

# An Evaluation of the Kernel Based Neural Ranking Model in NTCIR-13 WWW

Zhuyun Dai, Chenyan Xiong, Jamie Callan  
 Language Technologies Institute  
 Carnegie Mellon University  
 Pittsburgh, PA 15213, USA  
 {zhuyund, cx, callan}@cs.cmu.edu

## ABSTRACT

This paper describes CMUIR’s participation in the NTCIR-13 We Want Web (WWW) task. In the context of the Chinese subtask, we experimented with a neural network approach using the kernel based neural ranking model (K-NRM). The model learns a word embedding that encodes IR-customized soft match patterns from a Chinese search log. The learned model is then directly applied to re-rank the baseline run result lists of the Chinese subtask. We extend K-NRM to incorporate multiple document fields for richer text presentation. We also experimented with different re-ranking cutoffs to reduce the effect of the gap between training and testing domains. Evaluation results confirmed the effectiveness of K-NRM.

## Keywords

Web Search, Neural IR, Embedding

## Team Name

CMUIR

## Subtasks

Chinese

## 1. INTRODUCTION

In NTCIR-13, CMUIR group participated in the Chinese subtask of the We Want Web (WWW) task [3]. In this challenge, we took a neural ranking approach based on our previous work on the kernel-based neural ranking model (K-NRM) [5].

K-NRM is a neural ranking architecture that aims to model multi-level soft-match between queries and documents. It uses distributed representations to represent query and document words. Their similarities are constructed into a translation model. Then a kernel-pooling layer is used to softly count the frequencies of word pairs at different similarity levels, for instance, exact matches, strong soft matches, and weak soft matches. These multi-level signals are used as features in a ranking layer, which produces the final ranking score. In this work, we also extend K-NRM architecture to incorporate multiple representations of the document.

In this task, we trained K-NRM on a Chinese commercial search log with user clicks signals supervision labels. The trained models were then used directly to re-rank the baseline run results list retrieved from the SogouT-16 document collection [1]. Evaluation results show the effectiveness of

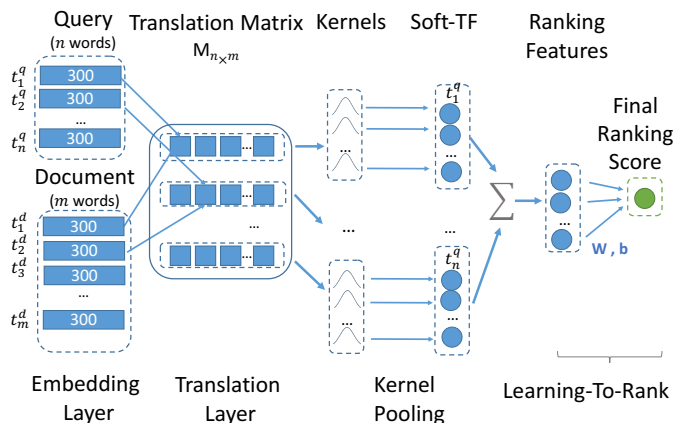


Figure 1: The Architecture of K-NRM. Given input query words and document words, the embedding layer maps them into distributed representations, the translation layer calculates the word-word similarities and forms the translation matrix, the kernel pooling layer generate soft-TF counts as ranking features, and the learning to rank layer combines the soft-TF to the final ranking score.

K-NRM despite many differences in the training and testing tasks.

## 2. MODEL ARCHITECTURE

We adopted the K-NRM model from our previous work. This section gives a brief description the architecture of K-NRM.

Given a query  $q$  and a document  $d$ , K-NRM generates a ranking score  $f(q, d)$  using query words  $q = \{t_1^q, \dots, t_i^q, \dots, t_n^q\}$  and document words  $d = \{t_1^d, \dots, t_j^d, \dots, t_m^d\}$ . As shown in Figure 1, K-NRM consists three components: translation model, kernel-pooling, and learning to rank.

### 2.1 Translation Model

K-NRM first uses an **embedding layer** to map each word  $t$  to an L-dimension embedding  $\vec{v}_t$ . Then it constructs a translation matrix  $M$ . Each element in  $M$  is the cosine similarity between a query word embedding and a document word embedding.

