Pre-trained Language Model for Web-scale Retrieval in Baidu Search

Yiding Liu[#], Guan Huang[#], Jiaxiang Liu, Weixue Lu, Suqi Cheng, Yukun, Li, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, Dawei Yin[*]
Baidu Inc., China
{liuyiding.tanh,chengsuqi}@gmail.com
{huangguan01,liujiaxiang,luweixue,liyukun01,shidaiting01,wangshuaiqiang,chengzhicong01}@baidu.com
yindawei@acm.org

ABSTRACT

Retrieval is a crucial stage in web search that identifies a small set of query-relevant candidates from a billion-scale corpus. Discovering more semantically-related candidates in the retrieval stage is very promising to expose more high-quality results to the end users. However, it still remains non-trivial challenges of building and deploying effective retrieval models for semantic matching in real search engine. In this paper, we describe the retrieval system that we developed and deployed in Baidu Search. The system exploits the recent state-of-the-art Chinese pretrained language model, namely Enhanced Representation through kNowledge IntEgration (ERNIE), which facilitates the system with expressive semantic matching. In particular, we developed an ERNIE-based retrieval model, which is equipped with 1) expressive Transformer-based semantic encoders, and 2) a comprehensive multi-stage training paradigm. More importantly, we present a practical system workflow for deploying the model in web-scale retrieval. Eventually, the system is fully deployed into production, where rigorous offline and online experiments were conducted. The results show that the system can perform high-quality candidate retrieval, especially for those tail queries with uncommon demands. Overall, the new retrieval system facilitated by pretrained language model (i.e., ERNIE) can largely improve the usability and applicability of our search engine.

KEYWORDS

Pretrained Language Models; Information Retrieval; Search

ACM Reference Format:
Yiding Liu[#], Guan Huang[#], Jiaxiang Liu, Weixue Lu, Suqi Cheng, Yukun, Li, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, Dawei Yin[*]. 2021. Pre-trained Language Model for Web-scale Retrieval in Baidu Search. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Search engines (e.g., Google, Baidu, Bing) are critical tools for people to find useful information from massive web documents. A modern search engine usually employ a multi-stage pipeline that gradually narrows down the number of relevant documents (i.e., web pages), where *retrieval* is usually known as the very first stage. It aims at identifying a few hundreds or thousands of relevant candidates from the entire billion-scale corpus, which has significant impact to the overall capability of a search engine.

Nevertheless, the unprecedented scale and diversity of web documents impose many challenges to the retrieval system. First (**C1**), *semantic matching* [20] is one of the most critical concern, while conventional methods based on text matching (e.g., BM25 [34]) may easily fail at modeling relevant information with different phrasing. Worse still, an increasing number of queries are in the style of natural language, making the semantic modeling even more challenging. Second (**C2**), both search queries and web contents are highly heterogeneous, following long-tail distributions. For example, a large amount of low-frequency queries (i.e., tail queries) have never been seen before by the search engine. As such, the semantics of tail queries and documents are difficult to be accurately inferred. Third (**C3**), to create significant impact to real-world applications, it also calls for practical solutions on deploying the retrieval model to serve web-scale data.

Extensive efforts from both academia and industry have been dedicated to tackle these challenges. To conduct semantic retrieval, a wealth of studies [8, 14, 33, 45] have explored various bi-encoder models (i.e., Siamese networks or two-tower models), where different representation learning techniques has been employed as semantic encoders. Given a query and a document, the semantic encoder takes the query text and the document text (e.g., title) as inputs, and respectively produces two embeddings for the relevance computation. Recently, BERT [4] has made significant progress on natural language understanding, and thus has also been applied as a more powerful semantic encoder [2, 16, 23]. The BERT-based bi-encoder and its variants have achieved the state-of-the-art performance on retrieval tasks [1, 44], which can be mainly attributed to the expressive deep attention-based structure (i.e., Transformer) and the pretraining and fine-tuning paradigm. Although considerable research progress has been made, there is still a lack of investigation on how to develop and deploy such retrieval models in the online environment of a search engine.

Present work. In this paper, we introduce a novel retrieval system that is fully deployed in Baidu Search[1], the dominant search engine in China. In particular, the retrieval system is equipped with the recent state-of-the-art Chinese pretrained language model (PLM), namely *Enhanced Representation through kNowledge IntEgration (ERNIE)* [38], which enables effective semantic matching in the system. More specifically, we design the retrieval system with several key insights to tackle the aforementioned challenges:

[#] Co-first authors.
[*] Dawei Yin is the corresponding author.

Conference'17, July 2017, Washington, DC, USA
2021. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] www.baidu.com

- To tackle the first challenge (**C1**), we leverage an ERNIE-based (i.e., Transformer-based) bi-encoder to perform expressive semantic matching. In particular, the Transformer encoders directly take the raw texts of queries and web documents as inputs and encode their semantics in latent embeddings. The deep structure of Transformer encoders allow the complicated semantics to be more comprehensively modeled. The dense attention over the raw texts can also keep the semantics of fine-grained context, such as using different prepositions (e.g., "for" vs. "to"), which is more friendly to conversational queries. Moreover, we integrate a poly-interaction scheme [16] and effective training data mining strategies, which further improves the effectiveness of the retrieval model.
- To tackle the second challenge (**C2**), we further propose a multi-stage training paradigm for optimizing the retrieval model. In particular, the training stages are designed with different data sources and objectives, which allows rich knowledge to be absorbed by the model. Compared with training-from-scratch, our proposed paradigm can further boost the generalization ability of the model, which is especially beneficial for tail queries.
- To tackle the third challenge (**C3**), we develop a system architecture that serves the proposed model for large-scale web retrieval. In particular, we incorporate the semantic matching with conventional text matching to collaboratively generate relevant candidates. Moreover, we further introduce a lightweight post-retrieval filtering module that provides an unified filtering for the retrieval, where more statistical features (e.g., clickthrough rate, dwell time) can be introduced to consider the overall quality. Overall, the system architecture allows the proposed model to work together with conventional text-matching workflow and offers flexibility for including other features.

To the best of our knowledge, this is one of the largest applications of pre-trained language models for web-scale retrieval. We anticipate this paper to provide our practical experiences and new insights for incorporating PLMs in web retrieval. The main contributions can be summarized as follows:

- **The retrieval model.** We leverage Transformers as the semantic encoders for web retrieval, and further exploit a poly-interaction scheme and several strategies for mining training data. The model can capture complicated semantic information underlying query and web documents, and is effective for semantic matching.
- **Training paradigm.** We introduce a novel multi-stage training paradigm that facilitates the retrieval model to learn rich information. This would significantly improve the generalization ability of the model, especially for the tail queries.
- **System design**. We design an effective and efficient system workflow to server the model, allowing it to seamlessly work with the conventional workflow to boost the overall performance. We also introduce a post-retrieval filtering module, which integrates statistical features and semantic relatedness that measure the overall quality of the candidates in an unified manner.
- **Evaluation**. We conduct extensive offline and online experiments to validate the effectiveness of the retrieval system. The results show that the deployment of the system can significantly improve the usability and applicability of the search engine.

