ICTNET at TREC 2017 Common Core Track

Xu Chang[1,2,3], Liying Jiao[1,2,3], Jinlong Liu[1,2,3], Weijian Zhu[1,2,3], Yuanhai Xue[1,2], Li Zha[1,2], Yue Liu[1,2], Xueqi Cheng[4]
1) Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190
2) Key Laboratory of Web Data Science and Technology, CAS
3) University of Chinese Academy of Sciences, Beijing, 100190
4) Institute of Network Technology，ICT（YANTAI），CAS
{changxu, jiaoliying, liujinlong, zhuweijian}@software.ict.ac.cn; {xueyuanhai, char, liuyue, cxq}@ict.ac.cn

1. INTRODUCTION

The common core track is a new track for TREC 2017. The track will serve as a common task for a wide spectrum of IR researchers, thus attracting a diverse run set that can be used to investigate new methodologies for test collection construction. This track provides the structured NYTimes corpus. Its primary goal is to get the relative documents from the corpus with the given topics, and there are 1855660 news across from 1987 to 2007 on NYTimes in this common core track. Two sets of topics are used for searching relative News, one of which is to be judged by NIST and crowd workers and the other is to be judged by crowd workers only. Participants need choose the first or the both topics according to the requirements. There are three common text information retrieval model: Vector Space Model, Probabilistic Model and Inference Network Model. BM25 is a probabilistic function to rank the list of matched documents according to a given query[4]. Solr uses the BM25 rank function when doing the search work. It is an open source enterprise search platform and can do the full-text search work. Solr runs in the servlet container. User can get the results from the web interface. In our work, we choose Solr as the tool for indexing documents and searching topics.

We have an exploration on the topics, which are distributed by the NIST. Most of the topics contain less than three words and some include even one word. Therefore, we need to expand the query words to query more information. We select the apache Solr[5] as our retrieve frame in order to improve the effectiveness on the automatic query expansion. Recently, neural network is widely used in NLP. We use the word-embedding model for automatic query expansion and the TextRank algorithm with the description of the topic to extract the keywords as the expansion words. Next, we build index for the corpus and get the query results with Solr.

2. QUERY PREPROCESS

In this section, we mainly preprocess the query words. First, we train a model to convert words into vectors by the approach in Section 2.1. Second, we extend the query words following an incremental procedure based on word embedding in Section 2.2. Finally, using the TextRank algorithm in Section 2.3, we obtain the keywords from the description of the topic.

2.1. CBOW[6]

Word2vec is a kind of algorithm used to generate the distributed description of words. In our experiment,

we use the CBOW（Continuous Bag of Words Model)model to produce embedded word. We will introduce the CBOW model in detail.

In the CBOW architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption)[7]. For the sentence, "the cat jumps over the lazy dog", we can set {“the”, ”cat”, ”jumps”, ”the”, ”lazy”, ”dog”} as the context to predict the current word “over”. The model can catch multiple different degrees of similarity between words such as semantic and syntactic similarity.

There are two parameter matrices $\boldsymbol{V} \in \mathbb{R}^{n\times|V|}$and $\boldsymbol{U} \in \mathbb{R}^{|V|\times n}$ in CBOW model. In addition, n is the size of embedding space, $|V|$ is the vocabulary size of training set. The specific calculation process is shown in Figure-1. First, it generates the one-hot word vectors for the input context of size k: $(x_{c-k}, x_{c-k+1}, \dots, x_{c+k})$.[8]Second, we multiply $x_{c-k}, x_{c-k+1}, \dots, x_{c+k}$ with the matrix $\boldsymbol{V}$, obtain the vectors $v_{c-k}, v_{c-k+1}, \dots, v_{c+k}$, and calculate the average vector $\hat{v}$ of these vectors. Third, we multiply $\hat{v}$ with the matrix $\boldsymbol{U}$ to produce score vector z. Finally, we convert the score vector z into probabilities by using softmax function. The Loss function of model is cross entropy, using the gradient descent to update the parameter matrices $\boldsymbol{V} \in \mathbb{R}^{n\times|V|}$and $\boldsymbol{U} \in \mathbb{R}^{|V|\times n}$. The rows of matrix $\boldsymbol{V}$ are actually our word vectors.

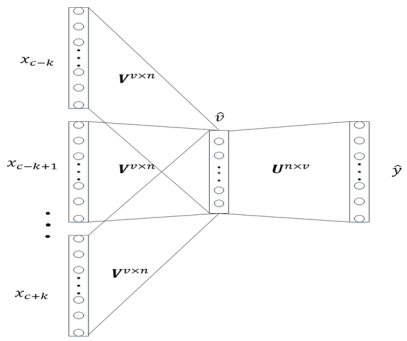

Figure-1

2.2. Query expansion based on word embedding

In this section, we implement two approaches to expand the query words according to the paper roy2016using[9].

