RUCIR at the NTCIR-14 We Want Web-2 Task

Xue Yang, Shuqi Lu, Shijun Wang, Han Zhang, and Zhicheng Dou*

Beijing Key Laboratory of Big Data Management and Analysis Methods, China
Key Laboratory of Data Engineering and Knowledge Engineering, MOE, China
School of Information, Renmin University of China, Beijing 100872, China
{ruc_yangx,lusq,wangshijun,zhanghanjl,dou}@ruc.edu.cn

Abstract. The RUCIR team participated in the Chinese and English subtasks of the NTCIR-14 We Want Web-2 (WWW-2) Task. In this paper, we illustrate our approach for solving the ad hoc Web search problem and display the official results. For both Chinese and English subtasks, we adopted a learning to rank framework to re-rank candidate documents for each query. We extracted several traditional ranking features for each query-document pair, and at the same time, we trained deep neural models to get matching scores, and use the matching scores as deep features. The traditional features and deep features are fused by the learning to rank model.

Keywords: Web search ranking · Learning to rank · Neural IR

Team Name. RUCIR

Subtasks. We Want Web-2 (Chinese, English)

1 Introduction

The RUCIR team participated in the Chinese and English subtasks of the NTCIR-14 We Want Web-2 (WWW-2) Task [7]. This report illustrates our approach in the task and displays the results.

Ad-hoc retrieval is a common task of information retrieval. In ad-hoc retrieval, given a *query* and a text-based *corpus*, the system returns the *relevant* items. *Query* is a short textual description of the user's information need. *Corpus* is a collection of textual documents. *Relevance* means satisfying the users information need. Essentially, information retrieval can be considered as a problem of matching a query and a document.

Traditional IR models focus on statistical information of documents or exact term matching information between queries and documents. In this task, we implement traditional features, including BM25 [10], TF [3], IDF [3], TF-IDF [3], LMIR [13], document length, perfect match and complete match.

However, information need is not just the literally sequence of terms. It contains semantic information about the real information need. There are lots of

* Corresponding author: Zhicheng Dou.

synonyms and polysemous words (or phrases) in natural languages, which are out of the consideration of exact matching methods. Therefore, there is a desire to introduce semantic matching methods into information retrieval. Recently, deep learning has been widely used to handle the semantic matching problem, such as paraphrase identification and question answering. Taking advantage of deep learning, several neural IR models have been proposed. Deep neural models [4, 2, 12, 11, 9] learn to match documents and queries for ad-hoc ranking using neural networks. In this task, we introduce several deep neural IR models to generate deep neural features, including ARC-I [4], ARC-II [4], DRMM [2], aNMM [12], MV-LSTM [11] and DUET [9].

We believe that both the traditional matching features and the semantic matching information generated by deep neural networks are useful to ad-hoc ranking. Therefore, in the WWW-2 task, we try to leverage both types of features. We use the learning to rank [5] framework, which is able to incorporate both types of features, to train the final ranking model. More specifically, we use the LambdaMART [1] model, since it's one of the best learning to rank frameworks. We use the trained ranking model to estimate the documents relevance, and re-rank the documents in the baseline ranking lists provided by the organizer.

The rest of the paper is organized as follows. Section 2 introduces our model for this task and illustrates the details. Section 3 shows evaluation results of our submitted runs and provides discussion. We conclude in Section 4.

2 Learning to Rank Model

In this section, we introduce our work on WWW-2 task. The basic framework of our model is LambdaMART, a well performed learning to rank algorithm. Then, we focus on feature engineering. We extract multiple features from query-document pairs, including traditional features and deep neural features. Based on the extracted features, we utilize LambdaMART and re-rank the unlabeled data in the light of obtained ranking scores.

2.1 Dataset

Official Dataset We use the official dataset for this task. More specifically, for Chinese Subtask, we use the SogouT-16 [6] and SogouQCL [14] as the document collection. SogouT-16 contains about 1.17B Web pages, but we only index part of the "Category B" version from SogouT-16. Sogou-QCL contains 537,366 queries and 5 kinds of weak relevance labels based on different click models for over 12 million query-document pairs. It contains 2,000 Chinese queries with the traditional relevance assessments which are annotated by three trained assessors as well. Although users' behaviour information is available, we haven't used them in the submitted runs due to the tight schedule of the task. For the English Task, we adopt the ClueWeb12-B13 as the document collection.