2 RELATED WORK

2.1 Semantic Retrieval in Web Search

Semantic retrieval is essential for a modern retrieval system. Typical structures of semantic retrieval models can be viewed as bi-encoders or Siamese networks [5], which comprises two encoders that conduct semantic modeling. Most of the existing studies mainly focus on designing the encoders with different representation learning techniques [8, 13, 14, 21, 33, 45]. A representative work, namely Deep Semantic Similarity Model (DSSM) [14], is one of the earliest DNN methods for semantic modeling with clickthrough data. The deep fully-connected architectures of the DSSM encoders can extract expressive semantic information and achieve superior performance on web search. Thereafter, other DNN-based retrieval methods have thrived over the past years [27], including those based on Convolutional Neural Networks (CNNs) [7, 35–37] and Recurrent Neural Networks (RNNs) [29, 30].

The common bi-encoder structure of those models allows a large number of document embeddings to be pre-computed and cached offline, which is very efficient for online retrieval. Therefore, such model structure has also been developed in real-world products [11, 13, 47]. For instance, Huang *et al.* [13] employ DNN-based bi-encoder in Facebook Search system, and introduce various practical experiences in the end-to-end optimization of the system. Zhang *et al.* [47] introduce embedding learning for the semantic retrieval in their E-commerce search service. Besides, Yin *et al.* [46] also adopts deep semantic matching in the core ranking module of Yahoo Search. Different from previous studies, we explore pre-trained language model with multi-stage training to better perceive the semantics behind queries and documents.

Despite the above-mentioned bi-encoder models, *interaction-based methods* are also widely used in many information retrieval systems [9, 10, 44, 48–52]. As such, another line of research for semantic matching is to model query-document *interaction* with DNNs [25, 28, 40, 44, 53]. However, they cannot cache the document embeddings offline, and thus are inefficient for retrieval. They are preferred for ranking stage, which will not be further discussed in this paper. In our search engine, interaction-based methods are exploited to build the PLM-based ranking system [53].

2.2 Pretrained Language Models

Pretrained Language Models (PLMs), such as ELMo [31], BERT [4] and ERNIE [38], have achieved monumental success in natural language understanding. Notably, the recent state-of-the-art PLMs [4, 22, 43] are usually based on Transformers [39], which exploit a deep structure with stacked multi-head attention and fully-connected layers. More importantly, they adopted unsupervised pretraining with large corpus, and thus can incorporate more useful knowledge in the models. As BERT and its successors (e.g., XLNet [43], RoBERTa [22]) exhibit superior capacity in understanding textual data, a handful of studies start to leverage them for semantic retrieval [2, 16, 24]. For instance, Chang *et al.* [2] propose an early attempt that introduces BERT-based bi-encoders for large-scale retrieval. It also studies the effects of several new pretraining tasks. Humeau *et al.* [16] advance such bi-encoder with attentive interaction for the query and document embeddings. Lu *et al.* [23] explore the negative sampling strategies for BERT-based bi-encoders at

both pretraining and finetuning stages. However, despite the initial success achieved by these work, there is still a lack of investigation on developing and deploying such model for web-scale retrieval, especially in real-world productions. In this paper, we propose to leverage the state-of-the-art Chinese pretrained language model, namely Enhanced Representation through kNowledge IntEgration (ERNIE) [38], for building the retrieval system of our search engine.

3 RETRIEVAL MODEL

In this section, we first present the task definition, and then introduce the details of our proposed retrieval model.

3.1 Task Definition

Given a query q, the web retrieval task aims to return a set of relevant documents from a large corpus D, where the size of the corpus can be tens of billions or more. We denote by D_q^+ the set of documents relevant to the query. Different from conventional retrieval with term matching, *semantic retrieval* computes the relevance score $f(\cdot)$ in a learned embedding space [2, 18, 19, 26, 42], using similarity metrics (e.g., dot-product, cosine similarity) as:

$$f(q, d) = \text{sim}(\phi(q; \theta), \psi(d; \beta)), \tag{1}$$

where ϕ (or ψ) is the representation model (i.e., encoder) parameterized by θ (or β) that encodes the query (or document) to dense embeddings.

3.2 Model Architecture

In this work, we are interested in parameterizing the encoders ϕ and ψ as deep Transformer models [39] due to its expressive power in modeling natural language.

The bi-encoder backbone. By implementing ϕ and ψ as deep Transformers, we refer the paradigm described in Eq. (1) as bi-encoder [16]. More specifically, let $\mathcal{T}$ be a multi-layer transformer encoder, which is a stack of n transformer blocks. Each block consists of a multi-head self-attention (MHA) sublayer followed by a feed-forward (FFN) sublayer, where MHA allows the model to jointly attend to different subspaces and FFN aggregates the attention results of different heads. The encoder takes $\{[\text{CLS}], T_1, \cdots, T_N\}$ (or $\{[\text{CLS}], T'_1, \cdots, T'_M\}$), i.e., the tokenzied sequence of the raw query (or document) text, as the input and outputs an encoded sequence $C, F_1, \cdots, F_N$ (or $C', F'_1, \cdots, F'_M$) as described in Figure 1. Note that [CLS] is a pseudo token that aggregates information in the encoder for the subsequent matching. For each transformer layer, the parameters are shared between query and document encoders (i.e., ϕ and ψ), which has multiple potential benefits, such as reducing the number of model parameters so as to control model complexity [6], introducing prior knowledge to regularize models [41], and saving the storage space or memory size [17]. The conventional bi-encoder computes the relevance score with a simple dot-product between the output CLS embeddings, i.e., $f(q, d) = C \cdot C'$.

Poly attention. Different from vanilla bi-encoder, we further develop a poly attention scheme that enables more flexibility for modeling query semantics. This idea is originated from poly-encoder [16] and is slightly customized in our retrieval model, which works differently during training and prediction phases.

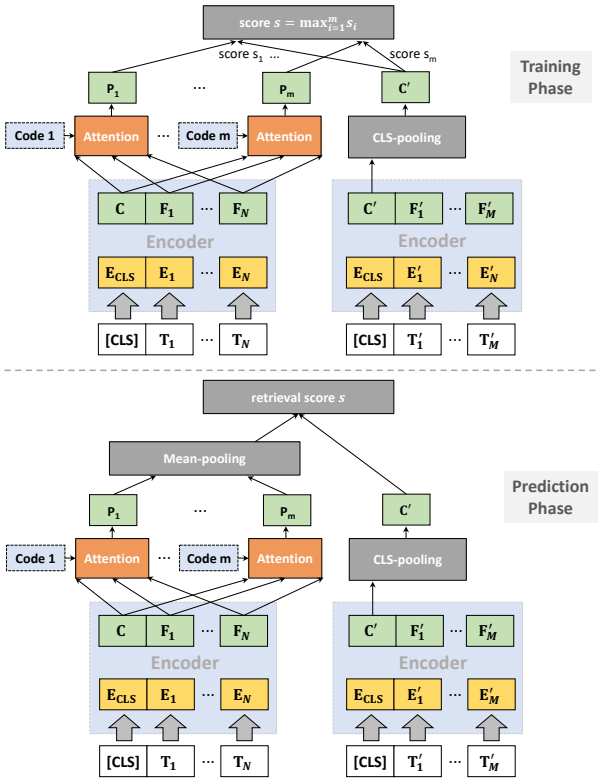

Figure 1: Model architectures and relevance computation.

