Beyond Lexical: A Semantic Retrieval Framework for Textual Search Engine

Anonymous

Abstract

Search engine has become a fundamental component in various web and mobile applications. Retrieving relevant documents from the massive datasets is challenging for a search engine system, especially when faced with verbose or tail queries. In this paper, we explore a vector space search framework for document retrieval. Specifically, we trained a deep semantic matching model so that each query and document can be encoded as a low dimensional embedding. Our model was trained based on BERT architecture. We deployed a fast k-nearest-neighbor index service for online serving. Both offline and online metrics demonstrate that our method improved retrieval performance and search quality considerably, particularly for tail queries.

1 Introduction

Search engine has been widely applied in plenty of areas on the internet, which receives a query provided by users and returns a list of relevant documents within sub-seconds, helping users obtain their desired information instantaneously. Numerous technologies have been developed and utilized in real-world search engine systems(Yin et al., 2016). However, the existing semantic gap between search queries and documents, makes it challenging to retrieve the most relevant documents from tens of millions of documents. Therefore, there is still a large proportion of search requests that can not be satisfied perfectly, especially for long tail queries.

A search engine system is usually composed of three main modules,

- query understanding module
- retrieval module
- ranking module

The query understanding module first parses the original query string into a structured query object(Riezler and Liu, 2010). More specifically, the query understanding module includes several subtasks, such as word segmentation, query correction, term importance analyze, query expansion, and query rewrite, etc. After the query string was parsed, an index module accepts the parsed query, and then retrieve the candidate documents.

We call this stage the retrieval stage or the first round stage. Most web-scale search engine systems use the term inverted index for document retrieval, where $term$ is the most basic unit in the whole retrieval procedure. In the first round stage, the retrieved documents are ranked by a simple relevance model, eg TF-IDF, BM25, and the top-N documents with the highest score are submitted to the next stage for ranking. Finally, the documents scored largest by a ranking function are returned to users eventually.

For a search system described above, the final retrieval performance is highly enslaved by these query understanding module. Take word segmentation as an example: this task segments raw continuous query string into a list of segmented terms. Since the word segmentation algorithm has the risk of wrong segmentation. If the error segmented term does not appear in the document space, then no document could be retrieved in the first round stage, and it will return a result page without any document which damages the user's experience seriously.

There is a lot of work focused on better understanding queries to retrieve more relevant documents. However, since the final performance is influenced by all parts of the query understanding module. Attempts to optimize only one part is usually hard to contribute to a significant enhancement. To avoid the problems mentioned above, we propose a novel complementary retrieval system that retrieves documents without the traditional term-based retrieval framework. That is, instead of parse raw query into a structured query, we directly map both queries and documents into a low dimension of embedding. Then in the online serving, the k-nearest-neighbor documents of the given query in the latent embedding space are searched for retrieval.

Recently, we have witnessed tremendous successful applications of deep learning techniques in information retrieval circle, like query document relevance matching (Huang et al., 2013) (Shen et al., 2014b) (Shen et al., 2014a), query rewriting (He et al., 2016), and search result ranking(Haldar et al., 2018)(Grbovic and Cheng, 2018). However, it is still hard to directly retrieve relevant documents using an end2end fashion based on k-nearest-neighbor search in latent space, especially for long tail queries.

The latest far-reaching advancement in natural language processing with deep learning, BERT(Devlin et al., 2018), provides a turning point to make end2end retrieval realizable. In this paper, we present a document retrieval framework as a supplement to the traditional inverted index based retrieval system. We design a new architecture to retrieve documents without a traditional term-based query understanding pipeline, which avoids performance decay by each subtask of query understanding. We use BERT architecture as the general encoder of query and document strings, then we fine-tuned the pre-trained BERT model with human annotated data and negative sampling technique. Finally, we conduct both offline and online experiments to verify our proposed method. To sum up, our main contributions are described below:

1. We design a novel end2end document retrieval framework , which is a supplement to traditional term-based methods.
2. Our model is trained on transformer architecture, and a series of training techniques are developed for performance enhancement.
3. The proposed techniques can not only be used in document retrieval but also have a significant improvement for search ranking.

The rest of the paper is organized as follows. We concisely review the related work in Section 2. Sections 3 mainly describes our proposed methods. Offline and online experiments are detailed given in Section 4 and Section 5 respectively. Finally, we conclude and discuss future work in Section 6.

