Progressively Optimized Bi-Granular Document Representation for Scalable Embedding Based Retrieval

Shitao Xiao♠, Zheng Liu♣, Weihao Han♦, Jianjin Zhang♦, Yingxia Shao♠, Defu Lian♥, Chaozhuo Li♣, Hao Sun♦, Denvy Deng♦, Liangjie Zhang♦, Qi Zhang♦, Xing Xie♣
♠: Beijing University of Posts and Telecommunications, Beijing China
♣: Microsoft Research Asia, Beijing, China
♦: Microsoft STCA, Beijing, China
♥: University of Science and Technology of China, Hefei, China
{zhengliu,weihan,jianjzh,cli,hasun,dedeng,liazha,qizhang,xingx}@microsoft.com
{stxiao,shaoyx}@bupt.edu.cn
liandefu@ustc.edu.cn

ABSTRACT

Ad-hoc search calls for the selection of appropriate answers from a massive-scale corpus. Nowadays, the embedding-based retrieval (EBR) becomes a promising solution, where deep learning based document representation and ANN search techniques are allied to handle this task. However, a major challenge is that the ANN index can be too large to fit into memory, given the considerable size of answer corpus. In this work, we tackle this problem with *Bi-Granular Document Representation*, where the lightweight sparse embeddings are indexed and standby in memory for coarse-grained candidate search, and the heavyweight dense embeddings are hosted in disk for fine-grained post verification. For the best of retrieval accuracy, a *Progressive Optimization* framework is designed. The sparse embeddings are learned ahead for high-quality search of candidates. Conditioned on the candidate distribution induced by the sparse embeddings, the dense embeddings are continuously learned to optimize the discrimination of ground-truth from the shortlisted candidates. Besides, two techniques: the contrastive quantization and the locality-centric sampling are introduced for the learning of sparse and dense embeddings, which substantially contribute to their performances. Thanks to the above features, our method effectively handles massive-scale EBR with strong advantages in accuracy: with up to +4.3% recall gain on million-scale corpus, and up to +17.5% recall gain on billion-scale corpus. Besides, Our method is applied to a major sponsored search platform with substantial gains on revenue (+1.95%), Recall (+1.01%) and CTR (+0.49%). Our code is available at https://github.com/microsoft/BiDR.

KEYWORDS

Bi-Granular Document Representation, Large-Scale Dense Retrieval

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3511957

ACM Reference Format:
Shitao Xiao, Zheng Liu, Weihao Han, Jianjin Zhang, Yingxia Shao, Defu Lian, Chaozhuo Li, Hao Sun, Denvy Deng, Liangjie Zhang, Qi Zhang, Xing Xie. 2022. Progressively Optimized Bi-Granular Document Representation for Scalable Embedding Based Retrieval. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM , 11 pages. https://doi.org/10.1145/3485447.3511957

1 INTRODUCTION

Ad-hoc search is an important component for today's online applications, e.g., recommenders and advertising platforms. In response to each input query, the system needs to select appropriate answers from the entire corpus which may serve user's search demand. Nowadays, the embedding-based retrieval (EBR) has become a promising solution, where deep learning based document representation and ANN search techniques are jointly used to handle this problem. EBR consists of the following steps. In the offline stage, the text encoder is learned to represent the input documents as embeddings (e.g., DPR [20], ANCE [37]), where queries and their correlated answers can be projected close to each other in the latent space. Then, the ANN index (e.g., HNSW [28], IVFADC [18]) is built for the embeddings of the entire corpus. In the online stage, the ANN index is hosted in memory. Once a search request is presented, the input query is encoded into its embedding, with which relevant answers are efficiently selected by searching the ANN index.

1.1 Embedding Based Retrieval At Scale

EBR is challenging when a massive-scale corpus is given, as the ANN index can be too large to fit into memory; e.g., one billion dim-768 floating-point embeddings will take 3TB RAM space, which is orders of magnitude larger than the capability of a physical machine. Such a problem becomes even severe for today's online platforms, where billions of answers are generated by the content providers. One solution of scaling up EBR is to partition the data into multiple shards, each of which is hosted on a different machine. The query will be routed to all the shards, and the search results will be aggregated in the final stage [11]. However, the above method operates at a huge cost. In this work, we tackle this problem with the *Bi-Granular Document Representation* (Figure 1), where answers are represented as two sets of embeddings: the lightweight *sparse*

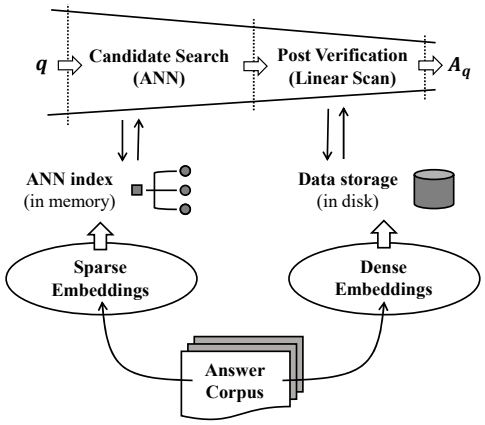

Figure 1: Bi-granular document representation based EBR. The answers are represented as bi-granular embeddings: the sparse embeddings are indexed and standby in memory for candidate search; the dense embeddings are hosted in disk and used for post verification.

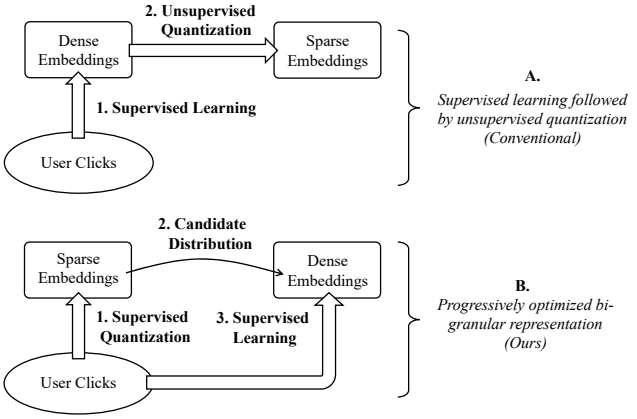

Figure 2: Comparison of bi-granular document representations. (A) The conventional method learns dense embeddings first and run unsupervised quantization for the sparse embeddings. (B) Our method learns the sparse embeddings first via supervision, based on which the dense embeddings are progressively optimized.

embeddings, which are indexed and standby in memory for coarse-grained candidate search; the heavyweight *dense embeddings*, which are maintained in disk for fine-grained post verification. For each query q, the candidate answers are selected via in-memory ANN search; then, the dense embeddings of the shortlisted candidates are loaded from disk, with which the fine-grained post verification result A_q is generated. With the above treatments, the EBR becomes scalable, where billions of embeddings can be hosted at merely tens of gigabytes RAM usage.

1.1.1 Limitations on representation learning. Despite the satisfaction of memory space constraint, the learning of bi-granular document representation remains a challenging problem. The existing EBR algorithms typically learn the dense embeddings directly; when sparse embeddings are needed, they are unsupervisedly quantized from the well learned dense embeddings [10, 16, 24] (Figure 2 A). However, we argue that the conventional way of generating bi-granular document representation is *improperly optimized*, making it *incompatible* with our massive-scale EBR workflow where the candidate-search and post-verification are successively performed.