In the training phase, as shown in the upper half part of Figure 1, we introduce a set of m context codes, i.e., $c_0, \cdots, c_{m-1}$, where each c_i extracts a global representation P_i by attending over all the outputs of the query encoder (i.e., $C, F_1, \cdots, F_N$) as

$$P_i = w_0^{(i)} \cdot C + \sum_{j=1}^{N} w_j^{(i)} \cdot F_j, \tag{2}$$

where $\left(w_0^{(i)}, \cdots, w_N^{(i)}\right) = \text{Softmax}\left(c_i \cdot C, \cdots, c_i \cdot F_N\right)$. These global context features $P_1, \cdots, P_m$ can be interpreted as different aggregations of the semantics from all the query terms, which reflects fine-grained query demands. Subsequently, each of the m global context features is compared with the document representation C' using dot-product, followed by a max-pooling to finalize the relevance score, i.e.,

$$f(q, d) = \max_{i=1}^{m} P_i \cdot C'. \tag{3}$$

In the retrieval/prediction phase, we employ a slightly different strategy due to a practical concern. In real application with massive web documents, document representations must be pre-computed to build indexes, which enable efficient retrieval with fast nearest neighbor search. In such case, relevance computation based on Eq. (3) is clearly infeasible, as such metric with max-pooling can hardly be supported by existing index schemes. Motivated by this, we propose to construct an unified surrogate embedding with mean pooling over all P_i, i.e., $\frac{1}{m} \sum_{i=1}^{m} P_i$. As such, the final relevance score

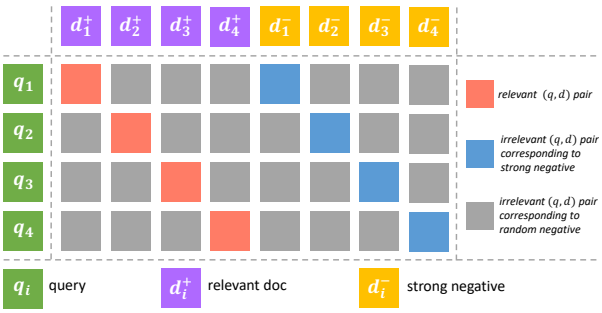

Figure 2: Illustration of negative sampling during training.

is defined as

$$f(q, d) = P_{avg} \cdot C' = \left(\frac{1}{m}\sum_{i=1}^{m} P_i\right) \cdot C'. \tag{4}$$

With such definition, index-based nearest neighbor search can be leveraged, which largely improves the applicability of the model.

Remark. As already mentioned, the relevance score is calculated inconsistently between training and predicting phases. In practice, we find out that such inconsistency would not undermine the model performance for semantic retrieval. Therefore, through bagging distinct features learnt by the context codes, we can achieve a better and more heterogeneous representation of the query for a more powerful semantic matching with that of the document.

3.3 Optimization

We formulate the learning procedure of the retrieval model using maximum likelihood estimation: Given a query q, all of its relevant documents D_q^+ and irrelevant ones D_q^-, the optimal parameters θ^*, β^* can be defined as:

$$\theta^*, \beta^* = \arg\max_{\theta,\beta} \sum_{q} \sum_{d^+ \in D_q^+} p(d^+|q, d^+, D_q^-), \tag{5}$$

where $p(d^+|q, d^+, D_q^-)$ is the probability that f can separate a relevant document d^+ from all irrelevant ones D_q^-. Note that we consider the context codes c_i are the parameters of the query encoder, which are jointly trained.

We implement $p(d^+|q, d^+, D_q^-)$ as a contrastive probability, i.e.,

$$p(d^+|q, d^+, D_q^-) = \frac{\exp\left(f(q, d^+)/\tau\right)}{\exp\left(f(q, d^+)/\tau\right) + \sum_{d^- \in D_q^-} \exp\left(f(q, d^-)/\tau\right)}, \tag{6}$$

where τ, which is normally set to 1, is the temperature of the softmax operation [12]. Using a higher value for τ produces a softer probability distribution over classes (i.e., relevant or irrelevant).

Furthermore, as it is inapplicable to know and use all the real relevant and irrelevant documents in the training, we reformulate the optimization problem with sampled relevant and irrelevant documents (respectively denoted as $\hat{D}_q^+$ and $\hat{D}_q^-$) as

$$\theta^*, \beta^* = \arg\max_{\theta,\beta} \sum_{q} \sum_{d^+ \in \hat{D}_q^+} p(d^+|q, d^+, \hat{D}_q^-). \tag{7}$$

3.4 Training Data Mining

To solve the optimization problem as Eq. (7), the key of model training lies on the construction of $\hat{D}_q^+$ and $\hat{D}_q^-$, such that a better model can be learned for semantic retrieval. In this subsection, we elaborate how we mine the positive and negative data for training.

Positives and negatives in different data sources. For mining positive and negative examples, we first consider two kinds of data sources that are commonly used in practice:

- *Search log data*, where the queries and documents are logged with click signal, i.e., whether the document is clicked by the user. For each query, we use those clicked results as positives and those exposed but non-click results as negatives, as clicks can roughly indicate the satisfaction of user's intent.
- *Manually labeled data* is usually collected with fine-grained grades (i.e., 0 to 4) assigned by human evaluation. For each query, the positive and negative examples are defined in a pairwise manner. For a given document that is considered as positive, other lower-grade documents under the same query can be considered as negatives.

In-batch negative mining. The goal of online retrieval is to distinguish a tiny fraction of relevant documents from massive irrelevant documents (i.e., $|D_q^+| \ll |D_q^-| \approx |D|$). Instead of logging (or labeling) many irrelevant documents, we leverage an in-batch negative mining scheme [18], which is a more efficient way to construct those completely irrelevant documents. To avoid confusion, we refer to the non-click (or lower-grade) samples as *strong negatives*, and the in-batch negatives as *random negatives*.

Particularly during the training of our model, we consider random negatives as those documents (i.e., positives and strong negatives) of other queries in the same mini-batch. As data are shuffled before fed to the model, the random negatives are generally quite distinct from the query, which helps mimic the online retrieval scenario, i.e., identifying positives from massive random negatives. In addition, to collect sufficient random negatives from each mini-batch, we exploit a simple yet practical solution, i.e., increasing the batch size. We apply mix-precision method that allows larger batch size during training with a fixed memory consumption. The batch size is set to make maximum use of GPU memory.

Figure 2 shows an example of the overall negative sampling strategies in a more intuitive way. In the example, the mini-batch of size 4 consists of four triplets $\{(q_i, d_i^+, d_i^-), i = 1, 2, 3, 4\}$. For the query q_1, the positive and strong negative documents are respectively denoted as d_1^+ and d_1^-, and $d_2^+, d_3^+, d_4^+, d_2^-, d_3^-, d_4^-$ are random negatives. For each row (i.e., each query) in Figure 2, there is one positive and a set of negatives, which can be used as d_+ and D_q^- in Eq. (7). Therefore, the training data mined with these strategies can be used to optimize the retrieval model based on Eq. (7).

