

testing passage and testing doc sets. The results convincingly show the effectiveness of ADORE.

The combination of ADORE and STAR achieves the best performance. ADORE+STAR greatly outperforms all baselines, especially the existing competitive knowledge distillation approach (TCT-ColBERT) and the periodic index refreshing approach (ANCE). Furthermore, it nearly matches BM25-BERT two-stage retrieval system on the document retrieval task.

We conduct an ablation study for ADORE to investigate the contribution of dynamic hard negatives and the LambdaLoss. The results are shown in Figure 5. It demonstrates that using dynamic hard negatives greatly improves the ranking performance, which verifies our previous theoretical analysis. The LambdaLoss can further boost the ranking performance for models like Rand Neg but cannot bring further improvement for STAR. A possible reason is that STAR already emphasizes top-ranking performance compared with methods like Rand Neg.

To illustrate how ADORE improves ranking performance, we plot a t-SNE example in Figure 6 using a query from TREC DL Doc set. ADORE uses the document encoder trained by BM25 Neg and further trains the query encoder. After training, ADORE maps the query closer to the relevant documents and thus improves the retrieval performance.

Section 6.2.3 argues that the optimal DR parameters may be different for different compressed indexes and thus ADORE can achieve better performance through end-to-end training. To investigate whether this is true, we use different compressed indexes to train and evaluate DR models. The results are shown in Table 3. We can see that end-to-end training better optimizes the ranking performance for different compression techniques. Thus, ADORE is suitable to improve the performance of compressed indexes.

8.4 Training Efficiency

This section presents the training efficiency of our proposed methods to answer **RQ3** from two aspects, namely training time and computational resources. Since ANCE is competitive in terms of effectiveness, we use it as our efficiency baseline.

8.4.1 Training Time. We test the training speed with 11GB GeForce RTX 2080 Ti GPUs and show the results in Table 4. It demonstrates the significant efficiency gains of our proposed methods compared with ANCE. The improvement comes from two aspects. Firstly, our

Table 4: Training efficiency comparison on MARCO Dev passage dataset. All methods use BM25 Neg model as the warm-up model. The training hours and speedup are separately rounded to integers.

Models	GPUs	Hours	Speedup
ANCE [29]	4	645	1x
STAR	1	33	20x
ADORE	4	4	179x
ADORE+STAR	4	37	18x

Table 5: ADORE's performance on passage dataset with different compressed indexes. A smaller PQ value corresponds to more compression. Training hours for $PQ=6$ and $PQ=12$ are blank because they are not supported on GPU. The last line shows the performance with uncompressed index.

PQ	Index Quality GB	Index Quality MRR@10	Train Hours	Dev MRR@10	Test NDCG@10
6	0.1	0.050	-	0.304	0.627
12	0.2	0.151	-	0.318	0.635
24	0.2	0.221	3.0	0.324	0.644
48	0.5	0.254	3.2	0.327	0.652
96	0.9	0.273	3.7	0.326	0.656
–	26	0.309	3.6	0.329	0.661

methods converge very fast. For example, on the passage retrieval task, ANCE needs 600k steps with batch size of 64 while ADORE needs 60k steps with batch size of 32. Secondly, to periodically update the static hard negatives, ANCE iteratively encodes the corpus to embeddings and builds temporary document indexes, which takes 10.75 hours each time with three GPUs. In contrast, STAR only builds one temporary index and ADORE does not even have this overhead. Note that although ADORE retrieves documents at each step, the search is very efficient and takes a total of 40 minutes, which is about 20% of the entire training time.

8.4.2 Computational resources. This section shows ADORE can greatly save computational resources with compressed document indexes, which is meaningful because GPU memory is usually limited. The experimental settings are as follows. We use the document embeddings generated by BM25 Neg and utilize product quantization (PQ) [13] to compress the index. Besides memory usage, we also report the search quality for the compressed index, which is measured with MRR@10 on MARCO Dev Passage dataset. The results on shown in Table 5.