Particularly, the conventional dense embeddings are learned for the *Global Discrimination*: the capability of finding a query's relevant answers from the entire corpus. However, considering that the dense embeddings are used for post verification, it needs to emphasize the *Local Discrimination*: the capability of selecting the ground-truth from the shortlisted candidates. Therefore, it should be optimized on top of the candidate distribution induced by the sparse embeddings, which is different from the answer distribution over the entire corpus. Besides, it is also worth noting that the unsupervised compression is lossy [6, 36], which prevents the full coverage of relevant answers in candidate search.

1.2 Progressively Optimized Representation

To address the above problem, we propose the *Progressively Optimized Document Representation*, with learning objectives corrected for the "candidate-search plus post-verification" based retrieval (Figure 2 B). The sparse embeddings are supervisedly learned first: instead of running unsupervised quantization for the well-learned dense embeddings, the sparse embeddings are generated from *contrastive learning*, which optimizes the *global discrimination* and helps to enable high-quality answers to be covered in candidate search. More importantly, the sparse embeddings will formulate the candidate distribution, on top of which the dense embeddings can be optimized for the *local discrimination*. In this place, a novel sampling strategy called *locality-centric sampling* is introduced to facilitate the effective optimization of local discrimination: a bipartite proximity graph is established first for the queries and answers based on the distance measured by sparse embeddings; then, while generating the training instances, a random query is selected as the entry, starting from which queries and answers are sampled over a local area on graph. In this way, the negative samples within the same batch become locally concentrated. By learning to discriminate the ground-truth from the massive amount of "in-batch shared negative samples", the representation model will be equipped with superior capability on local discrimination, which substantially contributes to the post verification's accuracy.

1.2.1 Highlighted Merits. Thanks to the above features, the retrieval accuracy can be notably improved for massive-scale EBR. Besides, we surprisingly find that our method even outperforms the SOTA EBR methods in direct comparisons (i.e., the SOTA baselines directly use dense embeddings for retrieval tasks, where no quantization loss is incurred). Knowing that the sparse embeddings can be served with extremely small costs, such an advantage indicates that our bi-granular document representation may also be applied to generic EBR, where a moderate-scale corpus is given.

Comprehensive experimental studies are performed with popular benchmarks on web search and a billion-scale dataset on sponsored search. According to the experimental results, our method achieves notable improvements against the SOTA baselines in both massive-scale and generic EBR scenarios. Meanwhile, it fully maintains the high scalability and running efficiency. Our method is also applied to a commercial search engine, which brings strong revenue and

CTR gains to its advertising service. Finally, The contributions of this paper are highlighted as follows.

- We tackle the massive-scale EBR problem on top of the progressively optimized bi-granular document representation. To the best of our knowledge, this is the first systematic work of its kind, which notably improves the retrieval accuracy.
- Leveraging the contrastive quantization and locality-centric sampling, the global and local discrimination can be effectively optimized for the sparse and dense embeddings.
- The comprehensive experimental studies verify the effectiveness of our proposed method in both massive-scale and generic EBR scenarios.

We'll present the related work, methodology, and experimental studies in the rest part. The supplementary material is also provided, showing more details like notations and additional experiments.

2 RELATED WORK

In this section, we review two fundamental techniques of embedding-based retrieval: the deep learning based document representation, and the indexing algorithms for scalable vector search.

• **Deep Document Representation**. Document representation is a fundamental part in EBR. Recently, deep learning based methods thrive thanks to the generation of semantic-rich embeddings [21, 33]. The latest methods intensively explore the pretrained language models, e.g., BERT [9], RoBERTa [25], XLnet [38]. Such models leverage large-scale transformers as backbones and get fully pretrained with massive corpus, which achieves strong expressiveness, and contributes to numerous downstream tasks, such as document retrieval [4, 20, 37] and question answering [14, 19, 22].

Besides, substantial progress has also been made in training methods. One such area is negative sampling. It is found that the representation learning is influenced by the scale of negative samples. Thus, the in-batch negative sampling is introduced [5, 12, 20], where negative samples can be augmented in a cost free way. Later on, the cross-device negative sampling is utilized in [31, 36], where negative samples can be shared across all the distributed devices. Recent studies also prove that the usefulness of "hard" negative samples. In [20, 27], hard negatives are collected by BM25 score. In [31, 37, 39], even harder negatives are collected from ANN search. With the increased hardness, the representation quality is reportedly improved by notable margins. Finally, efforts have also been paid to positive samples. In [4, 12, 40], positives are introduced by the self-supervision algorithms. Following the idea of knowledge distillation, the representation models are learned w.r.t. a more expressive teacher [23, 26, 31] (usually a cross encoder): with pseudo labels scored for the given candidate list, additional supervision signals can be provided for the representation model (*i.e.*, the student).

• **Indexing Algorithms**. A major building block of EBR is the ANN index. In short, the ANN index organizes high-dimensional vectors with certain data structures such that the proximity search can be performed with sub-linear time complexities. The ANN index has been studied for a long time. Many of the early works were developed based on space partitioning; e.g., LSH [8], BBF KD-Tree [2], K-means Tree [29]. Recently, the quantization-based approaches received extensive attention thanks to the effectiveness and compactness; e.g., in IVFADC [17, 18], the inverted index is combined with PQ (product quantization) to facilitate fast retrieval of ANN query. Another important class of ANN index is the graph-based techniques, where the proximity graph is built for the entire corpus. One such method is HNSW [28]: a multi-layer structure of hierarchical proximity graphs are constructed, which achieve effective retrieval of the nearest neighbours with logarithmic complexity. Later on, several works have been proposed for the better construction of proximity graphs, e.g., NSG [11] and Vamana [35], which lead to improved search performance. The graph-based technique is also combined with quantization-based approach, e.g., IVFOADC+G+P [1], where the proximity graph is constructed for the fast retrieval of top-K centroids from a large inverted index. As of this writing, the graph-based methods remain the most competitive ANN algorithms in terms of recall and efficiency.

One challenging problem of ANN is that the index can be oversized for the memory, when facing a massive-scale corpus. Although quantization based methods can help, they are reported to be limited in recall [32, 35]. The latest approaches resort to hybrid storage. In DiskANN [34, 35], the Vamana graph is built for the quantized embeddings, and is maintained in RAM for cost-grained search; meanwhile, the original embeddings are kept in disk for post verification. In HM-ANN [32], the NSW graph is maintained in slow memory (PMM) and the pivot points are hosted in fast memory (DRAM). However, the requirement of massive slow memory in HM-ANN is expensive and unavailable in many situations. So far, DiskANN remains the general solution for large-scale ANN due to its relative competitiveness in precision and serving cost.

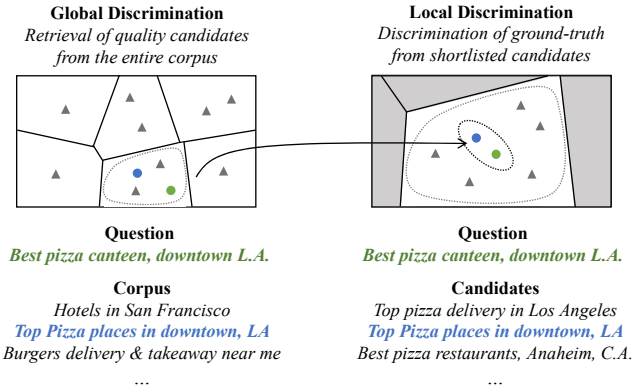

Figure 3: Progressive optimization process. Left: the sparse embeddings are learned first to optimize the global discrimination, which makes high-quality candidates to be retrieved. Right: the dense embeddings are learned to optimize the local discrimination conditioned on the candidate distribution, which ensures the effective discrimination of ground-truth (blue).