4 TRAINING PARADIGM

The training paradigm of pretraining and finetuning has been widely employed for model optimization in many natural language processing problems. Representations learned by such paradigm usually show competitive performance across many tasks. However, such superiority has not yet been fully exploited for web retrieval. In this section, we propose a novel multi-stage training paradigm. Particularly, as shown in Figure 3, we divide the entire training

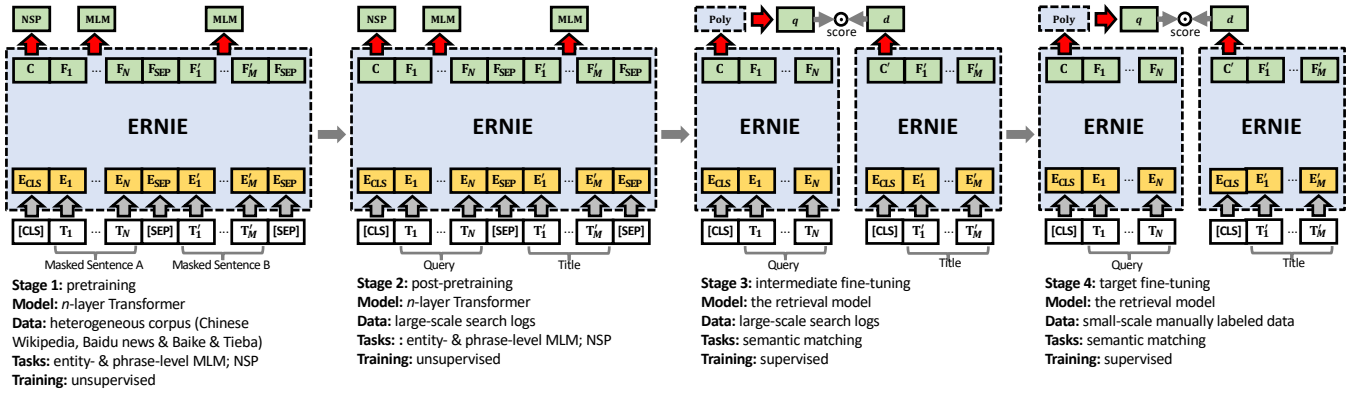

Figure 3: Training Paradigm of ERNIE-based Retrieval Model.

process into four stages: (1) **pretraining**, (2) **post-pretraining**, (3) **intermediate fine-tuning**, and (4) **target fine-tuning**. Each stage is carefully designed with different training data and objectives, which boost the generalization ability of the model. The overall training pipeline is shown in Figure 3.

4.1 Stage 1: Pretraining

In this stage, we follow the same pretraining process in [38] to train an ERNIE encoder (i.e., an n-layer transformer). It adopts heterogeneous corpus, i.e., Chinese Wikipedia, Baidu Baike (containing encyclopedia articles written in formal languages), Baidu news and Baidu Tieba (an open discussion forum like Reddits), to learn language representation enhanced by entity- and phrase-level masking strategies. The heterogeneous corpus contributes a large proportion of web documents in our search engine, hence the pretrained ERNIE model can effectively transfer knowledge to web retrieval. The pretraining is unsupervised, where the tasks include masked word prediction (i.e., MLM - Masked Language Modeling) and the next sentence prediction (NSP).

4.2 Stage 2: Post-pretraining

Search engine processes billions of various queries with diverse goals or intentions every day, such as medical advice, travel guides, latest news, etc. Pretraining only on document corpus might be limited to tackle all kinds of requests from users. In this stage, we transfer previous pretrained model and continue pretaining on both query and document corpus of Baidu Search with the same tasks (i.e., MLM and NSP) as Stage 1.

We name Stage 2 as post-pretaining, which keeps the model structure (i.e., a n-layer Transformer) and the training tasks while changes the training data. Specifically, we collect one-month (i.e., tens of billions of) user search logs for post-pretraining. The tokenized raw texts of the query and the title of the document are concatenated as the input during this stage. Then we apply the same masking strategies as ERNIE for MLM and take clicked documents as the next sentence of the query for NSP.

4.3 Stage 3: Intermediate Fine-tuning

From this stage, we start to fine-tune the retrieval model. We first fine-tune the retrieval model on an intermediate task, before fine-tuning again on the target task of interest [32]. In particular, we leverage the post-pretrained ERNIE produced by Stage 2, as the encoders of our retrieval model, which is subsequently optimized using the same training data (i.e., search logs) as Stage 2. The training objective is as presented in Eq. (7), and we construct positives and negatives for training as introduced in Section 3.4. Using an intermediate objective that more closely resembles our target task leads to better and faster fine-tuning performance.

4.4 Stage 4: Target Fine-tuning

Finally, we fine-tune our retrieval model produced by Stage 3 on a relatively small manually-labeled data, which we consider is the most accurate and unbiased data for learning the retrieval model. For the data collection, a set of queries are sampled from query logs, and a certain number of query-document pairs are labeled according to their relevance judged by human editors. A 0-4 grade is assigned to each query-document pair based on the degree of relevance. The training objective is as presented in Eq. (7), and we construct positives and negatives for training as introduced in Section 3.4.

5 DEPLOYMENT

In this section, we first introduce the embedding compression and quantization, which saves the online cost for retrieval. Then, we present the general picture of how the retrieval model works in the retrieval system. For the detailed implementation of our deployed model, please refer to the Appendix.

5.1 Embedding Compression and Quantization

We apply compression and quantization for the output embeddings of the retrieval model. On the document side, it can significantly reduce the memory cost for caching the embeddings; On the query side, it saves the transmission cost for query embedding, and thus improves the system throughput.

Compression. We jointly train a compression layer with the ERNIE encoders. The compression layer is implemented as a fully-connected

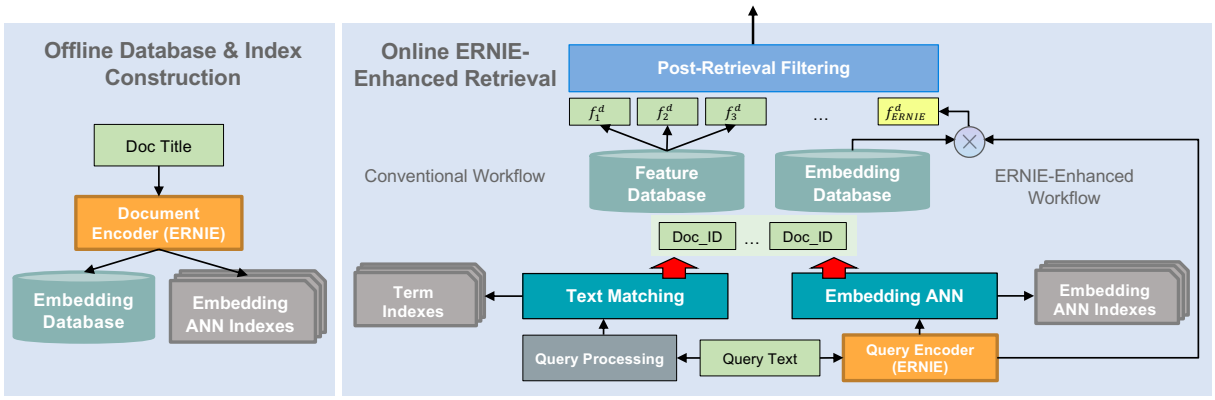

Figure 4: The overall workflow of the ERNIE-enhanced retrieval system.

layer that takes the output embeddings of the encoders (i.e., P_i and C') as inputs, and produces lower-dimensional embeddings. This would largely save the memory cost, with very slight decrease w.r.t. accuracy. In practice, we reduce the size of output embeddings from 768 to 256, which improves the overall efficiency of transmission and storage by 3 times.