Training Dataset For English task, we adopt Category A of ClueWeb09 and Category B of ClueWeb12 for training data. We also collect about 300 queries and their relevance judgment files from TREC09 to TREC14 competitions. For Chinese task, we adopt Category B of SogouT-16 and Sogou-QCL as training data. We also use the subset of Sogou-QCL which contains 2,000 queries and about 20 query-document relevance judgments for each query.

2.2 Feature Extraction

We consider ranking task as a machine learning problem. So, we need to extract various kinds of features for each query-document pair for model training. Each training sample is represented by a multi-dimensional feature vector, and each dimension of the vector is a feature which indicates the importance of the document with respect to the query in terms of the feature aspect. In our model, we implement two kinds of features: traditional features and deep neural features.

Traditional Features We implement several relevance features that are widely used in existing works. Noting that we have four fields for each document: body, anchor, title, and URL, so the following features are implemented over each field. We also implement another group of features for combining all content of the four fields (the whole document). We show the brief introduction to the features in Table 1. More specifically, we have the following features:

- TF, IDF, TF-IDF: TF (term frequency) denotes the number of occurrences of query term in document. IDF (inverse document frequency) is computed as follows:
$$idf(q_i) = log\frac{|C| - df(q_i) + 0.5}{df(q_i) + 0.5}$$
where $df(q_i)$ is the number of documents which contain query term q_i, $|C|$ is the number of all the documents in document collection. TF-IDF is the product of TF and IDF.
- BM25: BM25 (Best Match) is computed as follows:
$$BM25(d,q) = \sum_{q_i \in q} \frac{idf(q_i) \cdot tf(q_i,d) \cdot (k_1+1)}{tf(q_i,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})}$$
where $avgdl$ denotes the average document length in the corpus. In this task, we set $k_1 = 2.5$, and $b = 0.8$.
- LMIR: A smoothing method for language model. LMIR is computed as follows:
$$p_\mu(w|d) = \frac{c(w;d) + \mu(p(w|C)}{\sum_w c(w;d) + \mu}$$
The idea is to adjust the probabilities of according to the query.
- PerfectMatch: Boolean value. It identifies whether the entire query string occurs in the document.

Table 1. Traditional Features. (Anchor is only used in the English subtask)

Name	Description	Fields
BM25	BM25 with default parameters	(anchor), title, URL, body, whole
TF-IDF	TF-IDF model	(anchor), title, URL, body, whole
LMIR	Language model with Dirichlet smoothing	(anchor), title, URL, body, whole
TF	Sum of term frequency	(anchor), title, URL, body, whole
IDF	Sum of inverse document frequency	(anchor), title, URL, body, whole
DL	Document length	(anchor), title, URL, body, whole
PM	Perfect match	(anchor), title, URL, body, whole
CM	Complete match	(anchor), title, URL, body, whole

– CompleteMatch: Boolean value. It means whether all the query terms occur in the document. DIfferent from PerfectMatch, CompleteMatch doesn't consider the order of the query terms.

Deep Neural Features First, we use Word2vec [8] to obtain the distributed representations of queries and documents. We utilize Word2vec to obtain the vector of each term occurs in the corpus. Then the distributed representations of a query or a document can be defined as the mean vector of the terms in the corresponding query and the document. We then calculate the cosine similarity between a query representation and a document representation as a *embedding feature* for a query-document pair. Embedding features can be regarded as the basic use of word embedding and the pre-trained word embedding will be applied to deep neural models we utilize in this task.

To obtain deep neural features, we train deep neural matching models and use the trained models to calculate the matching scores of unlabeled query-document pairs as deep neural features. The deep neural models we used in this task are as follows:

ARC-I [4], which learns representation vectors for two pieces of text respectively with CNNs and gets the matching score by a multi-layer perceptron layer (MLP).

ARC-II [4], a variant of ARC-I. The model first learns the interaction representation of two texts by 1D convolution, rather than independent representation. After a series of 2D convolution and 2D pooling, the final matching score is calculated by an MLP layer.

DRMM [2], which generates matching histograms from interactions between each query term and document. All the histograms then go through MLP to get corresponding matching score and the final score is calculated by a softmax function.

aNMM [12], which replaces the position-shared weighting with value-shared weighting in ARC-II, and integrates the result of each query term with a softmax function.