2 Related Work

2.1 query understanding

There is a variety of work on search query understanding(Prakash and Patel, 2014), including query correction(Chen et al., 2007), query term weighting(Zheng and Callan, 2015), query expansion(Azad and Deepak, 2017) and query reformulation(Buck et al., 2017). In general, these kinds of methods coherently rewrite the raw query into a new query, by replacing, adding, or removing terms or phrases in the raw query. The rewritten query gets better expression and therefore can retrieve more relevant documents than the original one.

2.2 knn approximate & text embedding

Besides the inverted index, vector search engines(Gionis et al., 1999) have also been widely applied in many information seeking tasks, like image search(Ji et al., 2014) and recommendation system(Covington et al., 2016).

To retrieve documents using a vector search, we need to map a piece of text into a low-dimensional numerical vector. Various embedding techniques have been developed and proven to have the powerful capability of capturing the semantic meaning of natural language text(Mikolov et al., 2013), (Pennington et al., 2014) (Kusner et al., 2015). However, these kinds of models are still not capable of complicating text encoding, especially for long tail text queries.

2.3 deep matching

More recently, researchers have been describing the various architecture of neural models(Mitra and Craswell, 2017). In text relevance matching area, we can divide most models into two typical categories, namely representation(Huang et al., 2013) based models and interaction based models(Pang et al., 2016b) (Xiong et al., 2017) (Dai et al., 2018). The representation models , like DSSM, are trained to obtain high-level representations of query and document respectively, then use vector distance between the query and document embedding for text relevance score. While the interaction based models first compute the term correlation matrix between query and documents and calculate semantic matching similarity based on the correlation matrix. Both representation models and interaction based models could be trained from massive click feedback data(Joachims, 2002)(Agichtein et al., 2006) or industrial annotation. These two kinds of model architecture are broadly deployed in real-world search engine systems, especially in ranking phase. For the representational models, once we obtained the high-level representation of raw texts, we can retrieve documents through the k-nearest-neighbor space search. However, the performance of representation based models are usually poorer than interaction based models, which makes k-nearest-neighbor retrieval hard to deploy in the real-world systems, since too many irrelevant documents retrieved may even damage overall performance.

(Zamani and Croft, 2017) developed an architecture to transform the text into a sparse representation, while they still retrieve documents using a term-based index like lucene[1] because the non-zero value in the sparse representations is treated as virtual terms. (Bai et al., 2018) (Grbovic et al., 2016) developed a uniform query and document embedding framework by generating ngram embedding using user session and click data, and then generalize it to arbitrary text by mean average pooling of ngram embedding. Since ngram is a common and effective skill in a variety of NLP tasks, training a good ngram representation requires a massive of datasets, which may be a bottleneck for many researchers and companies. Meanwhile, the model capacity of DSSM and its' variations makes it not capable to capture complex semantic meanings of natural language.

Recently, ELMo(Peters et al., 2018), GPT-2(Radford et al., 2019) and BERT(Devlin et al., 2018) show the great power of unsupervised pre-training in NLP tasks. The BERT model is built on a 12 layer transformer architecture, pre-trained with large scale text data. The pre-trained models can be fine-tuned easily and outperform many state-of-art models in various NLP tasks. We used the pre-trained BERT-Base(Chinese)[2] model released by Google and fine-tuned the model for semantic representation. Our fine-tuned model outperformed many state-of-art models in deep relevance matching, and obtain a great success in semantic retrieval task.

[1]https://lucene.apache.org/

[2]https://github.com/google-research/bert

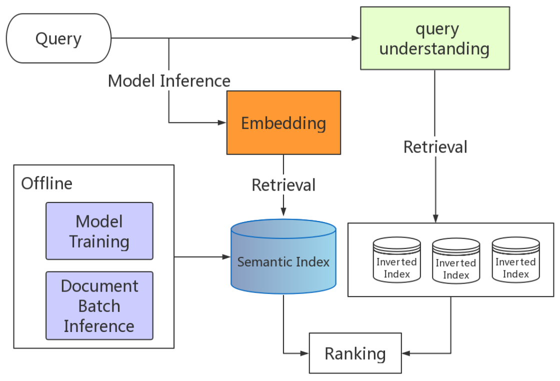

Figure 1: Overall framework of our proposed work.

