A Deep Top-K Relevance Matching Model for Ad-hoc Retrieval

Zhou Yang*, Qingfeng Lan†, Jiafeng Guo‡, Yixing Fan‡, Xiaofei Zhu*, Yanyan Lan‡, Yue Wang*, and Xueqi Cheng‡

*School of Computer Science and Engineering, Chongqing University of Technology
†University of Chinese Academy of Sciences
‡CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190,China
{yangzhou, fanyixing}@software.ict.ac.cn, {guojiafeng, lanyanyan}@ict.ac.cn, lanqingfeng14@mails.ucas.ac.cn, {zxf, wangyue}@cqut.edu.cn

Abstract. In this paper, we propose a novel model named DTMM, which is specifically designed for ad-hoc retrieval. Given a query and a document, DTMM firstly builds an word-level interaction matrix based on word embeddings from query and document. At the same time, we also compress the embeddings of both document word and query word into a small dimension, to learn the importance of each word. Specifically, the compressed query word embedding is projected into the term gating network, and the compressed document word embeddding is concatenated into the interation matrix. Then, we apply the top-k pooling layer (i.e., ordered k-max pooling) on the interaction matrix, and get the essential top relevance signals. The top relevance signals is associated with each query term, and projected into a multi-layer perceptron neural network to obtain the query term level matching score. Finally, the query term level matching scores are aggregated with the term gating network to produce the final relevance score. We have tested our model on two representative benchmark datasets. Experimental results show that our model can significanly outperform existing baseline models.

Keywords: Deep Learning, Relevance Matching, Ad-hoc Retrieval

1 Introduction

In traditional information retrieval models, the relevance of the document is measured according to the exact matching signals. That is to say, the relevance score is determined by the frequencies of query word from the document. These models often face the typical term mismatch problem [1] since the semantic matching signals are ignored. Recently, deep neural network have achieved great success in many natural language processing tasks. At the same time, these deep neural networks have also been applied in information retrieval, which called neural information retrieval (i.e., NeuIR). Neural information retrieval models, which measure the relevance based on continuous embedding [6], have achieved significant progress.

It is of great importance to model the word importance in retrieval models. In traditional retrieval models, they measured the word importance based on the inverse document frequency (i.e., IDF), which have been a specific requirement of retrieval models [2]. As these models only considered document words which equal to the query word, thus, it is sufficient to only take IDF of query words into consideration. Recently, neural retrieval models employ deep neural network to model the semantic matching between query words and document words. In this way, those words which relate to the query words are also used to measure the relevance. However, existing neural retrieval models ignored the importance of these non-query words, which is also critical in relevance judgement.Take the following case as an example:

Query: Introduction of animals living in water, such as sharks
A fragment of document A: Dolphins swimming in the water are looking for food.
A fragment of document B: A yellow puppy fell into the water.

From the above example we can see, compared with the exact matching signal water, dolphins and puppy as similar matching signal appear in the document A, B respectively. Given the semantic environment provided by water and sharks in query, the importance of dolphins should be greater than puppy [7]. So, matching errors can easily occur without emphasizing the importance of document words. When the importance of words is emphasized, it will have a beneficial effect on correct matching.

In this work, we take the document word importance into consideration while modeling the relevance between query and document. Specifically, we proposed a deep top-k relevance matching model (DTMM) for ad-hoc retrieval. DTMM is a deep retrieval model which takes the raw text of query and document as input, and extract relevance signals automatically through deep neural network to produce the final relevance score. Specifically, DTMM firstly build an interaction matrix, where each element denotes the interaction between the corresponding query word and document word. At the same time, we compressed the embedding of document word into a small dimension, and concatenate into the interaction matrix. In this way, the interaction matrix can not only capture the matching signals but also the document importance. Then, we apply the top-k pooling layer on the interaction matrix, and obtain the essential top relevance signals. The top relevance signals is associated with each query term, and projected into a multi-layer perceptron neural network to get the query term level matching score. Finally, the query term level matching scores are aggregated with the term gating network to produce the final relevance score.