MV-LSTM [11], which presents input text by bidirectional long short-term memory networks (bi-LSTMs), and builds interaction matrix with cosine similarity between text representations generated by different central words. A k-max pooling and a MLP are behind.

DUET [9], which combines the results of two deep neural models. One matches the local representations of query and document, another matches the learned distributed representations of query and document.

2.3 Model Training

From several learning to rank algorithms, we choose LambdaMART [1], a stable approach which can directly optimize evaluation metrics, to train, validate and test on the dataset.

We partition the training data into five equal parts for 5-fold cross validation. We select 4 folds for training and the other as validation data each time. The training set is used to learn ranking models. The validation set is used to tune the hyper parameters of the learning algorithms, such as the number of iterations in LambdaMART. After obtaining the hyper parameters, we finally train a model on these five folds for further testing.

2.4 Evaluation Metrics

In both Chinese and English tasks, we use nDCG@K, Q@K and nERR@K for evaluation.

nDCG@K is defined as follows:

$$nDCG@k = N_k^{-1} \sum_{i=1}^{n} g(r_i)d(i)$$

where k means the number of results to be evaluated. N_k denotes the maximum of $\sum_{i=1}^{n} g(r_i)d(i)$ and it is used for normalization. r_i denotes the relevance level of the document ranked at the i-th position. $g(r_i)$ denotes a gain and $d(i)$ denotes a discounting function.

Q@K is defined as follows:

$$Q@K = \frac{1}{min(K,R)} \sum_{r=1}^{K} J(r) \frac{C(r) + \beta cg(r)}{r + \beta cg^*(r)}$$

where R is the number of relevant documents. $J(r)$ denotes the relevance level of the document ranked at the r-th position. $C(r)$ is the number of relevant documents from position 1 to r. $cg(r)$ denotes cumulative gain at position r.

nERR@K is defined as follows:

$$nERR@K = \sum_{r=1}^{K} \frac{1}{r} \prod_{i=1}^{r-1} (1 - R_i)R_r$$

where $\prod_{i=1}^{r-1}(1 - R_i)R_r$ denotes the probability that user is satisfied with the document at position r and stops browsing.

3 Runs and Evaluation

3.1 Submitted Runs

For **Chinese** subtask, we submit the following five runs ("DE" in the run name means matching with query description, "CO" means matching with query content):

RUCIR-C-DE-PU-Base-1. In this run, we train learning to rank model based on *traditional features* and *embedding features* which are obtained by matching query description and multiple document fields.

RUCIR-C-CO-PU-Base-2. In this run, we train learning to rank model based on *traditional features* and *embedding features* which are obtained by matching query content and multiple document fields.

RUCIR-C-DE-PU-Base-3. In this run, we train learning to rank model based on *traditional features* which are obtained by matching query description and multiple document fields.

RUCIR-C-DE-PU-Base-4. In this run, we train learning to rank model based on *traditional features* and *deep neural features* which are obtained by matching query description and multiple document fields.

RUCIR-C-DE-PU-Base-5. In this run, we train learning to rank model based on *deep neural features* which are obtained by matching query description and multiple document fields.

For **English** subtask, we submit the following five runs ("DE" in the run name means matching with query description, "CO" means matching with query content):

RUCIR-E-DE-PU-Base-1. In this run, we train learning to rank model based on *traditional features* which are obtained by matching query description and multiple document fields.

RUCIR-E-CO-PU-Base-2. In this run, we train learning to rank model based on *traditional features* which are obtained by matching query content and multiple document fields.

RUCIR-E-DE-PU-Base-3. In this run, we train learning to rank model based on *traditional features* and *embedding features* which are obtained by matching query description and multiple document fields.

RUCIR-E-DE-PU-Base-4. In this run, we train learning to rank model based on *traditional features* and *deep neural features* which are obtained by matching query description and multiple document fields.

Table 2. Official results of Chinese subtask.

Runs	nDCG@10	Q@10	nERR@10
RUCIR-C-DE-PU-Base-1	0.4515	0.4228	0.5792
RUCIR-C-CO-PU-Base-2	**0.4866**	**0.4571**	**0.6044**
RUCIR-C-DE-PU-Base-3	0.4503	0.4223	0.5630
RUCIR-C-DE-PU-Base-4	0.4458	0.4226	0.5619
RUCIR-C-DE-PU-Base-5	0.2745	0.2404	0.3832

Table 3. Official results of English subtask.