3 Approach

In this section, we first illustrate our proposed semantic retrieval framework, which is composed of both offline and online parts respectively. Then, we introduce the model structure used for encoding queries and titles, and the techniques we used to boost the performance.

3.1 Deep Semantic Retrieval Framework

Figure 1 shows our proposed system architecture. The offline module includes model training, document embedding inference, and semantic index builder. While in online serving, both query's semantic embedding and traditional term base query parser are computed, and then those two results are sent to semantic index service and inverted index service respectively for document retrieval. Finally, documents retrieved from both index services are merged and sent to ranking service for document scoring.

3.2 Deep Semantic Representation Model

The pre-trained BERT model can be leveraged for semantic ranking and matching(Qiao et al., 2019) in various ways. We developed two models here: BERT(rep) and BERT(rel). The BERT(rep) model uses the pre-trained BERT model to obtain embedding of query and doc respectively, while the BERT(rel) model concatenates query and document first and get the one representation for a query document pair. The final score of a query, document pair is computed as below:

$$BERT(rep)(q,d) = dot(\frac{1}{L}\sum_{i=1}^{L}\vec{q}_i^{last}, \frac{1}{L}\sum_{i=1}^{L}\vec{d}_i^{last}) \quad (1)$$

In the equation 1, we use the mean average of last layer as encoder output for each query and document, and compute the dot product of two embedding as matching score, where L represents the max sequence length. We also tried directly using the last layer of [CLS] term's embedding, but performed worse than the average pooling described in equation 1.

$$BERT(rel)(q,d) = \vec{w} \times \vec{qd}_{cls}^{last} \tag{2}$$

The equation 2 use embedding of last layer's [CLS] token and weighted sum it to a scalar by vector $\vec{w}$, where $\vec{w}$ is a full connection layer with only one output. The model capacity of this method is more powerful than BERT(rel) because it calculates the term interaction between the query and document in the self-attention layers. However, since the BERT(rel) model is an interaction base model, this model can not be applied to semantic retrieval.

Both two models are trained through a supervised learning fashion, with a pairwise max margin hinge loss to distinguish relatively positive and negative samples. The loss function for one query is:

$$\frac{1}{M}\sum_{i}\sum_{j\neq i} max(0, \tau - (y_i - y_j) \times (p_i - p_j)) \tag{3}$$

where p_i and p_j represent to model score computed for each $< query, document >$ pair, and y_i and y_j is the label for each document respectively. τ is the hyper parameter called margin to determine how far the model need to push a pair away from each other. The margin parameter is tuned for the best performance here.

3.2.1 Additive Sampling

We use the additive data sampling technique to further enhance model performance. Therefore, the data we used to train our model is comprised of two parts, human annotated data, and negative sampled data. Negative sampling has been successfully applied in many tasks, such as neural language modelling(Mikolov et al., 2013), e-commerce list embedding(Grbovic and Cheng, 2018), graph embedding(Ying et al., 2018) and so on.

Sampling negative training instance is also useful for model training in this scenario, since different from traditional term-based retrieval method, the vector space search is much more likely to retrieve irrelevant documents. Thus we propose to augment more irrelevant documents. When the negative samples were added to training, the model learned to push relevant and irrelevant documents away from each other, then the model is more robust to noisy documents.

A straightforward way of negative sample mining is to select negative samples corresponding to a uniform distribution over the whole corpus, in particular, irrelevant documents here. However, this simple strategy fails to generate hard negative samples, which provide more important information for the model. Therefore, we propose another negative sampling method. At first, we train a baseline model with only human annotated data. Then we use this model to encode documents and queries. After that, we use an unsupervised cluster algorithm to assign each document and query a cluster id. Finally, we uniformly random selected negative documents from the cluster that query was distributed.