Quantization. Quantization is a very promising technique for boosting the efficiency and scalability of neural networks [15]. For deploying our model to efficiently serving massive search queries, we apply a commonly-used int8 quantization to the output embeddings, where the overall efficiency can further be boosted by 4 times. More details of the embedding quantization can be referred to the Appendix.

5.2 System Workflow

An important fact for building an effective retrieval system is that, neither explicit text matching nor embedding-based semantic matching can handle all the various kinds of queries. Instead, it is more promising to integrate the two types of retrieval methods together, to provide a better overall performance. Motivated by this, we developed a novel system workflow, which is depicted in Figure 4. Compared with conventional text matching, the system is upgraded with two modules:

- *ERNIE-based retrieval.* The system enables the ERNIE-based retrieval model to work in parallel with the conventional retrieval workflow (i.e., text matching), which offers high maintainability of the system.
- *ERNIE-enhanced post-retrieval filtering.* We further introduce an unified post-retrieval filtering module. It takes both retrieved documents from text matching and ERNIE-based semantic matching, and conduct effective post-retrieval filtering for further discovering high-quality candidates.

We elaborate the system workflow with the two modules, which including both offline and online stages.

Offline database and index construction. As shown in the left part of Figure 4, during the offline stage, we use the trained ERNIE encoder to compute the embeddings (with the compression and quantization) for all documents in the corpus, based on which we build 1) Approximate Nearest Neighbor (ANN) indexes, and 2) an embedding database for the above-mentioned two modules, respectively. In particular, the ANN indexes are deployed for the ERNIE-based retrieval stage, to enable efficient embedding-based candidate generation via fast nearest neighbor search. The embedding database is used for the post-retrieval filtering stage. It is a key-value database for efficient document embedding lookup with given document ids. Next, we introduce the online workflow and show how the database and indexes are used online.

Online integrated retrieval. For the online retrieval, we integrate conventional retrieval and the novel ERNIE-based semantic retrieval. The conventional retrieval first processes the query text with several operations (e.g., word segmentation, stop-word filtering), and then conducts keyword/term matching with term indexes (i.e., inverted indexes) to retrieve a set of documents. In parallel, semantic retrieval first fetch the query embedding. After online embedding compression and quantization, the query embedding is applied for semantic retrieval with the ANN indexes. The results produced by different retrieval approaches are merged to form a candidate pool.

Online post-retrieval filtering. After the integrated retrieval, we further introduce an online post-retrieval filtering stage, which further narrows down the scale of the candidates. To achieve this, the filtering module contains a lightweight ranking model (e.g., GBRank or RankSVM) that conducts an unified comparison for the retrieval candidates. Particularly for each candidate document, we fetch the statistical features (denoted as $f_1^d, f_2^d, f_3^d, \ldots$ in Figure 4), such as click-through rate, from the feature database. Note that the semantic matching score is also applied in this stage as an important feature (denoted as f_{ERNIE}^d), where the scores computed in semantic retrieval can be reused. For those candidates retrieved by text matching, where the semantic matching scores are missing, we fetch their embeddings from the embedding database, and then compute the scores with the query embedding. As the candidate pool formed by the online retrieval stage is small compared to the entire corpus, the online computation of the semantic matching scores does not significantly increase the time cost. By doing this, we unify the comparison of all candidate documents with several statistical features and a semantic feature, where the semantic modeling capacity of our retrieval model can also benefit the documents dug by conventional retrieval paradigm.

6 OFFLINE EVALUATION

We conduct an offline evaluation for the proposed retrieval model. The implementation details of the model and more ablation study can be found in the Appendix.

6.1 Datasets

We evaluate the retrieval model on the datasets that are collected from the real production environment. Note all the data does not contain any user-related information for the privacy concern.

Manually-labeled dataset (Manual). This dataset contains 6,423 queries and 147,341 query-document pairs. Each query has around 22 retrieved results on average. For each query, we collect documents from each stage of the search pipeline to ensure the diversity of the dataset. The dataset is labeled on our crowdsourcing platform, where a group of experts are required to assign an integer score varies from 0 to 4 to each query-document pair. The score represents whether the content of the document w.r.t. the query is off-topic (0), slightly relevant (1), relevant (2), useful (3) or vital (4).

Automatic-annotated datasets (Auto). We collect another two search log datasets, namely Auto-Rand and Auto-Tail, which contains random queries and tail queries, respectively. Here, the tail queries are identified as those who have no more than 10 searches per week. For each query, we use our search engine to collect the top-10 results in the final ranking stage as the positive examples (i.e., the relevant documents), and consider the top-10 results of other queries as negative examples. After filtering noise and meaningless queries, we finalize the two datasets, where Auto-Rand contains 18,247 queries and 112,091 query-document pairs, and Auto-Tail contains 78,910 queries and 750,749 query-document pairs.

6.2 Evaluation Metrics

Positive-Negative Ratio. We report the positive-negative ratio (PNR) on Manual dataset. For a given query q and the associated documents D_q, the PNR can be formally defined as

$$PNR = \frac{\sum_{d_i,d_j \in D_q} \mathbb{1}(y_i > y_j) \cdot \mathbb{1}(f(q,d_i) > f(q,d_j))}{\sum_{d_{i'},d_{j'} \in D_q} \mathbb{1}(y_{i'} > y_{j'}) \cdot \mathbb{1}(f(q,d_{i'}) < f(q,d_{j'}))}, \quad (8)$$

where $\mathbb{1}(\cdot)$ is an indicator function, i.e., $\mathbb{1}(a > b) = 1$ if $a > b$, and 0 otherwise. PNR measures the consistency between the manual labels and the model scores. We report the averaged PNR values over all queries in the experiments.

Recall@10. We report Recall@10 on Auto-Rand and Auto-Tail datasets. The metric is defined as $Recall@10 = |D_q \cap \tilde{D}_q|/10$, where $\tilde{D}_q$ represents the top-10 retrieval w.r.t. q using the retrieval model considering all the documents in a dataset as the corpus, and D_q is the set of collected ground truth. We report the averaged Recall@10 values over all queries in the experiments.