## 2.2 Kernel-Pooling

K-NRM then uses a set of RBF kernels to convert word-word interactions in the translation matrix  $M$  to query-document ranking features  $\phi(M)$ .

$$\phi(M) = \sum_{i=1}^n \log \vec{K}(M_i)$$

$$\vec{K}(M_i) = \{K_1(M_i), \dots, K_K(M_i)\}$$

$\vec{K}(M_i)$  applies  $K$  kernels to the  $i$ -th query word's row of the translation matrix, summarizing (pooling) it into a  $K$ -dimensional feature vector. The log-sum of each query word's feature vector forms the query-document ranking feature vector  $\phi$ . The RBF kernel  $K_k$  calculates how word pair similarities are distributed around it: the more word pairs with similarities closer to its mean  $\mu_k$ , the higher its value. Kernels with different  $\mu$  focuses on different soft-match patterns; for example, a kernel with  $\mu = 1$  calculates the number of words that exact match the query word, and a kernel with  $\mu = 0.5$  softly counts the number of document words whose similarities to the query word are close to 0.5.

## 2.3 Learning to Rank

The ranking features  $\phi(M)$  are combined by a ranking layer to produce the final ranking score:

$$f(q, d) = \tanh(w^T \phi(M) + b).$$

$\tanh$  is the activation function that squeezes the range of ranking score into  $[-1, 1]$  to facilitate the learning process. It is rank-equivalent to a typical linear learning to rank model.

## 2.4 Multiple representations

Web page consists of multiple fields, for example, title, URL and the body. Each field represents the document in a distinct way, and provides evidences of relevance from its own perspective. The original K-NRM model [5] only considers the interaction between the query and the document *title*. In this challenge, we extend K-NRM architecture to make use of multiple representations. Each field is viewed as a sequence of words, the translation matrix between the field  $f$  and the query,  $M_f$ , is calculated, and  $K$  features are pooled through kernel-pooling, generating a  $K$ -dimension feature vector  $\phi(M_f)$ . Given  $F$  fields,  $K \times F$  features will be generated. This leads to the ranking features as follows.

$$\Phi(M) = \phi(M_1) \oplus \dots \oplus \phi(M_f) \oplus \dots \oplus \phi(M_F),$$

where  $\oplus$  is the concatenation operation. These features are combined by the learning-to-rank layer to generate the final relevance score.

In this work, we let multiple fields share the word embeddings to reduce data sparsity and accelerate training. Future work will explore separate embeddings for different document fields.

## 3. EXPERIMENTAL SETUP

### 3.1 Dataset

K-NRM was trained with the same settings defined by Xiong et al [5]. Our training corpus is a sample of search logs from a major Chinese commercial search engine in China. We did not use the official training set because it only has

200 queries and was not enough to train the K-NRM model. Our training set includes 31M search sessions and 95,229 unique queries. The overlap between our training set and the NTCIR WWW task test set is small: only 12 (query, document) pairs appeared in both sets. These data points were removed from the training set.

Same as in [5], training labels were generated from user clicks. The DCTR click model [2] was used to infer the relevance labels of each (query, document) pair appeared in the search log.

The query log provides title and URL for each displayed documents. We trained and tested K-NRM using with configurations: 1) only using title and 2) using both title and URL as two representations of the documents. We will further discuss it in section 4.

### 3.2 Parameter Settings

All runs shared the same model parameters settings as in [5]: one exact match kernel of  $\mu = 1$  and  $\sigma = 0.00001$ , and 10 kernels with  $\mu$  ranged from 0.9, 0.7, ..., -0.9 and  $\sigma = 0.1$  were used for each field. Model was trained Adam Optimizer with a batch size of 16, learning rate of 0.01 and  $\epsilon = 1e - 5$ .

## 4. SUBMITTED RUNS AND EVALUATION

We submitted 5 runs with different model configurations. These runs differ in the document representations, the number of candidate documents to re-rank, and the word embeddings.

- **CMUIR-C-NU-Base-1:** The training query log provides the title and URL of each displayed document. This run used both of the fields. Following the settings in [5] where on average each testing query has 30 candidate documents, this run re-ranked the top 30 documents in the baseline result list. The model was trained end-to-end on the training search log.
- **CMUIR-C-NU-Base-2:** This run used the same model as CMUIR-C-NU-Base-1. Different from the first run, this run was tested to re-rank the top 100 documents.
- **CMUIR-C-NU-Base-3:** This run only used the title field. For each query, it re-ranks the top 30 documents in the baseline result list. It is trained end-to-end on the training search log.
- **CMUIR-C-NU-Base-4:** This run used the same trained model as CMUIR-C-NU-Base-3. For each query, it re-ranks the top 100 documents in the baseline result list.
- **CMUIR-C-NU-Base-5:** This run used the title field. For each query, it re-ranks the top 100 documents in the baseline result list. Different from the first 4 runs, this model used a fixed pre-trained word embedding, and only learned the learning-to-rank parameters. The pre-trained embeddings were obtained using the skip-gram method from word2vec [4] on the document titles displayed in training search log.