For convenient, we call this kind of negative sampling name of $NEG_{cluster}$, and globally sampled data name of NEG_{global}. We append $NEG_{cluster}$ and NEG_{global} to the raw dataset for per query and fine-tuned the model again to obtain our final model.re

We show the whole training procedure in the

Algorithm 1 Training Framework of our proposed model

```
Require:
     human annotated data $D$, BERT pre-trained model $M$
 1: $M_1 \leftarrow \{D, M\}$, fine-tune the model $M$ by $D$
 2: compute embedding $E$ for query and doc using $M_1$
 3: compute cluster centroids $C$ by $E$
 4: for all $d \in Docs$ do
 5:     compute closest centroid $C_d$ for $d$
 6: end for
 7: for all $q \in Query$ do
 8:     compute closest centroid $C_q$ for $q$
 9:     uniform sample $NEG_{global}$ from whole doc set
10:     uniform sample $NEG_{cluster}$ among docs where $C_d = C_q$
11:     $D_1(q) = \{D(q) \cup NEG_{global} \cup NEG_{cluster}\}$
12: end for
13: $M_2 \leftarrow \{D_1, M\}$, fine-tune the model $M$ by $D_1$
Ensure:
     BERT model $M_2$
```

Algorithm 1, and Table 1 illustrate examples of human annotated samples and auto-generated negative samples. Four titles are corresponding to each query, which represents the human annotated positive title, human annotated negative title, NEG_{global} title and $NEG_{cluster}$ title respectively. From the table, we can see that the $NEG_{cluster}$

Table 1: Examples of the dataset. There are four titles correspond to query “机器学习编程”(machine learning programming). They are annotated positive, annotated negative, NEG_{global} and $NEG_{cluster}$ respectively.

Query	Titles
机器学习编程 Machine Learning Programming	机器学习，在理论和编程方面要如何准备？ How to prepare programming parts to study machine learning
	深度学习如何入门？ How to get started to study deep learning
	什么是好朋友，有谁给过你怎样的深感动？ What is good friend, who has touched you deeply
	回忆我的编程之路 recall my history of programming

sample's meaning is much closer to query than that of NEG_{global}, which makes the model more robust for hard samples.

3.3 Online Serving

Once the model was trained, we need to serve it on the fly. We first computed the embedding of all documents and build a vector index using faiss[3] (Johnson et al., 2017), which was open sourced by facebook and support k-nearest-neighbor search for vector data in milliseconds. We developed a c++ based semantic index server to provide efficient concurrent online service. Our model was inferenced on a GPU server, and inference speed was accelerated 2 times faster than tf-serving through a c++ based library developed by us. During the online serving, when a query was received, the GPU server first inferences the query embedding, and downstream sends the query embedding to semantic index service for document retrieval. For the balance of efficiency and effect, we retrieve k most similar documents in the semantic service for next stage ranking, where k is set to 20 here.

4 Offline Experiments

In this section, we carry out offline experiments to illustrate the performance of our proposed semantic retrieval methods. In the experiment, we train the model with 1 epoch, use Adam(Kingma and Ba, 2014) with a learning rate of 10^{-5}, $\beta 1 = 0.9$, $\beta 2 = 0.999$.

4.1 DataSets

The data annotated by human editors is a list of triplets like <query, doc, relevance>. The relevance score has three grade 0, 1, 2, which represents *bad*, $fair$ and $excellent$ respectively. The dataset contains 36159 queries and 1181229 query doc pairs. Beside the dataset for training, we additionally annotated a small dataset for test, the test dataset contains 2703 queries and 84244 query-doc pairs. The summarize of dataset is shown at Table 2.

4.2 Evaluation Metrics

We evaluate our proposed model from ranking and retrieval aspects. We compared the ranking performance using Normalized Discounted Cumulative Gain(NDCG), and retrieval performance with Recall. The way how these metrics are calculated will be introduced in Section 4.4 and Section 4.5 respectively.

4.3 Baselines

- ClickSim
  A relevance matching model(Jiang et al., 2016) which use web-scale click data to generate term representations for query and document, and use cosine similarity to represent query document relevance.

- K-NRM
  An interaction based matching model using kernel pooling(Xiong et al., 2017).

- Match Pyramid
  An interaction based matching model using convolutions on term matching matrix(Pang et al., 2016a).

- DSSM
  A representation based model proposed by Microsoft Research(Huang et al., 2013). The

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

Table 2: Brief statistics of annotated data

	Query	QueryDoc	Excellent	Fair	Bad
TrainSet	36159	1181229	106181	357552	717464
TestSet	2703	84244	9552	17801	56891

Table 3: Recall performance of different models. The **Lexical** represents the traditional term-based retrieval.