We have conducted extensive experiments to verify the effectiveness of our model. Specifically, we tested our model in two benchmark datasets (i.e., MQ2007 and Robust04). The experimental result show that DTMM can significantly outperform existing baselines on both datasets. For example, the improvement of DTMM over the best neural ranking model (i.e., DRMM) on MQ2007 is about 8% in terms of MAP metric.

The next section discusses related work. Section 3 presents the Deep Top-K Relevance Matching . Experimental methodology is discussed in Section 4 and evaluation results are presented in Section 5. Section 6 concludes.

2 Related Work

In matching task, most of the traditional models are based on the exact matching signals, such as BM25 [11] and query likelihood [13]. The advantage of this model is that it takes into account the exact matching signals and different matching requirements.

In current neural deep model for matching task, there are two types of models. One is representation-focused deep matching models, another is interaction-focused deep matching models. Due to their emphasis on matching, interaction-focused matching deep models is more suitable for IR tasks.

The representation-focused deep matching models, will lose the exact matching signals, and which is important for IR. In these models, they mainly learn to express the lower dimensions of queries and documents before interacting. In these learning process, the exact signals will be lost. For example, ACR-I [4] is a typical model of it, with distributed representations. Also famous are representation-focused models DSSM [5] and CDSSM [12] using letter-tri-gram.

In the interaction-focused deep matching models, words is represented by a low dimension vector, usually expressed by word embedding. The model produces an interaction matrix and then matches the documents by learning the information from the interaction matrix. The advantage of these models is that they make full use of the exact signals. MatchPyramid [11] is a CNN based model that is easy to implement and pays attention to the matching of word level, phrase level, sentence level, but ignores the diverse matching requirement. DRMM is an interaction-focused deep matching model, which gives special importance to three factors of relevance matching [2], but neglects the importance of document when using histograms to solve diverse matching requirement. In DeepRank [9], it imitates human retrieval process and achieves good results, but this model is complex and can not be parallelized. Interaction-focused deep matching models and representation-focused deep matching models address the ranking task problem from different perspectives, and can be combined in the future [8].

3 A Deep Top-K Relevance Matching Model

Based on the above analysis, in view of the existing problems in the existing model, We proposed DTMM to solve the problem of ignoring document words importance. Our model is based on the DRMM. In general, it can be divided into four parts. The first part is to build the interaction matrix. DTMM constructs an interaction matrix with distributed representations of query and document words, In order to emphasize the importance of document, it adds the weight of document into the current interaction matrix to form a new interaction matrix.

The second part is the k-max pooling layer, which selects the top k strongest signals with the query dimension as the input of the next layer. The third part is fully connected network. The model sends the information into a multi-layer neural network and combines the weight value of the term gating network with the query terms to get a final score. After this, it uses hinge loss function as the objective function.

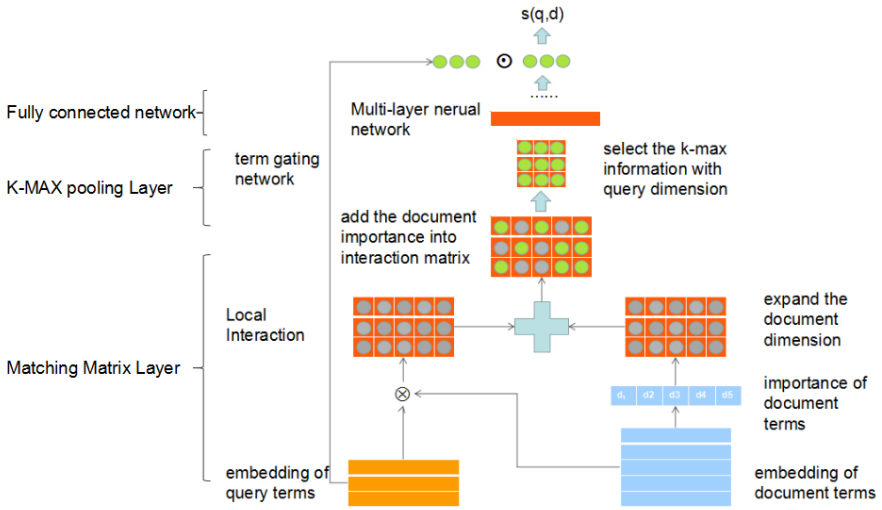

Fig. 1. Architecture of Deep Top-k Relevance Matching Model.