3 METHODOLOGY

The overall learning process is shown as Figure 3. In the first stage, the sparse embeddings are learned to optimize the *global discrimination*: to distinguish the right answers from the irrelevant ones in the entire corpus, such that query (green) and its related answers can be confined in the same laten area. In the second stage, the dense embeddings are learned to optimize the *local discrimination*: to identify the ground-truth from the shortlisted candidates promoted by the sparse embeddings.

3.1 Sparse Embedding Learning

We will first present the encoding network for the sparse embeddings. Then, we'll formulate the contrastive learning to optimize the global discrimination, together with its negative sampling strategy.

3.1.1 ADC Siamese Network. Note that the memory-efficient representation is only needed for the answers, which call for indexing and in-memory storage. Thus, the answers are quantized for sparse representation, while the queries remain densely represented. We adapt the siamese network with asymmetric distance computation (ADC) [18], in which the sparse embedding relies on its selected codewords to compute the distance with the dense embedding. Particularly, the answer's encoding involves the following steps. Firstly, each answer a is projected into its dense representation. Following the recent works on document representation [20, 27], we use BERT as the encoding backbone. The generated dense vector is denoted as $\mathbf{z}^a$. Secondly, the dense vector is quantized into binary codes. Here, the quantization is made through a differentiable PQ component [6]. The quantizer is parameterized with M codebooks, each codeword contains P codewords: $\{\mathbf{C}_{i,j}\}_{M,P}$. Given the dense representation $\mathbf{z}^a$, a codeword is selected from each of the codebooks:

$$\mathbf{C}_i^* = \arg\max_j \{\langle \mathbf{z}_i^a, \mathbf{C}_{i,j} \rangle\}, \tag{1}$$

where $\mathbf{z}_i^a$ is the i-th segment of $\mathbf{z}^a$. The argmax operator is realized based on the straight through estimation [3], such that it is end-to-end differentiable. The sparse embedding for the answer consists of M one-hot vectors, corresponding to the ID of each selected codeword. Finally, the sparse embedding concatenates all the selected codewords for distance computation:

$$f_s(a) = \text{concat}([\mathbf{C}_1^*, ..., \mathbf{C}_M^*]). \tag{2}$$

The query is directly encoded by BERT, which is the same as the intermediate result $\mathbf{z}^a$ in answer encoding.

3.1.2 Contrastive learning. Instead of running unsupervised quantization on the dense embeddings, the sparse embeddings are supervisedly learned for the optimized searching accuracy. As discussed, the sparse embeddings are used to select quality candidates from the entire corpus, which is illustrated by the following operation:

$$\widehat{A}_q \leftarrow \text{Top-N}(\{\langle g(q), f_s(a) \rangle | A\}). \tag{3}$$

In this place, $\widehat{A}_q$ is the candidate set, whose size N is much smaller than the corpus scale, *i.e.*, $\text{N} \ll |A|$. $g(\cdot)$ is the query's encoder and $f_s(\cdot)$ is the sparse answer encoder. We use inner product $\langle \cdot \rangle$ for the measurement of embedding similarity. The supervised learning is performed to minimize the following loss:

$$\arg\min_{g,\, f_s} \sum_q \sum_{a^+} \sum_{a^-} l(g(q), f_s(a^+), f_s(a^-)), \tag{4}$$

where a^+ and a^- are the positive and negative answers to the query, respectively. Here, we use InfoNCE [30] as our loss function:

$$\sum_{a^-} l(q, a^+, a^-) \leftarrow -\log \frac{\exp(\langle g(q), f_s(a^+) \rangle)}{\sum_{a^+ \cup A_q^-} \exp(\langle g(q), f_s(a) \rangle)}, \tag{5}$$

where A_q^- denotes the set of negative samples w.r.t. q and a^+.

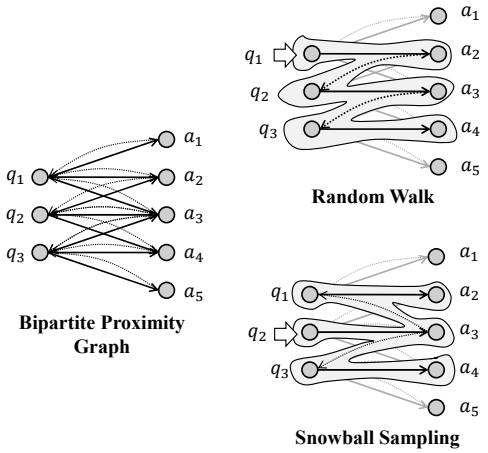

Figure 4: Locality Centric Sampling. (1) bipartite proximity graph construction: each query connects to its top-N candidate answers; each answer connects back to its original query. (2) Starting from the entry query, the training instances are sampled on a local patch via two optional strategies: random walk and snowball sampling.

3.1.3 Negative Sampling. Knowing that the scale of negative samples substantially contributes to the representation quality [5, 31], we leverage in-batch negatives [20], where the positive answer in one training instance is used for the augmentation of other instance's negative samples. Besides, we further sample one lexically similar negative for each query based on BM25, which is proved to be helpful for document representation [20, 27]. The lexically similar negatives are also shared across the training instances. With the above treatments, the negative sample set A_q^- in Eq. 5 becomes:

$$A_q^- = \{a^+\}_{q' \neq q} \cup \{a^-\}_B, \tag{6}$$

where $\{a^+\}_{q' \neq q}$ denotes the positive answers from other queries q', and $\{a^-\}_B$ means all the BM25 negatives within the same mini-batch batch. That is to say, each query may get almost $2|B|$ negative samples ($|B|$ is the batch size). Notice that we may also collect even "harder" negative samples than the lexical-similar ones with ANN-based negative sampling [31, 37, 39]; however, we experimentally find that having such treatments in the first stage does not contribute to the overall precision after post verification, given that the local discrimination will be fully optimized for the dense embeddings. As a result, we stay with the BM25 based negative samples for its effectiveness and simplicity.

3.2 Dense Embedding Learning

We rely on the siamese encoder with BERT backbone to generate the dense embeddings. The supervision objective is adapted, and the locality-centric sampling (LCS) is introduced to realize strong local discrimination. Finally, a post processing treatment is introduced to unify the query encoders learned in the first and second stage.