Descriptions and evaluation results of the submitted runs are listed in Table 1.

**Multiple Representations.** In this task, we explore adding URLs as a second representation. As can be seen from Table 1, models with both title and URL fields (CMUIR-C-NU-Base-1,2) performed consistently better than their title-only competitors (CMUIR-C-NU-Base-1,3).

Table 1: Submitted runs and evaluation results.

	Representations	Reranking Depth	Embedding	nDCG@10	Q@10	nERR@10
CMUIR-C-NU-Base-1	<b>Title, URL</b>	<b>30</b>	<b>End-to-End</b>	<b>0.6145</b>	<b>0.6294</b>	<b>0.7583</b>
CMUIR-C-NU-Base-2	Title, URL	100	End-to-End	0.5873	0.5955	0.7046
CMUIR-C-NU-Base-3	Title	30	End-to-End	0.6059	0.6163	0.7406
CMUIR-C-NU-Base-4	Title	100	End-to-End	0.5667	0.5780	0.7086
CMUIR-C-NU-Base-5	Title	100	Pre-trained	0.5915	0.5996	0.7372

**Re-ranking depth.** This analysis investigates the effects of re-ranking depth by comparing the trained K-NRM model on re-ranking the top **30** or **100** documents of the baseline run. The evaluation results in Table 1 show that in the end-to-end models (CMUIR-C-NU-Base-1,2,3,4), re-ranking top 30 always have higher performance than re-ranking top 100. This is because the domain differences between the training task and the testing task. K-NRM was trained on a commercial search log. In the training search log, there are only a few documents associated with each query (10-30 per query); all of these candidate documents are of high quality. On the other hand, the testing scenario is to re-rank a BM25 baseline run. Documents at the lower positions of the baseline result lists are likely to be of lower quality, and may contain patterns that diverge from the training data. Using the top candidate documents reduced the differences between the training and testing data, thus yielded higher ranking precisions.

**Pre-trained Embeddings.** Due to the training/testing gap discussed above, we also tested a K-NRM model with fixed-embedding (CMUIR-C-NU-Base-5). The embedding layer was pre-trained and fixed during the training. This model was expected to be less sensitive to corpus changes. As shown in Table 1, CMU-C-NU-Base-5 performed better than the end-to-end models on re-ranking the top 100 documents (CMU-C-NU-Base-2,4). It confirms that the fixed-embedding model is less sensitive to gap between the training and testing scenarios. Due to limited number of submissions, we do not have the evaluations of the fixed-embedding model on re-ranking top 30 candidate documents. However, evaluations on other datasets show that the fixed-embedding model performed worse than the end-to-end models at re-ranking high quality candidate documents [5].

Our best run, CMUIR-C-NU-Base-1, achieved nDCG@10 of 0.6145 and is among the best runs of the Chinese subtask. Its advantage consists of using multiple representations of the documents, reducing domain differences by re-ranking the high-quality candidate documents, and end-to-end learning a IR-customized word embedding.

The training task and the testing task differs in many aspects: different corpora, different quality of the initial rankings, and different labels (clicks v.s. manual relevance labels). In this work, the K-NRM models were directly applied to re-rank the testing result lists after training despite the many differences between the training and the testing. The good performance on the testing set demonstrates K-NRM’s robustness and the ability to generalize. We believe the performance could be further enhanced by addressing the domain differences. For example, one could fine-tune the trained model with the training set provided by the WWW Chinese subtask.

## 5. CONCLUSION

In this paper, we present our methods on NTCIR-13 WWW task, for Chinese subtask. We experimented with the kernel-based neural ranking model K-NRM based on our previous work [5]. Evaluation results show the effectiveness of K-NRM. Our analysis suggests that the model can benefit from multiple documents representations. The analysis also reveals the effects of domain differences between training and testing. One promising future direction is to bridge the domain differences with transfer learning approaches.

## 6. REFERENCES

- [1] L. Cheng, Z. Yukun, L. Yiqun, X. Jingfang, Z. Min, and M. Shaoping. SogouT-16: A new web corpus to embrace ir research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 2017.
- [2] A. Chuklin, I. Markov, and M. d. Rijke. Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 7(3):1–115, 2015.
- [3] C. Luo, T. Sakai, Y. Liu, Z. Dou, C. Xiong, and J. Xu. Overview of the ntcir-13 we want web task. In *Proceedings of the NTCIR-13 Conference*, 2017.
- [4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Advances in Neural Information Processing Systems 2013 (NIPS)*, 2013.
- [5] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th annual international ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2017)*. ACM, 2017.