We can see that PQ significantly reduces memory consumption and the performance loss is minor. After compression, ADORE is able to run on only one 11 GB GPU compared with four in Table 4. Specifically, $PQ=96$ reduces the GPU memory footprint to 3% and yields little performance loss (0.326 vs. 0.329). A smaller PQ further reduces GPU memory footprint. Therefore, ADORE is applicable in a memory-constrained environment. Note that an over-compressed index ($PQ=6$) achieves poor search quality (MRR@10: 0.05), and

therefore ADORE almost degenerates into random negative sampling. ADORE with PQ=6 achieves similar MRR@10 with Rand Neg (0.304 vs. 0.301).

9 CONCLUSION

This paper investigates how to effectively and efficiently train the DR models. Firstly, we theoretically formalize the training process and compare different training methods. We reveal why hard negative sampling outperforms random negative sampling. Secondly, we investigate the current popular static hard negative sampling method and demonstrate its risks through theoretical analysis and empirical verification. Finally, based on our analysis, we propose two training methods that employ hard negatives to optimize the DR models. Experiments on two widely-adopted retrieval datasets show that they achieve significant performance improvements and efficiency gains compared with other effective methods. Their combination achieves the best retrieval performance.

There are still some remaining issues for future work. Firstly, how to train the document encoder directly based on the retrieval results remains to be explored. Secondly, this paper applies DR to ad-hoc search. Future work may examine the proposed methods in other tasks that require a retrieval module, such as Open Question Answering.

ACKNOWLEDGMENTS

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).
[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.
[3] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.
[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. In *Text REtrieval Conference (TREC)*. TREC.
[5] Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687* (2019).
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.
[7] Yingqi Qu Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daixiang Dong, Hua Wu, and Haifeng Wang. 2020. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. *arXiv preprint arXiv:2010.08191* (2020).
[8] Luyu Gao, Zhuyun Dai, Zhen Fan, and Jamie Callan. 2020. Complementing lexical retrieval with semantic residual embedding. *arXiv preprint arXiv:2004.13969* (2020).
[9] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: a new estimation principle for unnormalized statistical models. In *Proceedings of*

[10] *the 13th International Conference on Artificial Intelligence and Statistics*. 297–304.
[10] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909* (2020).
[11] Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2020. Learning-to-Rank with BERT in TF-Ranking. *arXiv preprint arXiv:2004.08476* (2020).
[12] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based Retrieval in Facebook Search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.
[13] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.
[14] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* (2019).
[15] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).
[16] O. Khattab and M. Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).
[17] Hang Li and Jun Xu. 2014. Semantic matching in search. *Foundations and Trends in Information retrieval* 7, 5 (2014), 343–469.
[18] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *arXiv preprint arXiv:2010.11386* (2020).
[19] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and trends in information retrieval* 3, 3 (2009), 225–331.
[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: a robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[21] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).
[22] Yi Luan, Jacob Eisenstein, Kristina Toutanove, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval. *arXiv preprint arXiv:2005.00181* (2020).
[23] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).
[24] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375* (2019).
[25] Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A general approximation framework for direct optimization of information retrieval measures. *Information retrieval* 13, 4 (2010), 375–397.
[26] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*. Springer, 232–241.
[27] Fumin Shen, Wei Liu, Shaoting Zhang, Yang Yang, and Heng Tao Shen. 2015. Learning binary codes for maximum inner product search. In *Proceedings of the IEEE International Conference on Computer Vision*. 4148–4156.
[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Advances in neural information processing systems*. 5998–6008.
[29] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv preprint arXiv:2007.00808* (2020).
[30] Ming Yan, Chenliang Li, Chen Wu, Bin Bi, Wei Wang, Jiangnan Xia, and Luo Si. 2019. IDST at TREC 2019 Deep Learning Track: Deep Cascade Ranking with Generation-based Document Expansion and Pre-trained Language Modeling.. In *TREC*.
[31] Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible ranking baselines using Lucene. *Journal of Data and Information Quality (JDIQ)* 10, 4 (2018), 1–20.
[32] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962* (2019).
[33] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. RepBERT: Contextualized Text Embeddings for First-Stage Retrieval. *arXiv preprint arXiv:2006.15498* (2020).