3.2.1 Supervision objective. The dense embeddings are learned conditionally w.r.t. the candidate distribution induced by the sparse embeddings: to identify the ground-truth from the candidates promoted by the sparse embeddings. Such a capability is called the

Algorithm 1: Training

Input : $\{(q, a^+)\}$, A
Output: f_s, f_d, g, g', g''

begin
  % (*Learn sparse embeddings*)
  f_s, $g \leftarrow$ argmin InfoNCE in Eq. 5;
  % (*Learn dense embeddings*)
  construct bipartite proximity graph $\mathcal{G}$;
  while *not converge* **do**
    get mini-batch B by performing LCS on $\mathcal{G}$;
    update f_d, g' by minimizing InfoNCE in Eq. 7 on B;
  $g'' \leftarrow$ argmin InfoNCE in Eq. 9;

local discrimination, where the InfoNCE loss is adapted as follows:

$$\sum_{a^-} l(q, a^+, a^-) \leftarrow -\log \frac{\exp(\langle g'(q), f_d(a^+)\rangle)}{\sum_{a^+ \cup \widehat{A}_q^-} \exp(\langle g'(q), f_d(a)\rangle)}. \quad (7)$$

Here, $g'(\cdot)$ and $f_d(\cdot)$ are the query encoder and dense answer encoder (we'll learn a new query encoder $g'(\cdot)$ first, and then unify $g'(\cdot)$ and $g(\cdot)$ in the post processing). $\widehat{A}_q^-$ are negative samples collected for this stage, which are different from the ones used in first stage, as Eq. 6. Particularly, $\widehat{A}_q^-$ become a subset of answers randomly sampled from the candidate set:

$$\widehat{A}_q^- \leftarrow \mathcal{S}(\widehat{A}_q). \quad (8)$$

Here, $\widehat{A}_q$ is the top-N candidates generated as Eq. 3, and $\mathcal{S}(\cdot)$ is the random selection operator. We empirically find that $|\widehat{A}_q^-|$ should be large enough in order to effectively optimize the local discrimination. However, one challenging problem of having a large $|\widehat{A}_q^-|$ is the heavy encoding cost. In many cases, it is infeasible to keep a large sample size as the GPU capacity is limited and the encoding cost per answer can be high. To overcome this problem, the locality-centric sampling is proposed to expand $\widehat{A}_q^-$ in a cost-efficient way.

3.2.2 Locality Centric Sampling. We introduce LCS for the augmentation of $\widehat{A}_q^-$. Firstly, the Bipartite Proximity Graph is constructed, which organizes the queries and answers based on their sparse embedding similarity. Secondly, the random sampling is performed on a local patch of the graph, which iteratively generates queries and their corresponding hard negatives. By doing so, the negative sample of one query may also be shared as a hard negative sample to another query within the same mini-batch. Details about LCS are introduced as follows.

• **Bipartite Proximity Graph**. We first construct the bipartite proximity graph for the queries and answers based on their sparse embeddings similarity, as Figure 4. Particularly, the Top-N candidate answers $\widehat{A}_q$ are selected for each query, as Eq. 3 (N=200 while constructing the graph). Then, a forward edge is added (solid line), which connects query q to an answer a in $\widehat{A}_q$. Besides, a backward edge is also added (dash line), which points back to q from a.

• **Sampling on Graph**. Starting from a random query q as the entry point, one of the following strategies: Random Walk and Snowball Sampling (Figure 4), is performed to generate a mini-batch of training instances. Each strategy has its pros and cons in different situations, which will be discussed in our experiments.

Algorithm 2: ANN Search

% (*Offline Index Construction*)
begin
  infer sparse & dense embeddings: $\{f_s(a)|A\}$, $\{f_d(a)|A\}$;
  build HNSW with $\{f_s(a)|A\}$ and host it in memory;
  maintain $\{f_d(a)|A\}$ in disk;
% (*Online ANN Serving*)
begin
  encoder query as $g''(q)$;
  $\widehat{A}_q \leftarrow$ search HNSW for Top-N($\{\langle f_s(a), g''(q)\rangle|A\}$);
  load $\{f_d(a)|\widehat{A}_q\}$ into memory;
  $A_q \leftarrow$ post re-rank for Top-K($\{\langle f_d(a), g''(q)\rangle|\widehat{A}_q\}$);

Random Walk: the sampling follows a zigzag path. For each visited query q_i, a random answer a_i^- is selected from q_i's connected candidates (with ground-truth excluded); then, another query q_{i+1} is sampled from the backward connections of a_i^-. It can be comprehended that the next answer a_{i+1}^- sampled for query q_{i+1} is similar with a_i^-; therefore, it is a potential candidate to q_i, which will contribute to the local discrimination.

Snowball Sampling: one visited query q will sample a random answer a^- from its neighbours; then all directly connected queries of a will be visited in the next step. Compared with random walk, the training instances collected by snowball sampling can be more concentrated within a local area, which improves the in-batch samples' hardness but potentially reduces the diversity.

The sampling process is consecutively run to generate $\{q, a^+, a^-\}$ (the positive answer a^+ is directly collected when q is visited). The sampling completes when the mini-batch is filled up by the collected instances. Finally, the negative answers for a query q will be: $\{a^+\}_{q' \neq q}$ (others' positives) plus $\{a^-\}_B$ (negatives within the batch), which means each query will get as many as 2$|B|$-1 locally concentrated negative samples.

3.2.3 Query Unification. So far, we've learned two query encoders $g(\cdot)$ and $g'(\cdot)$ paired with the sparse and dense answer embeddings, respectively. Although this may not cause extra latency in online serving, as both encoders may run in parallel, it still incurs twice encoding cost. In case the computation capacity is limited, we unify the query encoders in a simple but effective way, so that each query only needs to be encoded once. Particularly, we initialize $g''(\cdot)$: $g''(\cdot) \leftarrow g'(\cdot)$, and fix the answer encoders: $f_s(\cdot).fix$, $f_d(\cdot).fix$ (the fixed answers let us maintain a massive negative sample queue [15], which benefits the fine-tuning effect). Then, we finetune the query encoder to minimize the combined InfoNCE loss:

$$\begin{aligned} \arg\min_{g''} \sum_q \sum_{a^+} l(g''(q), f_s(a^+).fix, f_s(a^-).fix|A_q^-) + \\ \sum_q \sum_{a^+} l(g''(q), f_d(a^+).fix, f_d(a^-).fix|\widehat{A}_q^-). \end{aligned} \quad (9)$$

The upper part is the InfoNCE loss in Eq. 5; the lower part is the InfoNCE loss in Eq. 7. By optimizing the above combined loss, the unified query encoder $g''(\cdot)$ is learned. We find that the original precision can be well-preserved after the unification.

3.3 Training and Search Algorithm

The training is summarized as Alg. 1. Firstly, the sparse answer encoder $f_s(\cdot)$ and the query encoder $g(\cdot)$ are learned to minimize the InfoNCE loss in Eq. 5. Secondly, the bipartite proximity graph $\mathcal{G}$ is constructed. The mini-batch B is sampled by performing LCS on $\mathcal{G}$, with which the dense encoder $f_d(\cdot)$ and the query encoder $g'(\cdot)$ are learned w.r.t. the InfoNCE loss in Eq. 7. Finally, the query encoder is unified as $g''(\cdot)$ w.r.t. the joint InfoNCE in Eq. 9.

The ANN search (Alg. 2) involves two parts: the offline index construction, and the online serving of ANN request. For the offline stage: the sparse embeddings $f_s(a)$ and dense embeddings $f_d(a)$ are inferred for the entire answers A. Then, HNSW [28] is constructed for $f_s(a)$, which is standby in memory. Meanwhile, all the dense embeddings $\{f_d(a)|A\}$ are stored in disk. For online ANN serving: each input query q is encoded as $g''(q)$, which is used to search HNSW for the Top-N candidates $\widehat{A}_q$. Then, the dense embeddings are loaded into memory for the candidates. Finally, the fine-grained Top-K result A_q is generated by ranking $\hat{A}_q$ based on $\{f_d(a)|\widehat{A}_q\}$. Note that when computation capacity is sufficient, we'll still maintain two query encoders $g(q)$ and $g'(q)$ running in parallel, which may slightly benefit the retrieval accuracy.

4 EXPERIMENTAL STUDIES

The experimental studies focus on five issues. First and foremost, the evaluation of massive-scale EBR. Secondly, the evaluation of generic EBR (with a moderate-scale corpus). Thirdly, the efficiency and scalability. Fourthly, the impact from each of the proposed techniques. Finally, the impact to real-world search platforms. Part of the settings and additional results are shown in the appendix due to limited space.