6.3 Offline Experimental Results

In the offline evaluation, we report the experimental results of the proposed model during different training stages. Here, the performance of pretraining and post-pretraining are reported when using the n-layer Transformer in a bi-encoder model. Besides, we also include a baseline method, i.e., the model that is used in the system before deploying the ERNIE-base retrieval model.

Table 1 shows the results, where some key findings are observed:

Table 1: Offline experimental results on different stages.

Dataset	Manual	Auto-Rand	Auto-Tail
Metric	PNR	Recall@10	Recall@10
Pretrain	1.34	18.92%	12.10%
Post-pretrain	1.43	35.99%	19.10%
Intermediate Fine-tune	2.00	83.14%	60.98%
Target Fine-tune	**2.48**	**87.90%**	**71.13%**
Online baseline	1.77	85.22%	65.79%

- The performance on different stages indicate that our proposed training paradigm is able to gradually improve the performance of the retrieval model. In particular, we can see that the model in the pretrain stage does not contain any domain knowledge w.r.t. the retrieval tasks, and thus performs poorly, where the Recall@10 is below 20% on both Auto datasets. However, after applying the multi-stage training paradigm, the Recall@10 of the model finally reaches 87.90% and 71.13% on Auto-Rand and Auto-Tail, respectively.
- By applying the multi-stage training, the retrieval model can outperforms our online baseline w.r.t. Recall@10 on both Auto datasets. Moreover, we also observe that the relative improvement of our proposed model over the online base is large on Auto-Tail ($\Delta = 8.1\%$) than Auto-Rand ($\Delta = 3.1\%$), in terms of Recall@10. This shows that our proposed method can better discover relevant results for tail queries, which is very helpful in improving the user experience for the search engine.
- In addition, we can see that the proposed model can beat the online baseline by a large margin w.r.t. PNR, where the value is improved from 1.77 to 2.48. This tells us that the proposed model not only can retrieval relevant documents, but also prefer high-quality results, i.e., the documents with higher manually-labeled grades.

Overall, our proposed model is able to gain superior performance on semantic retrieval through the multi-stage training, and beat the online baseline by a significant margin.

7 ONLINE EVALUATION

To investigate the impact of our proposed system to the search engine, we deploy the new system and conduct online experiments to compare it with the old retrieval system.

7.1 Interleaved Comparison

Interleaving [3] is a commonly-used technique for evaluating industrial information retrieval systems (e.g., recommender systems, search engines). In interleaved comparison, the results of two systems are interleaved and exposed together to the end users, whose click behaviors would be credited to the system that provides the clicked results. The gain of the new system A over the base system B can be quantified with Δ_{AB}, which is defined as

$$\Delta_{AB} = \frac{wins(A) + 0.5 * ties(A,B)}{wins(A) + wins(B) + ties(A,B)} - 0.5, \quad (9)$$

where $wins(A)$ (or $wins(B)$) is a counter that would be increased by 1 if the results produced by A (or B) is preferred by the user, and $ties(A,B)$ is increased by 1 otherwise. Intuitively, $\Delta_{AB} > 0$

Table 2: Interleaved comparison.

	ENINE-based retrieval				Post-retreival filtering			
	Query type		Query length		Query type		Query length	
	Rand	Tail (freq < 3)	Short (≤10)	Long (10 <)	Rand	Tail (freq < 3)	Short (≤10)	Long (10 <)
Δ_{AB}	+0.368%	+0.992%	+0.345%	+0.426%	+0.112%	+0.274%	+0.066%	+0.191%
Δ_{AB}-tw	+0.281%	+0.783%	+0.253%	+0.352%	+0.226%	+0.454%	+0.176%	+0.315%

*All the values are statistically significant (t-test with $p < 0.05$).

Table 3: Relative improvement on manual evaluation.

	ERNIE-based retrieval		Post-retrieval filtering	
	Rand	Tail	Rand	Tail
Δ_{DCG}	+0.17%	+0.22%	+0.10%	+0.65%
Δ_{GSB}	+3.50%	+7.50%	+3.96%	+3.13%

*All the values are statistically significant (t-test with $p < 0.05$).

means A is better than B. To further reduce the bias in the evaluation, we further introduce a time-weighted version of Δ_{AB}, denoted as Δ_{AB}-tw, where the counters are weighted by the post-click dwell time (mapped to $[0, 1]$ with a sigmoid function). As such, the Δ_{AB}-tw can better reflect the relevance of the results with higher confidence, as users would stay in a web page longer if it is relevant.

We conduct balanced interleaving experiments [3] for comparing the ERNIE-enhanced system against the old retrieval system. The results are shown in Table 2, which comprise the performance of different modules and for different types of queries.

- First, we observe $\Delta_{AB} > 0$ and Δ_{AB}-tw > 0 in all the experiments, which indicate that the new system can increase user clicks with more relevant web documents.
- Second, the results also indicate that both the ERNIE-based retrieval and post-retrieval filtering can boost the effectiveness of the system, while the ERNIE-based retrieval generally achieves larger gain than the post-retrieval filtering, as it is effected before the filtering stage.
- Third, we can see that the new system can achieve better performance on the queries with low search frequency, i.e., tail query. This validates that our proposed retrieval system has more significant improvement for the tail queries.
- Also, the improvements on long queries (e.g., natural language queries) is larger than on short queries, which might indicate that the system can better handle the natural language queries.

7.2 Online A/B Test

We also conduct online A/B Test that compares the new system with the old system for one week. In the online A/B test, We mainly focus on the metrics that directly related to user experience. The results show that the proposed retrieval system can largely improve the overall user experience of the search engine. In particular, the number of click behaviors has increased by 0.31%. The number of click-and-stay behavior has increased by 0.57%. The average post-click dwell time has increased by 0.62%. The click-through rate has increased by 0.3%. All the reported values are statistically significant with $p < 0.05$. This shows that accurate semantic modeling and semantic matching in the retrieval stage is very helpful for improving the user engagement for the search engine.

7.3 Manual Evaluation for Online Cases

To more comprehensively show the impact of our proposed system, we further conduct a manual evaluation on the final ranking results with some real user-generated queries. This directly reflects the quality of the results exposed to the end users.

Data preparation. We log a set of (several hundreds) queries and the corresponding final impressions, i.e., the top-ranked web documents in the final ranking stage, using the ERNIE-enhanced and the old retrieval systems. Note that the data logging is conducted by multiple rounds to eliminate randomness. We filter out whose queries that have identical impressions between the two systems, and then use the rest for the manual evaluation. The relative improvement validated by manual evaluation is given in Table 3.

Discounted Cumulative Gain (DCG). We first log a dataset and manually label the data with 0 to 4 grades, and then report the relative improvement w.r.t. the average Discounted Cumulative Gain (DCG) over the top-4 final results of all queries). The DCG is a widely-used metric and thus we omit its definition here. As shown in Figure 3, the results again show that our proposed system is able to improve the effectiveness of retrieval, especially for tail queries.