Methods	Recall Num	Recall Rate
Lexical	12963	54.9%
DSSM	5	n.a
DSSM+Lexical	12963	54.9%
BERT(rep)	9794	**41.5%**
BERT(rep)+Lexical	16394	**69.4%**

model proposed here using word vectors pretrained on document title corpus. And three full connection layer with size of 300, 300, and 128 dimensions are used for text encoding.

4.4 Recall Performance

We use metric Recall to evaluate the model's retrieval performance here. This metric measures how many relevant documents are retrieved by a given model. For a given query q, the Recall rate is calculated as,

$$Recall_q = \frac{Ret_q \cap Rel_q}{Rel_q} \quad (4)$$

where Ret_q represents the retrieved documents for q, Rel_q stands for all the relevant documents for query q, where relevant documents are defined as document relevance annotated larger than 0 here.

To evaluate the recall performance offline, we first built semantic index both for our model and baseline model. We computed representation for document title of each model, then we used the representation embedding to build semantic index. Once queries' embedding of each model were computed, we retrieved the top k documents by k-nearest-neighbor search. Besides comparing the recall measure of different models only using semantic index, we compared the recall enhancement when the semantic index was added to the lexical inverted index. We used a commercial term-based inverted index engine developed by us and build a lexical index with it. Both lexical inverted index and semantic index were built to retrieve documents, with top 300 and top 20 respectively. Then we calculated the recall of the union set.

In the experiment, since document size of testset is small, we need a larger document corpus to make the recall measured more accurately. Therefore, both semantic index and lexical index were built with all human annotated data, including trainset and testset. And recall metric were calculated using only queries in the testset.

Table 3 shows the result of different models, BERT(rep) outperforms baseline model DSSM significantly in the recall measure. And after adding our model as a supplement to the lexical index, the recall rate is improved from 54.9% to 69.4%. While the baseline model, DSSM performs poorly on this task.

4.5 Ranking Performance

- NDCG score

Since our proposed model could not only be applied in document retrieval but also applied in the ranking stage. We measured the model's ranking quality through Normalized Discounted Cumulative Gain(NDCG). For a ranked document list, the NDCG for a query is calculated as,

$$NDCG_n = \frac{DCG_n}{IDCG_n} \quad (5)$$

where $IDCG_n$ represents the DCG score when the list was perfectly ranked by relevance. We compute following variation of Discounted Cumulative Gain(DCG)(Järvelin and Kekäläinen, 2002),

$$DCG_n = \sum_{i=1}^{N} \frac{2^{label_i}}{\log_2(i+1)} \quad (6)$$

According to the equation 6, higher relevance label contribute to higher weight in the computation. We calculate $NDCG$ with different rank list size of $\{1, 3, 5\}$ respectively. Table 4 shows that our model is superior to the state of art deep relevance matching models, and BERT(rep) model is slightly worse than the BERT(rel) model since BERT(rel) model uses self-attention between the query and title tokens before aggregates final score. However, both the BERT(rep) model and BERT(rel) model outperform other baselines significantly.

Table 4: Ranking performance between different models.

Method	**NDCG@1**	**NDCG@3**	**NDCG@5**
Match Pyramid	0.7332	0.7312	0.7425
K-NRM	0.712	0.7118	0.7251
ClickSim	0.619	0.6164	0.6315
BERT(rep)	**0.7775 (6.04%)**	**0.7754 (6.04%)**	**0.7849 (5.71%)**
BERT(rel)	0.8009	0.7962	0.8044

- feature importance in ranking model

Table 5: Feature importance in GBDT ranking model

FeatureName	**ImpFraction**
BERT(rep)	**34.11%**
ClickSim	10.65%
K-NRM	4.72%
Match Pyramid	2.82%

We feed the doc product of query doc embedding into a gbdt ranking model(Burges, 2010) as a relevance feature, and observe the feature importance after the tree model was trained. The feature importance was computed by the statistics collected during the tree ensemble training procedure. Table 5 shows that without adding BERT(rel) feature, the BERT(rep) feature ranks first in the ranking function, and accounts for 34% of importance among all features in our ranking function.

4.6 Analysis of Negative Samples

In Section 3.2.1, we described two negative sampling generator method: the NEG_{global} samples and $NEG_{cluster}$ for training data enhancement. We tuned the negative samples size, and obtained the best performance with 10 NEG_{global} and 10 $NEG_{cluster}$ respectively. After adding negative samples, the average negative sample size for a given query increased from 19.9 to 39.9. Table 6 shows the model performance with different kinds of negative samples. Only adding NEG_{global} can improve NDCG@3 at about 0.5%, when adding $NEG_{cluster}$, the NDCG@3 is further improved by 0.8%. Therefore, the overall measurements are enhanced by 1.4% after additive sampling.