4.1 Settings

4.1.1 Datasets. The TREC 2019 Deep Learning (DL) Track is a widely used benchmark on web search, which is taken as our primary experimental subject. The benchmark includes two tasks: the passage retrieval **Web Search (P)**, with a corpus of 8,841,823 answers; and the document retrieval **Web Search (D)**, with a corpus of 3,213,835 answers. We follow the official settings in [7]. Besides, we utilize an industrial dataset on **Sponsor Search**, whose corpus contains one billion ads from a commercial search engine. As far as we know, this might be the largest corpus used in relevant studies, which helps us better analyze the performance in massive-scale EBR. In this dataset, each query needs to match the ground-truth ad (clicked by the user). There are 3,000,000 queries for the training set and 10,000/10,000 queries for the valid/test sets. Each query has one ground-truth advertisement. Note that the datasets are highly diversified in answers' lengths: Web Search (D) has an average length of 448.5 tokens, which is much longer than Web Search (P) (79.6) and Sponsored Search (8.1). Such a difference brings substantial impact to some of the experiment, which will be shown in our analysis.

4.1.2 Baseline Methods. We make comparison with the latest document representation methods, which achieve strong performances on tasks like web search and question answering. 1) **DPR** [20], where in-batch negative samples are used for the training of siamere BERT based document encoders. It achieves competitive dense retrieval performance on open domain QA tasks. 2) **DE-BERT** [27], which adapts DPR by introducing extra hard negative samples. 3) **ANCE** [37], which iteratively mines hard negatives globally from the entire corpus. It is one of the strongest dense retrieval baselines on MS MARCO benchmark. 4) **STAR** [39], which improves ANCE by introducing in-batch negatives for stabilized and more accurate document representation. We use the following ANN indexes in the experiments. To evaluate the massive-scale EBR performance, the SOTA scalable ANN algorithm **DiskANN** [35] is leveraged. To evaluate the generic EBR performance, we adopt **HNSW** [28], which is one of the most effective ANN algorithms in practice.

4.2 Analysis

4.2.1 Evaluation of massive-scale EBR. The evaluations of massive-scale EBR are shown in Table 1, in which the baseline document representations are combined with DiskANN (## + **DISK**): the dense embeddings are compressed by PQ and indexed by Vamana graph. Our method (**BiDR**: **Bi**-granular **D**ocument **R**epresentation) has two forms: **BiDR (RD)**, where the LCS is performed via Random Walk; and **BiDR (SN)**, where the LCS is performed via Snowball sampling. Both baselines and our methods receive 1000 answers from candidate search, and re-rank them for the fine-grained Top-K answers. According to the experiment results, both BiDR (RD) and (SN) outperform the baselines with consistent and notable advantages; e.g., the **Recall@10** can be relatively improved by as much as **4.34**% and **2.66**% on Web Search (P) and (D), compared with the strongest baselines. Given that Sponsored Search has a billion-scale corpus (more than 100× larger than Web Search), BiDR's advantages in recall rate become expanded, e.g., the **Recall@10** can be relatively improved by **17.45**%.

More detailed analysis is made as follows. The performance of BiDR (SN) is better than BiDR (RD) on Web Search (P) and Sponsored Search, but becomes worse than BiDR (RD) on Web Search (D). One notable difference is that Web Search (P) and Sponsored Search have much smaller answer lengths (79.6 and 8.1) than Web Search (D) (448.5). Thus, the encoding of each training instance (one query with its positive and negative answers) takes much smaller memory consumption for both datasets. Knowing that the batch size is up-bounded by the GPU RAM capacity, it will call for a large amount of LCS sampling steps to fill up a mini-batch when applied to Web Search (P) and Sponsored Search. As a result, the random walk may get divergent and introduce a great deal of "remote negative samples", which is unfavorable to the local discrimination. In contrast, snowball helps to keep the sampling concentrated, which alleviates this problem. However, the number of LCS steps becomes highly limited in Web Search (D), given the large memory consumption per training instance. In this case, the random walk helps to introduce more "diversified hard negative samples", which provides more informative supervision signals than the snowball sampling.

As for baselines: DE-BERT is stronger than DPR; meanwhile, ANCE further outperforms DE-BERT. It indicates hard negatives, especially the ANN hard negatives, are beneficial. Besides, STAR achieves the highest baseline performance, where in-batch and ANN negatives are jointly used. Our experiments in 4.2.4 will show that the effects of ANN and in-batch negatives are different: the former one is more related with the local discrimination, while the latter one is more related with the global discrimination. The two objectives may become contradicted in certain situations.

	Web Search (P)			Web Search (D)			Sponsored Search		
Methods	**R@10**	**R@50**	**R@100**	**R@10**	**R@50**	**R@100**	**R@10**	**R@50**	**R@100**
DPR + DISK	0.5297	0.7610	0.8317	0.5969	0.8083	0.8623	0.3205	0.5043	0.5545
STAR + DISK	0.5834	0.7807	0.8331	0.6807	0.8575	0.8925	0.3438	0.5480	0.6007
ANCE + DISK	0.5745	0.7804	0.8331	0.6680	0.8336	0.8659	0.3258	0.5247	0.5757
DE-BERT + DISK	0.5537	0.7682	0.8265	0.5998	0.8089	0.8563	0.3279	0.5174	0.5659
BiDR (RD)	0.6048	0.8156	0.8753	**0.6988**	**0.8821**	0.9176	0.3985	0.6467	0.7152
BiDR (SN)	**0.6087**	**0.8169**	**0.8786**	0.6959	0.8808	**0.9225**	**0.4038**	**0.6489**	**0.7158**

Table 1: Evaluation of massive-scale EBR. BiDR (RD) and (SN) outperform all baselines with notable advantages.

	Web Search (P)					Web Search (D)				
Methods	**R@10**	**R@20**	**R@30**	**R@50**	**R@100**	**R@10**	**R@20**	**R@30**	**R@50**	**R@100**
DPR + HNSW	0.5328	0.6413	0.6984	0.7653	0.8405	0.5983	0.7059	0.7562	0.8110	0.8688
STAR + HNSW	0.5919	0.6917	0.7468	0.8038	0.8611	0.6816	0.7775	0.8236	0.8648	0.9044
ANCE + HNSW	0.5842	0.6858	0.7427	0.7991	0.8596	0.6728	0.7633	0.8060	0.8476	0.8840
DE-BERT + HNSW	0.5540	0.6580	0.7121	0.7709	0.8350	0.6112	0.7142	0.7660	0.8195	0.8707
BiDR (RD)	0.6048	0.7098	0.7604	0.8156	0.8753	**0.6988**	**0.7964**	**0.8399**	**0.8821**	0.9176
BiDR (SN)	**0.6087**	**0.7116**	**0.7604**	**0.8169**	**0.8786**	0.6959	0.7924	0.8357	0.8808	**0.9225**

Table 2: Evaluation of generic EBR. BiDR (RD) and (SN) maintain their leading performances after baselines' switching to HNSW.