Side-by-side comparison. Besides, we also conduct a side-by-side comparison between the two systems. We log another dataset, and require the human experts to judge whether the new system or the base system gives better final results. Here, the relative gain is measured Good vs. Same vs. Bad (GSB) as

$$\Delta_{GSB} = \frac{\#\text{Good} - \#\text{Bad}}{\#\text{Good} + \#\text{Same} + \#\text{Bad}}, \tag{10}$$

where #Good (or # Bad) indicates the number of queries that the new system provides better (or worse) final results. Table 3 shows that for both random queries and tail queries, the new ERNIE-enhanced system can significantly outperform the base system.

8 CONCLUSION

In this paper, we described the novel retrieval system that is facilitated by pretrained language model (i.e., ERNIE). We developed and deployed the system in Baidu Search, which is highly effective in conducting semantic retrieval for web search. The system employs 1) an ERNIE-based retrieval model, 2) a multi-stage training paradigm and 3) a unified workflow for the retrieval system. Extensive offline and online experiments has shown that the retrieval system can significantly improve the effectiveness and general usability of the search engine.

REFERENCES

[1] Yinqiong Cai, Yixing Fan, Jiafeng Guo, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2021. Semantic Models for the First-stage Retrieval: A Comprehensive Review. *arXiv preprint arXiv:2103.04831* (2021).

[2] 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).

[3] Aleksandr Chuklin, Anne Schuth, Ke Zhou, and Maarten De Rijke. 2015. A comparative analysis of interleaving methods for aggregated search. *TOIS* 33, 2 (2015), 1–38.

[4] 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).

[5] Xingping Dong and Jianbing Shen. 2018. Triplet loss in siamese network for object tracking. In *ECCV*. 459–474.

[6] Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016. Zero-Resource Translation with Multi-Lingual Neural Machine Translation. In *EMNLP*. 268–277.

[7] Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. 2014. Modeling interestingness with deep neural networks. (2014).

[8] Jianfeng Gao, Kristina Toutanova, and Wen-tau Yih. 2011. Clickthrough-based latent semantic models for web search. In *SIGIR*. 675–684.

[9] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, Lixin Zou, Yiding Liu, and Dawei Yin. 2020. Deep Multifaceted Transformers for Multi-objective Ranking in Large-Scale E-commerce Recommender Systems. In *CIKM*. 2493–2500.

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

[11] Malay Haldar, Prashant Ramanathan, Tyler Sax, Mustafa Abdool, Lanbo Zhang, Aamir Mansawala, Shulin Yang, Bradley Turnbull, and Junshuo Liao. 2020. Improving Deep Learning For Airbnb Search. In *KDD*. 2822–2830.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[13] 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 *KDD*. 2553–2561.

[14] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. 2333–2338.

[15] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *JMLR* 18, 1 (2017), 6869–6898.

[16] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969* (2019).

[17] Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *TACL* 5 (2017), 339–351.

[18] 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.

[19] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *ACL*. 6086–6096.

[20] Hang Li and Jun Xu. 2014. Semantic matching in search. *Foundations and Trends in Information retrieval* 7, 5 (2014), 343–469.

[21] Yiding Liu, Yulong Gu, Zhuoye Ding, Junchao Gao, Ziyi Guo, Yongjun Bao, and Weipeng Yan. 2020. Decoupled Graph Convolution Network for Inferring Substitutable and Complementary Items. In *CIKM*. 2621–2628.

[22] 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).

[23] Jing Lu, Gustavo Hernandez Abrego, Ji Ma, Jianmo Ni, and Yinfei Yang. 2020. Neural Passage Retrieval with Improved Negative Contrast. *arXiv preprint arXiv:2010.12523* (2020).

[24] Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. TwinBERT: Distilling Knowledge to Twin-Structured Compressed BERT Models for Large-Scale Retrieval. In *CIKM*. 2645–2652.

[25] Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. *ç* 26 (2013), 1367–1375.

[26] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval. *arXiv preprint arXiv:2005.00181* (2020).

[27] Bhaskar Mitra, Nick Craswell, et al. 2018. *An introduction to neural information retrieval*. Now Foundations and Trends.

[28] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *WWW*. 1291–1299.

[29] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and R Ward. 2014. Semantic modelling with long-short-term memory for information retrieval. *arXiv preprint arXiv:1412.6629* (2014).

[30] Hamid Palangi, H Palangi, L Deng, Y Shen, J Gao, X He, J Chen, X Song, and R Ward. 2015. Deep Sentence Embedding Using the Long Short Term Memory Network: Analysis and Application to Information Retrieval. arXiv. org.

[31] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[32] Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel Bowman. 2020. Intermediate-Task Transfer Learning with Pretrained Language Models: When and Why Does It Work?. In *ACL*. 5231–5247.

[33] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *IJAR* 50, 7 (2009), 969–978.

[34] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge.

[35] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*. 373–382.

[36] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*. 101–110.

[37] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW*. 373–374.

[38] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (2019).

[39] 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 *NeurIPS*. 5998–6008.

[40] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI*, Vol. 30.

[41] Yingce Xia, Xu Tan, Fei Tian, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2018. Model-level dual learning. In *International Conference on Machine Learning*. PMLR, 5383–5392.

[42] 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).

[43] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237* (2019).

[44] Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained Transformers for Text Ranking: BERT and Beyond. In *WSDM*. 1154–1156.

[45] Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *CoNLL*. 247–256.

[46] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, et al. 2016. Ranking relevance in yahoo search. In *KDD*. 323–332.

[47] Han Zhang, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, and Wen-Yun Yang. 2020. Towards Personalized and Semantic Retrieval: An End-to-End Solution for E-commerce Search via Embedding Learning. *arXiv preprint arXiv:2006.02282* (2020).

[48] Xiangyu Zhao, Haochen Liu, Hui Liu, Jiliang Tang, Weiwei Guo, Jun Shi, Sida Wang, Huiji Gao, and Bo Long. 2020. Memory-efficient Embedding for Recommendations. *arXiv preprint arXiv:2006.14827* (2020).

[49] Xiangyu Zhao, Chong Wang, Ming Chen, Xudong Zheng, Xiaobing Liu, and Jiliang Tang. 2020. Autoemb: Automated embedding dimensionality search in streaming recommendations. *arXiv preprint arXiv:2002.11252* (2020).

[50] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *KDD*. 2810–2818.

[51] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. 2020. Pseudo Dyna-Q: A reinforcement learning framework for interactive recommendation. In *WSDM*. 816–824.

[52] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Jimmy Xiangji Huang, and Dawei Yin. 2020. Neural Interactive Collaborative Filtering. In *SIGIR*. 749–758.