Table 6: Model performance with different sampling, where HM represents the human annotated data. We show the NDCG@3 here because the NDCG of other rank performs similarly.

Model	**NDCG@3**
HM	0.7615
HM + NEG_{global}	0.7669
HM + NEG_{global} + $NEG_{cluster}$	**0.7754**

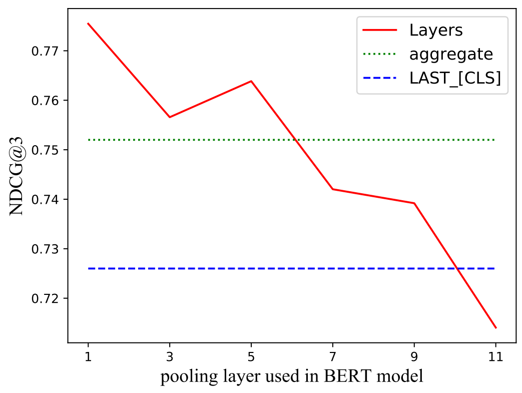

Figure 2: Affect of using different layers and pooling method. The x-axis represents the layer used for text pooling, starting from 1 to 11, and the y-axis represents the NDCG@3 metric.

4.7 Results of different pooling method of BERT

In this paper, we use the reduce-mean of the last layer as BERT(rep) model's pooled output. Different layers of BERT may own different aspects of knowledge about the input sequence. To verify the effectiveness of different layers, we trained different models, with pooled output from different layers respectively. From Figure 2, the red solid line shows that the layer closest to last obtains higher NDCG measure. This is reasonable since higher layers make the model contains more parameters.

Besides comparing the results of different layers, we also developed a method aggregate the embedding of all layers. In this method, an attention layer calculates the weight across different layers, therefore a weighted sum of each layer's embedding on each position is the final representation of each term. After that, we used reduce-mean of all terms' embedding as the final pooled output. The result of aggregation is shown as the green dot line, which does not outperform simple average pooling on the last layer. Meanwhile, we also tried using [CLS] term's embedding of the last layer as pooled output, but it behaved even worse. In conclusion, using mean average pooling of the last layer as final pooled output performs best in this

Table 7: Clicked Search Rate(CSR) of experimental groups and control groups. We set two control groups and two experimental groups to eliminate the online traffic bias.

Group	Total	Top	Torso	Tail
Control-1	76.74%	75.52%	80.32%	74.97%
Control-2	76.74%	75.493%	80.37%	74.98%
Exp-1	**77.30%**	75.46%	80.35%	**76.05%**
Exp-2	**77.31%**	75.53%	80.38%	**76.03%**

scenario, even though some work claims aggregating layers is useful (Kondratyuk, 2019).

5 Online Evaluation and Case Study

5.1 Online A/B Testing

After offline evaluations, we conduct an online a/b test to further verify our proposed system. In the online experiment procedure, 40 percent of online traffic were randomly distributed to four groups, 2 control groups, and 2 experimental groups. The metric we used to evaluate is the Clicked Search Rate(CSR), which is computed as:

$$CSR = \frac{SearchNum_{clicked}}{SearchNum} \qquad (7)$$

After a week's observation, as shown in Table 7, the overall CSR of two experimental groups both surpass two control groups by 0.65%, which is relatively a huge improvement to our experience. We also examined the online performance for queries with different frequency. We split queries into Top, Torso, and Tail by query search times in a day. Since our proposed method mainly focuses on boosting the performance of long tail queries, we can see the CSR metric is not significant in the Top and Torso query part. But the metric increased by nearly 1.05 % in the Tail part, which contributed to the largest algorithm iteration in the first half of 2019.

5.2 Case Study

This section highlights some good cases after our system was deployed online.

We show the final result ranked at top 6 for query "送外卖不认识路" (do not know the way to deliver food) at Table 8, where SEMANTIC represents the document retrieved from the proposed semantic index, and LEXICAL for traditional term-based inverted index.