4.2.2 Evaluation of generic EBR. In this part, the baseline ANN index is switched to HNSW, which is more accurate than DiskANN but no longer memory efficient. Given that the Sponsored Search dataset has a billion-scale corpus, which is too large for HNSW to fit into memory, we leverage Web Search (P) and (D) for the evaluation. The experiment results are shown as Table 2: with the utilization of HNSW, the baselines' performances are improved compared with their previous results in Table 1. At the same time, BiDR (RD) and (SN) still maintain their advantages over the strongest baselines; e.g., **Recall@10** can be relatively improved by **2.66%** and **2.52%** on different datasets. Such a finding indicates our bi-granular embeddings' competitiveness against the conventional EBR methods, which rely on a unified set of dense embeddings. The explanation about our advantage is made as follows. *The global discrimination and local discrimination are two different objectives, whose effects can be contradicted*: a too strong local discrimination is unfavorable to the global discrimination, and vice versa. Therefore, instead of enforcing a unified set of embeddings, it's better to leverage two sets of embeddings which work collaboratively to optimize each of the objectives. This point will be further clarified by our experiments on sampling strategies in 4.2.4. Besides, it should also be noted that the maintenance of bi-granular embeddings is not expensive at all: for one thing, the bi-granular embeddings are inferred offline; for the other thing, the sparse embeddings are highly memory and search efficient, where little extra cost is introduced compared with the method which merely uses the dense embeddings.

4.2.3 Efficiency and Scalability. The **evaluation of efficiency** is shown as Figure 5: **A1**/**B1** for Web Search (P)/(D), **C1** for Sponsored Search. In our experiment, the time latency and recall rate are measured for different scales of candidates. On one hand, the expanded candidate size will contribute to a higher recall rate, as the ground-truth answers are more likely to be included (the ANN search will be reduced to linear scan when the entire corpus is included). On the other hand, the expanded candidates will increase the cost of search and post verification, which incurs a higher time latency. In our experiment, the candidate size is increased from 100 to 10000, which leads to the growth of recall rate as well as time latency. For Web Search (P) and (D): we observe that the **Recall@100** about BiDR (RD) and (SN) soon reach a highly competitive level (less than **0.1 ms**, with only around **300** candidates included), which indicates the time efficiency of our method. Compared with A1 and B1, it takes more latency for BiDR to reach a stable competitive Recall@100 on Sponsored Search, as shown in C1 (about **8ms**, with **5000** candidates retrieved and re-ranked). Considering that Sponsored Search has a billion-scale corpus (over ×100 larger than the Web Search datasets), the increase of search cost can be expected. Besides, the observed latency is indeed competitive in reality, as it is measured on the basis of one physical machine (detailed specification is made in Appendix). By comparison, "STAR + DiskANN" (the strongest baseline) is limited when the candidate size is small; this is mainly caused by the lossy unsupervised quantization of DiskANN. Besides, the performance gap between BiDR and "STAR + DiskANN" still remains when huge amounts of candidates are included (**10000**); this is resulted from the superiority of our progressively optimized bi-granular embeddings, as analyzed in 4.2.2.

The scalability is evaluated as Figure 5 $\{A2, B2, C2\}$. The scale of sparse embeddings is gradually increased by expanding the codebooks, whose memory cost grows from 64 to 1024 bit. The data quantization will be less lossy when more bits are allocated to the sparse embeddings; thus, the recall rate can be improved. For all datasets: BiDR soon grows to high recall rates with less than **256 bit** (per answer) memory cost, which means an one-billion corpus can be hosted for high-quality ANN search with merely **32 GB** RAM cost. Such a finding reflects our capability of handling massive-scale EBR scenarios. Similar to the observations made in efficiency analysis: 1) for Sponsored Search, all methods suffer from low recall accuracy at small bit rates; and 2) DiskANN+STAR's recall rates are limited at small bit rates, and the performance gap remains at high bit rates. We attribute these observations to the same reasons as explained in our efficiency analysis.

4.2.4 Ablation Studies. The detailed impact is investigated for contrastive quantization, locality-centric sampling, and unified query.

Quantization. The evaluation of our *contrastive* quantization is shown in the upper part of Table 3, with three representative baselines compared: 1) the typical unsupervised method *PQ* [17]; 2) *OPQ* [13], which iteratively learns the embedding's projection and

	Web Search (P)			Web Search (D)			Sponsored Search		
Quantization	**R@50**	**R@100**	**R@1000**	**R@50**	**R@100**	**R@1000**	**R@50**	**R@100**	**R@1000**
PQ	0.7127	0.7867	0.9349	0.4032	0.4900	0.7488	0.2413	0.3167	0.6000
OPQ	0.7598	0.8270	0.9519	0.7864	0.8490	0.9483	0.2629	0.3320	0.6140
DPQ	0.7696	0.8399	0.9558	0.7993	0.8571	0.9543	0.3359	0.4288	0.7218
Contrastive	**0.7801**	**0.8460**	**0.9605**	**0.8182**	**0.8702**	**0.9612**	**0.4067**	**0.5004**	**0.7659**
Sampling	**R@10**	**R@50**	**R@100**	**R@10**	**R@50**	**R@100**	**R@10**	**R@50**	**R@100**
ANN	0.6021	0.8101	0.8728	0.6921	0.8709	0.9109	0.3835	0.6302	0.7041
ANN + In-batch	0.6014	0.8092	0.8727	0.6861	0.8698	0.9107	0.3755	0.6254	0.6999
Random Walk	0.6048	0.8156	0.8753	**0.6988**	**0.8821**	**0.9176**	0.3985	0.6467	0.7152
Snowball	**0.6087**	**0.8169**	**0.8786**	0.6959	0.8808	0.9225	**0.4038**	**0.6489**	**0.7158**
Unified Query	**R@10**	**R@50**	**R@100**	**R@10**	**R@50**	**R@100**	**R@10**	**R@50**	**R@100**
Unified (RD)	0.6027	0.8121	0.8663	**0.7003**	0.8785	0.9147	0.3962	0.6346	0.7046
w.o. Unified (RD)	0.6048	0.8156	0.8753	0.6988	**0.8821**	0.9176	0.3985	0.6467	0.7152
Unified (SN)	0.6003	0.8098	0.8704	0.6924	0.8775	0.9143	0.4002	0.6399	0.7064
w.o. Unified (SN)	**0.6087**	**0.8169**	**0.8786**	0.6959	0.8808	**0.9225**	**0.4038**	**0.6489**	**0.7158**

Table 3: Ablation studies: the impact of (1) contrastive quantization, (2) locality-centric sampling (LCS) and (3) query unification. (Recall@1000 is reported for the quantization methods, considering that the Top-1000 answers are selected in candidate search.)