Formally, suppose that query and document are given in the form of vector sets: $q = \{q_1, q_2, ...q_M\}$ and $d = \{d_1, d_2, ...d_N\}$ represent the embedding of query and document separately, where M denotes the length of query, N denotes the length of document. The models we propose are as follows:

$$g_{qi} = softmax(w_{qi}q_i), g_{dj} = w_{dj}d_j \quad i = 1, 2, ...M, \quad j = 1, 2, ...N \tag{1}$$

$$z_i^0 = T_k(q_i \cdot d + g_{dj}) \qquad i = 1, 2, ...M, \qquad j = 1, 2, ...N \tag{2}$$

$$z_i^k = a_k(w^k z_i^{k-1} + b^k) \qquad i = 1, 2, ...M, \qquad j = 1, 2, ...N \tag{3}$$

$$s = \sum_{i=1}^{M} g_{qi} z_i^L \qquad i = 1, 2, ...M \tag{4}$$

In detail, g_{qi} and g_{dj} represent the weight of query and document words respectively, and w_{qi} and w_{dj} are the weights of corresponding neural nodes respectively.The difference is that the importance of query words is calculated from softmax. $\cdot$ denotes the interaction operator between a query term and the

document term, and DTMM model uses dot as interaction operator. T_k is a Top-K pooling function,namely k-max pooling function with the query dimension. z_i^k denotes the output of each neural layer, specially, z^0 is the first layer denoting interaction matrix. In the second equation, w^k and b^k are the weight matrix and bias of the k layer neural network, respectively. a_k is the activation function of each neural network layer, here we use softplus. g_i is the weight coefficient of the i-th query word, s is the final score of the model.The structure of the model is shown in Figure 1.

Next we will discuss the main components in the deep Top-K relevance matching model, including interaction matrix layer, k-max pooling layer, multi-layer neural network, model training and reveal how they are solved the problem before.

3.1 Interaction matrix layer

Initial interaction matrix: Given a query and a document, each word is in a distributed representation, interacted with dot to form the initial interaction matrix. To emphasize that the different words in the document have different significance levels, the importance of the document words is added to the matrix to form the final interactive matrix.

Document term importance: Our model first expresses the words of the document by embedding, and then uses a fully connected neural network to map each word into a small dimension, indicating the importance of the words. For example, there are 300 words in the document, the embedding dimension is 50, and dimension of the embedding matrix is 300*50. After mapping with a fully connected neural network, it is represented as 300*1 dimension. We need to add document term importance into the interaction matrix. Concretely, if the query has 5 words, then expanding the document term importance matrix to 5*300*1 dimension, then add it to the corresponding interaction matrix before.

3.2 K-max pooling layer

We observed that in essence, the matching histograms used in the DRMM model is actually sorted, in the meantime, the unimportant words in document are also added to the calculation of histograms. Through our research, A few words in document with a high correlation of the query basically determine the final score, while a large number of low correlation degrees are found. The low correlation signals, like stop words is little or has negative impact. On this basis, we propose a top-k pooling function based on query dimension to select the optimal signals, removing the bad signals. After processing of k-max pooling layer, the length of whole document dimension is K [3]. It forms a fixed value and provides the conditions for entering a fully connected neural network.

In addition, in order to achieve end to end training, we use the k-max pooling function instead of the histograms in the DRMM model. It make full use of

the existing word embedding to accelerate the training, it breaks through the limitations of the original word embedding and maximally avoids disadvantages.

3.3 Multi-layer neural network

Query term importance: Since importance of different query words is different, we use a weight coefficient to distinguish it. The greater the weight have, the more important the word is in the query. DTMM use a weighted network to calculate the weight coefficients of different query words. Specifically, we use the softmax function as a activation function. All of this imitates query term importance in DRMM.