[53] Lixin Zou, Shengqiang Zhang, Hengyi Cai, Dehong Ma, Suqi Cheng, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained Language Model based Ranking in Baidu Search. In *KDD*.

```
def model_encode_query(tokens):
  all_embeds = ERNIE_encoder.get_all_outputs(tokens)
  poly_embeds = poly_attention(all_embeds, context_codes)
  return [fc_compression(poly_embeds[i]) for i in range(m)]


def model_encode_doc(tokens):
  cls_embed = ERNIE_encoder.get_cls_output(tokens)
  return fc_compression(cls_embed)


def train_interaction(q, pos, neg):
  all_logits1, all_logits2 = [], []
  for i in range(m):
    # in-batch random negative sampling via matrix multiplication
    pos_logits_with_rand_neg = matmul(q[i], pos.T)
    neg_logits_with_rand_neg = matmul(q[i], neg.T)
    all_logits1.append(pos_logits_with_rand_neg)
    all_logits2.append(neg_logits_with_rand_neg)
  max_logits1 = reduce_max(all_logits1)
  max_logits2 = reduce_max(all_logits2)
  final_logits = concat(max_logits1, max_logits2)
  loss = softmax_with_cross_entropy(final_logits)
  return loss


def predict_interaction(q, d):
  avg_q = reduce_mean(q)
  return reduce_sum(avg_q * d)


def train(query_tokens, pos_doc_tokens, neg_doc_tokens):
  query_embed = model.encode_query(query_tokens)
  pos_doc_embed = model_encode_doc(pos_doc_tokens)
  neg_doc_embed = model_encode_doc(neg_doc_tokens)
  loss = train_interaction(query_embed, pos_doc_embed, neg_doc_embed)
  apply_optimization(loss)


def predict(query_tokens, doc_tokens):
  query_embed = model_encode_query(query_tokens)
  doc_embed = model_encode_doc(doc_tokens)
  score = predict_interaction(query_embed, doc_embed)
  return score
```

Figure 5: Pseudo code of the model training and prediction.

9 APPENDIX

9.1 Implementation Details

Computation resources. We implement the proposed retrieval model with PaddlePaddle (version 1.6.2) [2], an open-source library for developing and deploying deep learning models. The model is pretrained and finetuned with 32 and 8 NVIDIA V100 GPUs (32GB Memory), respectively, on our distributed training platform.

Parameter settings. We use 6-layer Transformers as the query and document encoders. In the input layer, we tokenize a given query or document title into Chinese characters as integer tokens, and map them into a set of embedding vectors with size 768. For each Transformer layer, we set the dimension of each hidden token representation as 768, the number of heads as 12, i.e., each head produces a 64 dimensional output. Besides, we set the number of context codes (i.e., m) as 16, and the dimension of the compression layers as 256. For model optimization, we use an Adam optimizer, and set the learning rate as 2e-5 and the batch size as 160 in all stages. For each training stage, we apply 4,000 warm up steps for the learning rate, and a 0.01 decay rate afterwards. During the training, we use a 0.1 dropout rate for all the layers to random drop the attention weights. Other unmentioned details are set as the same as vanilla ERNIE model.

[2]https://github.com/PaddlePaddle/Paddle

Table 4: Ablation study on the improvements of our model.

ID	Model	Search log	Manual data	Storage / doc
0	Base	2.15	1.76	3072 bytes
1	0 + Poly	2.23	1.81	3072 bytes
2	1 + IBN	2.33	2.06	3072 bytes
3	2 + Compression	2.26	2.00	1024 bytes
4	3 + Quantization	2.27	2.01	256 bytes

Table 5: PNR values when varying the scoring method.

Method	Search log	Manual data
$\max_{i=1}^{m} P_i \cdot C'$	2.205	1.955
$\left(\frac{1}{m}\sum_{i=1}^{m} P_i\right) \cdot C'$	2.205	1.987
$P_0 \cdot C'$	2.161	1.940
$P_1 \cdot C'$	2.176	1.969
$P_2 \cdot C'$	1.131	1.909
$P_3 \cdot C'$	2.204	1.976

Implementations. We present the pseudo code to depict the overall implementation of our method for the sake of reproducibility. Note that the the in-batch random negative samples are implemented in train_interaction() by matrix multiplications, which is very efficient in practice. Such implementation is also adopted by other related work (e.g., https://github.com/chijames/Poly-Encoder).

9.2 Details of Embedding Quantization

In particular, the output query and document embeddings are all quantized from float32 to uint8. According to output embeddings on a large-scale validation dataset, we calculate the data range (s_i^{min}, s_i^{max}) for each dimension i of the output embeddings. Then, we divide the data range into $L = 255$ equal intervals of Length Q_i, where $Q_i = (s_i^{max} - s_i^{min})/L$. For the value r_i in dimension i of a given output embedding, when performing quantization, its quantized index $QI_i(r_i)$ is calculated by

$$QI_i(r_i) = \lfloor (r_i - s_i^{min})/Q_i \rfloor.$$

This index is in range $[0, 255]$, which can be representation as a 8-bit integer. When performing online inference, we recover a quantized value $\tilde{r_i}$ by

$$\tilde{r_i} = QI_i(r_i) * Q_i + Q_i/2 + s_i^{min}$$

to approximate r_i. This quantization only causes a small loss of precision which is ignorable for inference, but achieves significant improvements on development efficiency. The quantitative experimental results can be found in Table 4. On the document side, this local quantization helps the system further save huge storage cost. On the query side, it significantly reduces the transmission cost, as the relevance computation for a given query would be distributed to multiple workers. Through the local quantization, the transmission and storage overheads further decrease to 1/4.

9.3 Offline Ablation Study

We further conduct several offline experiments to study the impact of some details of the retrieval model. For the experiments, we use

two validation datasets, i.e., a search log dataset and a manually-labeled dataset. The construction of the two types of datasets can be found in Section 3.4. The manually-labeled dataset is the same as we introduced in Section 6.1, and the search log data is isolated from the large-scale training data, which contains 300,000 query-document pairs. Note that all the following experiments are conducted for the model trained after intermediate finetuning (i.e., Stage 3).

9.3.1 Comparison with vanilla bi-encoder. Compared with vanilla ERNIE-based bi-encoder, our retrieval model is quite different, i.e., facilitated with poly attention mechanism, in-batch negative sampling strategy, and compression & quantization. To clarify the influence of each of these differences, Table 4 shows the offline experiments of how these distinct features are layered up in our model. Note that the base (i.e., vanilla ERNIE-based bi-encoder) is optimized with hinge loss, where the margin is set to 0.1. We can see from the table that 1) the poly-attention and in-batch negatives (denoted as IBN) can significantly improve the model performance on both datasets, 2) the compression would largely reduce the storage cost, but slightly sacrifice the effectiveness, 3) the quantization is shown to be loss-free w.r.t. the PNR metric on the offline datasets, which is very promising to be adopted online.

9.3.2 The training-prediction inconsistency. As mentioned in Section 3.2, we apply inconsistent schemes to finalize the output score during training and prediction. Here, we conduct experiments to investigate how the inconsistency would affect the performance of the model. Table 5 shows the PNR values of the model using different scoring methods on three extract validation datasets. Note that the model used here is an intermediate version, and thus the results might not be aligned with other experiments. In the table, the first row represents the scoring method used in training (i.e., Eq. (3)), the second row represents the scoring method used in prediction (i.e., Eq. (4)), and P_0 to P_4 indicate using fixed single global representation for the prediction, who are randomly sampled from the all 16 of them. We can see that our adopted mean-pooling method (i.e., the second row) performs similarly to the scoring method used in training. Thus, such inconsistency does not undermine the performance of the model during prediction.