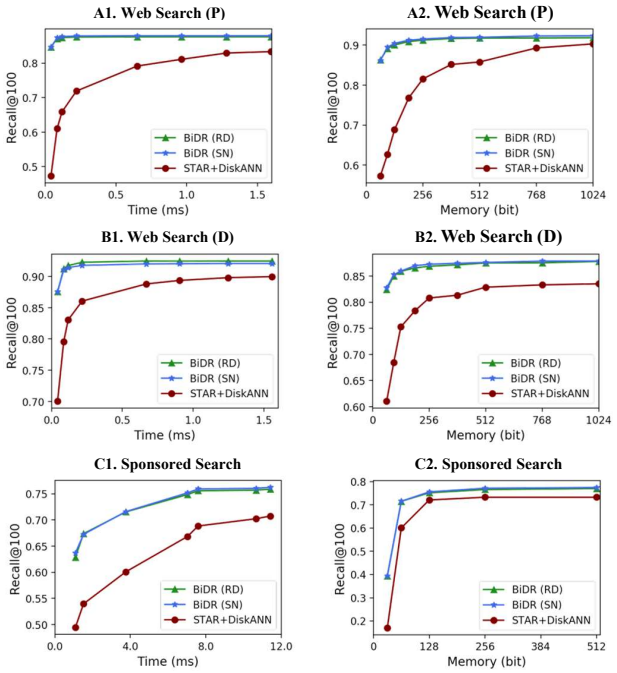

Figure 5: Evaluation of efficiency (#1) and scalability (#2). BiDR reaches higher recall rates with less latency and memory cost.

codebooks to minimize the reconstruction loss; 3) *DPQ* [6], where the reconstruction loss and matching loss are jointly minimized within one network. The recall rates are reported for the candidate search result, whose size is much larger than the final retrieval result. We may find that the direct use of PQ is lossy, whose accuracy is the lowest of all. Other baselines are improved thanks to the effective reduction of reconstruction loss; however, they are still inferior to our contrastive quantization, where the InfoNCE loss (the objective for retrieval accuracy) is minimized for the sparse embeddings. In fact, we find that the gap in recall rate between our contrastive quantization and the continuous upper bound (where dense embeddings are used without quantization) can be even smaller than 1e-3 for the Top-1000 candidates (the default size of candidates). Such a tiny loss is almost ignorable in practice.

Sampling. The impacts of locality-centric sampling methods (LCS): *Random Walk* and *Snowball* Sampling, are evaluated against two baseline strategies: 1) *ANN* negative sampling as used by ANCE [37], where each query is assigned with a fixed amount of negatives sampled by ANN; 2) *ANN + In-batch* as used by STAR [39], where the ANN negatives are combined with in-batch negatives. The evaluation results are shown as the middle part of Table 3, in which both Random Walk and Snowball outperform the baseline sampling strategies. Such an observation verifies that LCS is beneficial to the learning of dense embeddings, where the highly concentrated in-batch negatives may provide strong supervision signals for the local discrimination. Besides, as discussed in 4.2.1, Random Walk and Snowball take the lead under different conditions. For Web Search (P) and Sponsored Search where a great deal of consecutive LCS samplings are needed, Snowball achieves higher accuracy because of the better preservation of locality. For Web Search (D) where LCS sampling steps are limited to be small, Random Walk performs better as more diversified samples can be introduced.

One interesting observation to emphasize: although STAR is generally better than ANCE (Table 1 and 2), its sampling strategy (ANN + In-batch) is worse than the one used by ANCE (ANN) when applied to our dense embeddings' learning. A fundamental difference between the two strategies is that the ANN negatives focus on the local discrimination, while the ANN + In-batch negatives aim to reach a trade-off between the local and global discrimination. Knowing the dense embeddings are learned for post verification, where the local discrimination matters the most, such a trade-off is undesirable and leads to a negative effect on retrieval accuracy. This observation echoes our analysis in 4.2.2, which indicates that the effects of local and global discrimination are not always consistent. On top of our progressively optimized bi-granular document representation, *i.e.*, with the sparse embeddings optimized for the global discrimination and the dense embeddings optimized for local discrimination, the retrieval accuracy can be effectively enhanced.

Query Unification. We analyze the impact of unifying the query encoder (Table 3). It is observed that the loss from query

unification is small; e.g., the gaps between the non-unified results are merely **-0.0015** ∼ **0.0084** (**0.0031** on average) in terms of **Recall@10**. Such a performance loss is far smaller than BiDR's advantages over the baselines (shown in Table 1 and 2), which means the query unification may well preserve the retrieval accuracy while cutting down BiDR's online serving cost. Meanwhile, given that the performance of "w.o. Unified" is still slightly better in general, people may keep the non-unified query encoders running in parallel when the system's capacity is abundant.

4.2.5 Online Performance. BiDR has been successfully applied to a major sponsored search platform in the world, which retrieves ads in response to user's search queries. The previous system already adopted an ensemble of competitive scalable EBR solutions, such as BERT based sentence embeddings plus DiskANN, whose performances are on par with the strongest baselines in our offline studies. Even with such a strong comparison group, BiDR achieves notable improvements for the key performance indicators: there have been **+1.95% RPM gain** (revenue per mille, i.e., per thousand impressions), **+1.01% Recall gain**, and **+0.49% CTR gain**, which means substantial improvements on profit and user experience.

5 CONCLUSION

In this work, we proposed the progressively optimized bi-granular document representation for scalable embedding based retrieval. The contrastive quantization was used to generate the sparse embeddings, which led to precise and memory-efficient candidate search. Conditioned on the candidate distribution, the dense embeddings were continuously learned with locality-centric sampling, which achieved superior post verification accuracy. Comprehensive offline and online evaluations were performed, whose results verified the effectiveness, efficiency and scalability of our method.

REFERENCES

[1] Dmitry Baranchuk, Artem Babenko, and Yury Malkov. 2018. Revisiting the inverted indices for billion-scale approximate nearest neighbors. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 202–216.

[2] Jeffrey S Beis and David G Lowe. 1997. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE, 1000–1006.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

[4] Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. *arXiv preprint arXiv:2002.03932* (2020).

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[6] Ting Chen, Lala Li, and Yizhou Sun. 2020. Differentiable product quantization for end-to-end embedding compression. In *International Conference on Machine Learning*. PMLR, 1617–1626.

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820* (2020).

[8] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*. 253–262.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Miao Fan, Jiacheng Guo, Shuai Zhu, Shuo Miao, Mingming Sun, and Ping Li. 2019. MOBIUS: towards the next generation of query-ad matching in baidu's sponsored search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2509–2517.

[11] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2017. Fast approximate nearest neighbor search with the navigating spreading-out graph. *arXiv preprint arXiv:1707.00143* (2017).

[12] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. *arXiv preprint arXiv:2104.08821* (2021).

[13] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence* 36, 4 (2013), 744–755.

[14] 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).

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9729–9738.

[16] 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.

[17] 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.

[18] Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. 2011. Searching in one billion vectors: re-rank with source coding. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 861–864.

[19] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics* 8 (2020), 64–77.

[20] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 6769–6781.

[21] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. PMLR, 1188–1196.

[22] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300* (2019).

[23] 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).

[24] Yiding Liu, Weixue Lu, Suqi Cheng, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained Language Model for Web-scale Retrieval in Baidu Search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3365–3375.

[25] 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).

[26] Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. Twinbert: Distilling knowledge to twin-structured compressed BERT models for large-scale retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2645–2652.

[27] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, Dense, and Attentional Representations for Text Retrieval. *Transactions of the Association for Computational Linguistics* 9 (2021), 329–345.

[28] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[29] Marius Muja and David G Lowe. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)* 2, 331-340 (2009), 2.

[30] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[31] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 5835–5847.

[32] Jie Ren, Minjia Zhang, and Dong Li. 2020. Hm-ann: Efficient billion-point nearest neighbor search on heterogeneous memory. *Advances in Neural Information Processing Systems* 33 (2020).

[33] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd international conference on world wide web*. 373–374.