Multi-layer neural network: After that, DTMM constructs a multi-layer neural network. Based on the powerful data fitting and learning ability of neural network, the features are extracted one by one. With the gradual deepening of the network, the extracted features become more and more abstract. Because the importance of each query is different, the output of the multi-layer neural network combined with the query importance to get a final score, which is used for document sorting.

3.4 Model training

The task of Ad-hoc information retrieval is a kind of sorting problem, so we use one of the sort loss functions hinge loss as the loss function in the model training.

$$L(\theta) = mean[\sum_{q} \sum_{d^+ \in D_q^+, d^- \in D_q^-} max(0, 1 - s(q, d^+) + s(q, d^-))] \tag{5}$$

In detail, θ represents all the parameters to be learned in the model, q denotes query, d^+ comes from the positive sample document sets D^+, which represents the documents that is positively related to the query. d^- comes from the negative sample document sets D^-, which represents the documents that is not related to the query. The method widen the gap between positive and negative samples, and makes the positive score larger than the negative case over 1. DTMM is optimized by back-propagation algorithm.

4 Experimental Methodology

In this section, We compared with several classical models and achieved good results. Next, we elaborate on the detailed process, results and analysis of our experiments.

4.1 Dataset

Million Query Track 2007: It is called MQ2007 for short. The data set is a subset of the LETOR4.0, which is collected by the web crawler from the domain name GOV2 web site, and the user clicks are used as the basis for the sorting of the document, Including 25M documents and 10000 queries. MQ2007 has a total of 58730 documents and 1501 queries. Among them, the words in the document and the query are lowercased and indexed, and the corresponding words are extracted with the Krovetz stem analyzer. In addition, referring to the list of stop words in INQUERY, we removed the stop words in the query. The parameters of the data set are detailed in table 1.

Robust04: Robust04 is a small news dataset.here we use Robust04-title as one of our data set. The topics are collected from TREC Robust Track 2004. Here the Robust04-Title means that the title of the topic are used as query. The collection is consist of 0.5M documents and 250 queries. The vocabulary size is 0.6M, and the collection length is 252M. More clearly described in the lower table 1.

Table 1. Statistics of collections used in this study. Here we tested our model DTMM on two data sets MQ2007 and robust04.

	MQ2007	robust04
query number	1501	250
document number	58730	324541

4.2 Baseline methods

Our baseline includes traditional models, including BM25 [11], and some recent neural ranking models. One type is representation-focused deep matching models, including ACR-I [4], DSSM [5], CDSSM [12], and another interaction-focused deep matching models as follows: ACR-II [4], MatchPyramid [11], DRMM [2]. In detail, when we use the MQ2007 as our dataset, the metrics of all of our models are set in [9], and when we use robust04-title, the metrics we referencing to are set in [2].

We select some neural deep matching models for comparison, and we will introduce these models below:

ARC-I: ARC-I is used in sentence completion, response matching, and paraphrase identification, which is a representation-focused model. ARC-I has been tested on a set of NLP tasks including response matching, sentence completion, and paraphrase identification.

DSSM: DSSM is a excellent model for web search. The original paper mentioned that training DSSM requires a lot of data. In the following experiments, it does not show excellent results.

CDSSM: DSSM is an improved version of CDSSM. It mainly changes the dense layer in DSSM to the convolution layer, getting more structural information by this way, and has improvement in performance.

ARC-II: It is an improved version of ACI-I. It has noticed the importance of interaction, and has learned interactive information earlier than ARC-I. ARC-I and ARC-II has no public code, so it is re-implemented and applied to the comparison model [2].

MatchPyramid: It is a widely used model, and its applications include paraphrase identification and paper citation matching. There are three versions of MatchPyramid. We choose the best models to compare. The model involved in the comparison is the original model provided by the author.

DRMM: DRMM is an interaction-focused model, With different types of histogram mapping functions (i.e., CH, NH and LCH) and term gating functions (i.e., TV and IDF). We choose the best model of the result to compare. Similarly, the model involved in the comparison is the original model provided by the author.

4.3 Implementation details

This section describes the configurations of DTMM.