Runs	nDCG@10	Q@10	nERR@10
RUCIR-E-DE-PU-Base-1	0.3137	0.2973	0.4469
RUCIR-E-CO-PU-Base-2	**0.3489**	**0.3352**	**0.4917**
RUCIR-E-DE-PU-Base-3	0.3137	0.2973	0.4469
RUCIR-E-DE-PU-Base-4	0.3293	0.3094	0.4602
RUCIR-E-DE-PU-Base-5	0.2876	0.2659	0.4188

RUCIR-E-DE-PU-Base-5. In this run, we train learning to rank model based on *deep neural features* which are obtained by matching query description and multiple document fields.

3.2 Experimental Results

Table 2 and Table 3 show the evaluation results of our submitted runs.

From Table 2 and Table 3, we find that matching with original query achieves better performance than matching with query description, i.e., `RUCIR-C-CO-PU-Base-2` and `RUCIR-E-CO-PU-Base-2` outperform other runs (containing "DE" in the name). Perhaps in a matching problem, original query is more suitable to interact with document rather than query intent. In the Chinese subtask, the combination of traditional features and embedding features (`RUCIR-C-CO-PU-Base-2`) outperforms other feature sets. In the English subtask, traditional features (`RUCIR-E-CO-PU-Base-2`) make the best performance. Besides, deep neural features take their advantage in semantic matching combined with traditional IR features. For example, nERR of `RUCIR-E-DE-PU-Base-4` is 0.4602, while `RUCIR-E-DE-PU-Base-1` is 0.4469. It illustrates that both exact matching and semantic matching contribute to ad-hoc retrieval.

4 Conclusions

The RUCIR team participated in the Chinese and English subtasks of the NTCIR-14 We Want Web-2 (WWW-2) Task. We applied a learning to rank framework for both Chinese and English subtasks, and tried to combine deep neural models with traditional IR models by feature engineering. The experimental results show that traditional features make more contribution than deep

neural features. We will make further explorations on the performance of deep neural features in the future work.

Acknowledgements

Zhicheng Dou is the corresponding author. This work was supported by National Key R&D Program of China No. 2018YFC0830703, National Natural Science Foundation of China No. 61872370, and the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China No. 2112018391.

References

1. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning **11**(23-581), 81 (2010)
2. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A deep relevance matching model for ad-hoc retrieval. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. pp. 55–64. ACM (2016)
3. Hiemstra, D.: A probabilistic justification for using tf× idf term weighting in information retrieval. International Journal on Digital Libraries **3**(2), 131–139 (2000)
4. Hu, B., Lu, Z., Li, H., Chen, Q.: Convolutional neural network architectures for matching natural language sentences. In: Advances in neural information processing systems. pp. 2042–2050 (2014)
5. Liu, T.Y., et al.: Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval **3**(3), 225–331 (2009)
6. Luo, C., Zheng, Y., Liu, Y., Wang, X., Xu, J., Zhang, M., Ma, S.: 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. pp. 1233–1236. ACM (2017)
7. Mao, J., Sakai, T., Luo, C., Xiao, P., Liu, Y., Dou, Z.: Overview of the ntcir-14 we want web task. Proceedings of the 14th NTCIR Conference on Evaluation of Information Access Technologies (2019)
8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
9. Mitra, B., Diaz, F., Craswell, N.: Learning to match using local and distributed representations of text for web search. In: Proceedings of the 26th International Conference on World Wide Web. pp. 1291–1299. International World Wide Web Conferences Steering Committee (2017)
10. Robertson, S., Zaragoza, H., et al.: The probabilistic relevance framework: Bm25 and beyond. Foundations and Trends® in Information Retrieval **3**(4), 333–389 (2009)
11. Wan, S., Lan, Y., Guo, J., Xu, J., Pang, L., Cheng, X.: A deep architecture for semantic matching with multiple positional sentence representations. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
12. Yang, L., Ai, Q., Guo, J., Croft, W.B.: anmm: Ranking short answer texts with attention-based neural matching model. In: Proceedings of the 25th ACM international on conference on information and knowledge management. pp. 287–296. ACM (2016)

13. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. ACM Transactions on Information Systems (TOIS) **22**(2), 179–214 (2004)
14. Zheng, Y., Fan, Z., Liu, Y., Luo, C., Zhang, M., Ma, S.: Sogou-qcl: A new dataset with click relevance label. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. pp. 1117–1120. ACM (2018)