[34] Aditi Singh, Suhas Jayaram Subramanya, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. 2021. FreshDiskANN: A Fast and Accurate Graph-Based ANN Index for Streaming Similarity Search. *arXiv preprint arXiv:2105.09613*

(2021).
[35] Suhas Jayaram Subramanya, Rohan Kadekodi, Ravishankar Krishaswamy, and Harsha Vardhan Simhadri. 2019. Diskann: Fast accurate billion-point nearest neighbor search on a single node. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 13766–13776.
[36] Shitao Xiao, Zheng Liu, Yingxia Shao, Defu Lian, and Xing Xie. 2021. Matching-oriented Embedding Quantization For Ad-hoc Retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 8119–8129. https://doi.org/10.18653/v1/2021.emnlp-main.640
[37] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*. https://openreview.net/forum?id=zeFrfgyZln
[38] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).
[39] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1503–1512.
[40] Yan Zhang, Ruidan He, Zuozhu Liu, Kwan Hui Lim, and Lidong Bing. 2020. An unsupervised sentence embedding method by mutual information maximization. *arXiv preprint arXiv:2009.12061* (2020).

A ALGORITHM DETAILS

A.1 Notation Table

Notation	Description
A	entire corpus
$\widehat{A}_q$	candidate set for query q
C	codebook for the quantization module
$g(\cdot)$	query encoder in stage one
$g'(\cdot)$	query encoder in stage two
$g''(\cdot)$	unified query encoder
$f_s(\cdot)$	sparse answer encoder
$f_d(\cdot)$	dense answer encoder
$l(\cdot)$	InfoNCE loss function

Table 4: Notation Table

A.2 Locality Centric Sampling

The pseudo-code is showed in Algorithm 3:

Algorithm 3: Pseudocode of Locality Centric Sampling

```
# query2keys:
#   for each query, storage its positive answers
# query2negs:
#   queries' adjacency list for bipartite proximity graph
# neg2queries:
#   answers' adjacency list for bipartite proximity graph
# queries_set: set of unvisited queries
# random_sample: randomly sample one item

def get_next_query(q_queue, queries_set, strategy):
    # obtain the next query from queue
    if len(q_queue) == 0:
        return random_sample(queries_set)
    if strategy == 'RandomWalk':
        return q_queue[-1]
    if strategy == 'SnowBall':
        return q_queue[0]

def generate_batch_data(queries_set,
                        query2keys,
                        query2negs,
                        neg2queries,
                        strategy):
    # generate data for mini-batch training
    while len(queries_set) > 0:
        batch_data = []
        while len(batch_data) < batch_size:
              q = get_next_query(q_queue, queries_set,
                    strategy)
            k = random_sample(query2keys[q])
            n = random_sample(query2negs[q])
            batch_data.append([q, k, n])

            q_queue.extend(neg2queries[n])
            # delete q from set of unvisited queries
            queries_set.remove(q)
            # exit when all queries have be processed
            if len(queries_set) == 0: break
        return batch_data
```

B MORE EXPERIMENTS RESULTS

More results are provided in additional to the reported values in Table 1 and Table 3. The observations are consistent with our conclusion in Section 4.2.

B.1 Evaluation of massive-scale EBR

B.2 Ablation study

C NEGATIVE SAMPLING IN STAGE ONE

Despite that the ANN negative samples are useful in learning dense embeddings, we find that it does not contribute to the overall retrieval accuracy when applied to stage one (i.e., sparse embeddings' learning), as claimed in Section 3.1.3. Such a point is reflected by the following experimental results; i.e., "w.o. ANN" constantly leads to better performances.

	Web Search(P)		Web Search(D)		Sponsor Search	
Methods	**R@20**	**R@30**	**R@20**	**R@30**	**R@20**	**R@30**
DPR + DISK	0.6373	0.6937	0.7038	0.7529	0.4086	0.4575
STAR + DISK	0.6767	0.7289	0.7775	0.8236	0.4404	0.4919
ANCE + DISK	0.6744	0.7252	0.7567	0.7962	0.4170	0.4659
DE_BERT + DISK	0.6574	0.7102	0.7090	0.7600	0.4233	0.4682
BiDR(RD)	0.7098	**0.7604**	**0.7964**	**0.8399**	0.5160	0.5776
BiDR(SN)	**0.7116**	**0.7604**	0.7924	0.8357	**0.5217**	**0.5821**

Table 5: More results on massive-scale EBR in addition to Table 1.

D IMPLEMENTATION DETAILS

D.1 System Specifications

We conduct all experiments on one machine with eight NVIDIA-A100-40G GPUs and two AMD EPYC 7V12 64-Core CPUs. The models are implemented with python 3.6 and pytorch 1.8.0. We also use FAISS[1] in the implementation.

[1] https://github.com/facebookresearch/faiss

	Web Search(P)		Web Search(D)		Sponsor Search	
Sampling	**R@20**	**R@30**	**R@20**	**R@30**	**R@20**	**R@30**
ANN	0.7048	0.7553	0.7845	0.8304	0.4998	0.5604
ANN + In-batch	0.7017	0.7513	0.7814	0.8291	0.4903	0.5557
Random Walk	0.7098	**0.7604**	**0.7964**	**0.8399**	0.5160	0.5776
Snowball	**0.7116**	**0.7604**	0.7924	0.8357	**0.5217**	**0.5821**
Unified Query	**R@20**	**R@30**	**R@20**	**R@30**	**R@20**	**R@30**
Unified (RD)	0.7008	0.7536	0.7927	0.8368	0.5072	0.5664
w.o. Unified (RD)	0.7098	**0.7604**	**0.7964**	**0.8399**	0.5160	0.5776
Unified (SN)	0.7031	0.7536	0.7922	0.8367	0.5113	0.5765
w.o. Unified (SN)	**0.7116**	**0.7604**	0.7924	0.8357	**0.5217**	**0.5821**

Table 6: More results on ablation studies in addition to Table 3.

Sampling Method		Web Search(P)			Web Search(D)		
Sparse	Dense	**R@10**	**R@50**	**R@100**	**R@10**	**R@50**	**R@100**
w.ANN	RW	0.6021	0.8115	0.8710	0.6880	0.8678	0.9079
w.o.ANN	RW	**0.6048**	**0.8156**	**0.8753**	**0.6988**	**0.8821**	**0.9176**
w.ANN	SN	0.5966	0.8065	0.8659	0.6832	0.8740	0.9110
w.o.ANN	SN	**0.6087**	**0.8169**	**0.8786**	**0.6959**	**0.8808**	**0.9225**

Table 7: Impact of ANN negatives in sparse embedding learning.

D.2 Text Encoder

We use RoBERTa-base [25] as the backbone of text encoder for all the methods. The output embedding of the "[CLS]" token is used for the text representation result. For Sponsor Search dataset, we reduce the embedding size from 768 to 64 by learning an additional linear projection layer.

D.3 Hyper-parameter

We use AdamW optimizer to update models' parameters. For Web Search(D) dataset, the batch size is 160, and the documents are truncated to a maximum of 512 tokens. For Web Search(P), the batch size is set to be 1024 and the maximal length of documents is 128. For Sponsor Search, we set the maximal length of ads as 12, which allows us to use a larger batch size: 2048. The learning rates for Web Search(D), Web Search(P) and Sponsor Search are 5e-5, 1e-4, and 2e-4 respectively.

For the sparse embedding learning, the number of codewords in each codebook is fixed to be 256. We use OPQ to initialize the PQ module and the default number of codebooks is 96 on Web Search(P) and Web Search(D) datasets. Since the embedding size is smaller, we only use 8 codebooks for Sponsor Search. For dense embedding learning, the top-200 irrelevant documents ranked by sparse embedding are used as the hard negative set to construct the bipartite proximity graph. Following [37] and [39], we use DE-BERT as the warm-up model and random sample one hard negative from top-200 documents for ANCE and STAR models.