Embedding size: We use the embedding with 50 dimension size, it is trained by the GloVe [10] model in advance. During the training process, we have not synchronized training embedding due to the small amount of data. Through our statistics, the vocabulary of corpus is 193367.

K-max pooling layer size: The k-max pooling layer selects 512 optimal signals, and other weak signals will not be input into the neural network. Through our research, different features and number of it in the data set affect the setting of this parameter size.

Multilayer neural network size: The size of the multi-layer neural network is set to [512,512,256,128,64,32,16,1], with activation function of softplus.

Model optimization: Optimization using Adam optimizer, with e = 1-5, learning rate = 0.001 and batch size =100. we conducted on MatchZoo development, it is an open source matching model development platform using keras tensorflow, including most advanced matching model nowadays. [1]

5 Evaluation Results

5.1 Ranking accuracy

Obviously, our proposed DTMM model has a significant improvement over baseline. The experimental results of the model in MQ2007 and robust04 are as follows:

On the MQ2007 data set, all the representation-focused models(including DSSM, CDSSM, ARC-I) and most interaction-focused models (including DRMM, ARC-II, MatchPyramid) do not work as well as BM25 [11]. In the previous model, only DRMM performs better than the BM25. The performance of the representation-focused models is generally not as good as the performance of

[1] The source of MatchZoo: https://github.com/faneshion/MatchZoo

Table 2. Comparison of different retrieval models over the MQ2007.

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10	P@1	P@3	P@5	P@10	MAP
BM25	0.358	0.372	0.384	0.414	0.427	0.404	0.388	0.366	0.450
DSSM	0.290	0.319	0.335	0.371	0.345	0.359	0.359	0.352	0.409
CDSSM	0.288	0.288	0.297	0.325	0.333	0.309	0.301	0.291	0.364
ARC-I	0.310	0.334	0.348	0.386	0.376	0.377	0.370	0.364	0.417
DRMM	0.380	0.396	0.408	0.440	0.450	0.430	0.417	0.388	0.467
ARC-II	0.317	0.338	0.354	0.390	0.379	0.378	0.377	0.366	0.421
MatchPyramid	0.362	0.364	0.379	0.409	0.428	0.404	0.397	0.371	0.434
DTMM	0.458	0.459	0.468	0.499	0.517	0.479	0.458	0.426	0.504

Table 3. Comparison of different retrieval models over the robust04.

Model	NDCG20	P@20	MAP
BM25	0.418	0.370	0.255
DSSM	0.201	0.171	0.095
CDSSM	0.146	0.125	0.067
ARC-I	0.066	0.065	0.041
DRMM	0.431	0.382	0.279
ARC-II	0.147	0.128	0.067
MatchPyramid	0.330	0.290	0.189
DTMM	0.463	0.432	0.314

interaction-focused model. To some extent, this illustrates the role of the three factors emphasized by relevance mathing in IR. The improvement of DTMM against the best deep learning baseline (i.e. DRMM) on MQ2007 is 20.6% wrt NDCG@1, 15% wrt P@1, 8% wrt MAP, which illustrates the superiority of our model on the IR task.

On the robust04 data set, the performance of most interaction-focused models is also obviously better than the representation-focused models. But one exception is that the interaction-focused model ARC-II has the same performance as the CDSSM, and is inferior to the representation-focused model DSSM. This may be duing to the uneven distribution of features in this data set. When ARC-II intercepts text length, it removes the important feature at the end of the document, which has an impact on model performance. In the same way, most of the interaction-focused models and representation-focused models can not exceed BM25 performance except DRMM model. On this data set, DTMM also achieves the best effect, compared to the best model DRMM. the improvement of DTMM against the best deep learning baseline (i.e. DRMM) on robust04 is 7.4% wrt NDCG@20, 13% wrt P@20, 12.5% wrt MAP, respectively. It needs to be explained that the reason we choose these indexes is because on the MQ2007 dataset, we refer to [2], and on the robust04 refer to, we refer to [9].

Table 4. Comparison of different version of DTMM. Where $DTMM_{no}$ represents the model without document words importance, the other is the complete model.