In this case, three documents are retrieved from semantic index, and the relevance is also much better than the document from traditional inverted index. Notice that there are many ways to express "不认识路"(do not know the way) in Chinese, while the semantic index retrieved documents indeed capture the several alternatives of expressing it: "不知道路线", "不认路", "不懂路". And the term retrieved document only contains the same term "不认识路" as query expressed.

Table 8: Top ranked titles for query "送外卖不认识路" (do not know the way to deliver food)

Index	Title
SEMANTIC	配送外卖不知道路线怎么办? What if the food delivery do not know the deliver route?
LEXICAL	本人是送外卖新手，不知道其中的送餐技巧 I am new to food delivery, and I do not know the deliver skills.
SEMANTIC	为什么外卖员不认路? Why does the food delivery do not know deliver route?
LEXICAL	恶劣天气叫外卖的行为是否恰当? Is the behavior of takeaway in bad weather appropriate?
LEXICAL	我想送外卖，又怕不认识路。 I want to be a food delivery, but I am afraid I don't know the way.
SEMANTIC	不懂路可以送美团外卖吗 Can I be a food delivery at MeiTuan if I am not familiar with route?

6 Conclusion

In this paper, we present an architecture for semantic document retrieval. In this architecture, we first train a deep representation model for query and document embedding, then we build our semantic index using a fast k-nearest-neighbor vector search engine. Both offline and online experiments have shown that retrieval performance is greatly enhanced by our method.

For the future work, we would like to explore a more general framework that could use more signals involved for semantic retrievals, like document quality features, recency features, and other text encoding models.

References

Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Re-*

search and development in information retrieval, pages 19–26. ACM.

Hiteshwar Kumar Azad and Akshay Deepak. 2017. Query expansion techniques for information retrieval: a survey. *arXiv preprint arXiv:1708.00247*.

Xiao Bai, Erik Ordentlich, Yuanyuan Zhang, Andy Feng, Adwait Ratnaparkhi, Reena Somvanshi, and Aldi Tjahjadi. 2018. Scalable query n-gram embedding for improving matching and relevance in sponsored search. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 52–61. ACM.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2017. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830*.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Qing Chen, Mu Li, and Ming Zhou. 2007. Improving query spelling correction using web search results. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 181–189.

Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, New York, NY, USA.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 126–134. ACM.

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*.

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity search in high dimensions via hashing. *very large data bases*, pages 518–529.

Mihajlo Grbovic and Haibin Cheng. 2018. Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 311–320. ACM.

Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, Ricardo Baeza-Yates, Andrew Feng, Erik Ordentlich, Lee Yang, and Gavin Owens. 2016. Scalable semantic matching of queries to ads in sponsored search advertising. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 375–384. ACM.

Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tao Xu, Shulin Yang, Huizhong Duan, Qing Zhang, Nick Barrow-Williams, Bradley C Turnbull, Brendan M Collins, et al. 2018. Applying deep learning to airbnb search. *arXiv preprint arXiv:1810.09591*.

Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1443–1452. ACM.

Po Sen Huang, Xiaodong He, Jianfeng Gao, Deng Li, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Acm International Conference on Conference on Information and Knowledge Management*.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Wan Ji, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. 2014. Deep learning for content-based image retrieval:a comprehensive study. In *Acm International Conference on Multimedia*.

Shan Jiang, Yuening Hu, Changsung Kang, Tim Daly Jr, Dawei Yin, Yi Chang, and Chengxiang Zhai. 2016. Learning query and document relevance from a web-scale click graph. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 185–194. ACM.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Daniel Kondratyuk. 2019. 75 languages, 1 model: Parsing universal dependencies universally. *arXiv preprint arXiv:1904.02099*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016a. A study of matchpyramid models on ad-hoc retrieval. *arXiv: Information Retrieval*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016b. Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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*.

Abhay Prakash and Dhaval Patel. 2014. Techniques for deep query understanding. *arXiv: Information Retrieval*.

Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of bert in ranking. *arXiv preprint arXiv:1904.07531*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Stefan Riezler and Yi Liu. 2010. Query rewriting using monolingual statistical machine translation. *Computational Linguistics*, 36(3):569–582.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014a. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014b. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 373–374. ACM.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*, pages 55–64. ACM.

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 *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–332. ACM.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. *knowledge discovery and data mining*, pages 974–983.

Hamed Zamani and W Bruce Croft. 2017. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514. ACM.

Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 575–584. ACM.