Model	NDCG@3	NDCG@5	NDCG@10	MAP
$DTMM_{no}$	0.424	0.435	0.469	0.490
DTMM	0.459	0.468	0.499	0.504

5.2 Performance on DTMM without document words importance

Table 4 shows the comparison between DTMM and DTMM without adding document word importance. Where $DTMM_{no}$ represents the model without document words importance, the other is the complete model. In the evaluation of ndcg@3, ndcg@5, ndcg@10 and MAP, the complete DTMM was higher than the incomplete model 8.25%, 7.58%, 6.39%, 2.85% respectively. It shows that emphasizes the importance of the different words in the document is meaningful.

5.3 Performance on different k-max pooling layer of DTMM

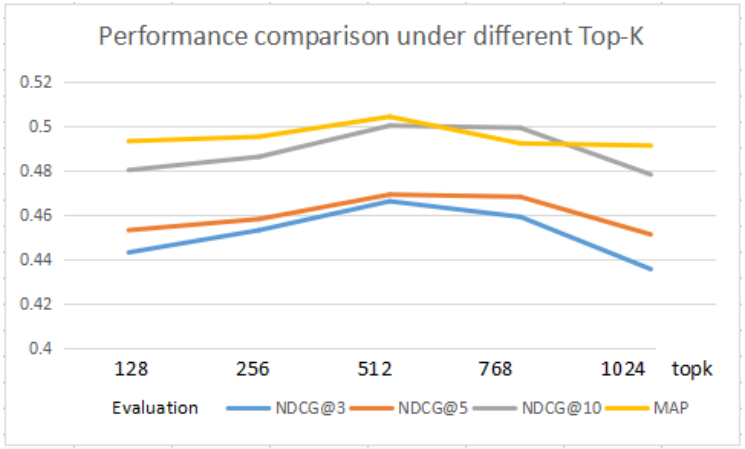

Fig. 2. Performance comparison on DTMM with different Top-K. The abscissa is the size of the top k window, and the ordinate is the size of the indicator. Our indicators include NDCG@3, NDCG@5, NDCG@10, MAP

Figure 2 shows the performance of different versions of DTMM on the MQ2007 data set. Obviously, with the parameter selection from small to large, the performance of the model first improves and then decreases. It can be seen that different top-k has an impact on the performance of the model. Since each data set has different characteristic types and numbers, it is important to select the parameter Top-k. When the selected Top-k is too large or too small, it will have different effects on the model. If it is too small, the selected features are insufficient. If the Top-k is too large, the unimportant information is selected and it will have an impact on the performance.

6 Conclusion

This article presents a DTMM model for ad-hoc tasks. It emphasizes the importance of document words for IR task, and also proposes a k-max pooling method based on query dimension, which can eliminate noise while retaining the strongest signals. The model includes three parts: interaction matrix layer, k-max pooling layer and multi-layer neural network. Each part of the model can be parallelized, making large-scale commercial offerings possible.

For the future work, as DTMM is at word-level, we will consider adding phrase-level and sentence-level matching signals to the it. More generally, we will delve into the factors that are favorable to IR and introduce them into the new model.

7 Acknowledgments

This work was funded by the 973 Program of China under Grant No. 2014CB340401, the National Natural Science Foundation of China (NSFC) under Grants No. 61425016, 61472401, 61722211, and 20180290, the Youth Innovation Promotion Association CAS under Grants No. 20144310, and 2016102, and the National Key R&D Program of China under Grants No. 2016QY02D0405.

References

1. W. B. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*, volume 283. Addison-Wesley Reading, 2010.
2. J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64. ACM, 2016.
3. J. Guo, Y. Fan, Q. Ai, and W. B. Croft. Semantic matching by non-linear word transportation for information retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 701–710. ACM, 2016.
4. B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.
5. P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM, 2013.
6. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
7. T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

8. B. Mitra, F. Diaz, and N. Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299. International World Wide Web Conferences Steering Committee, 2017.
9. L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng. Text matching as image recognition. In *AAAI*, pages 2793–2799, 2016.
10. J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
11. S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241. Springer-Verlag New York, Inc., 1994.
12. Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. 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, 2014.
13. C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.