2.2.1. Pre-retrieval kNN based approach

We assume that the given query word set Q is $\{q_1 \dots q_m\}$. We define the set of candidate expansion words is C as

$$C = \bigcup_{q\in Q} NN(q) \tag{1}$$

In Equation (1), NN (q) are the nearest k neighbors of q in extended query word set. We compute the mean cosine similarity between each candidate word and all the query words in C as shown in the follow formula.

$$Sim(w, Q) = \frac{1}{|Q|}\sum_{q_i\in Q} w \cdot q_i$$

The candidate word set is in descending order of the value of $Sim(w, Q)$. The top K words are chosen as the final expansion words.

2.2.2. Pre-retrieval incremental kNN based approach

The second method is a simple extension of the Pre-retrieval kNN based approach. Compared with the method in Section 2.2.1, it follows an incremental procedure rather than the procedure that the words are selected according to the similarity to each query word in a single step. The second method is based on the assumption that the most similar words have lower drift than the words occurring later in the list. Because the most similar words are the most appropriate candidate expansion words, we assume that the expansion words should be similar to each other. According to the assumption above, we use an iterative process to prune words of NN (q).

First, we arrange the nearest neighbors of q in descending order of similarity to q. The candidate word list is $w_1, w_2, \dots, w_N$. Then, we prune the K least similar neighbors to obtain word list $w_1, w_2, \dots, w_{N-K}$. Next, we add the first word w_1 to the expansion word set and reorder the list $w_2, \dots, w_{N-K}$ in descending order of similarity to w_1. We repeat the process, prune the K least similar neighbors in the new list to obtain list $w'_2, \dots, w'_{N-2K}$ and add the first word w'_2 to the expansion word set, for l times. At each step, the nearest neighbors list is reordered and pruned based on the list obtained in the previous step. Finally, we get the candidate expansion word set of q as $NN_l(q)$. In this set, the words are similar to each other and the query word q. With regards to the parameter l, high value may cause query drift while low value will perform similarly to the first method. In our model, we choose $l = 5$. Finally, we construct the expansion words set as in Section 2.2.1, except that use $NN_l(q)$ in place of $NN(q)$.

2.3. TextRank

TextRank[10] algorithm is based on PageRank [11]. It is usually used to generate keywords and summaries from the text. We use TextRank with the description of the topics to generate keywords for extending the query words.

First, we split the text into words. Every word is considered as a node in the network. We set parameter k as the window size. For example, in the sentence, "the cat jumps over the lazy dog", we could set k to four. Then we obtain the windows, which consist of "the cat jumps over", "cat jumps over the", "jumps over the lazy" and "over the lazy dog" .if two words are included in one window, there will be an unbounded edge to connect them. At last, we call the PageRank algorithm to generate keywords from text.

3. EVALUATION

In our work, we make six submissions about 50-topic set judged by NIST assessors. In the submission of ICT17ZCJL02, query word are manually expanded, while in other submissions the expansion is automatic

Run Tag	MAP	NDCG	P@10T	Description
ICT17ZCJL01	0.1837	0.4117	0.4508	Title only Solr Retrieve Framework
ICT17ZCJL03	0.1513	0.3785	0.4080	Query expansion with google news corpus
ICT17ZCJL05	0.1434	0.3607	0.3794	Query expansion with NYTimes corpus
ICT17ZCJL06	0.1620	0.3821	0.4284	Use the description below the topic
ICT17ZCJL07	0.1816	0.4103	0.4672	Trim the weight for the query term
Best	0.5377	0.7699	0.9160	Best trec results
Median	0.2280	0.6787	0.5480	Median trec results

Table 1. Performance of different Run Tag, compared to the best and the median results of the track

Table 1 shows the results of different submitted runs and the average of best and median score of this common core track. We find that weight of query word causes many retrieve differences according to ICT17ZCJL06 and ICT17ZCJL07. Additionally, there is a decrease on the ICT17ZCJL03 and ICT17ZCJL05 compared with ICT17ZCJL01, which shows the phenomenon called "query drift" as we add new query words. This problem causes the semantics of new query terms drift from the originals, but this effect is weakened when we add to the keywords of description in ICT17ZCJL06. We raise the weight of the original query words on ICT17ZCJL07. This method improves the hit rates on top 10 performance and keep the overall performance stable at the mean time.

4. CONCLUSION& Acknowledgements

In this paper, we study how to extend the query word automatically and compare the effects of several approaches. The evaluation shows that query expansion using the word-embedding model and TextRank to extract the key words from the description below the topic perform badly because of the query drift. We solve the problem of query drift by trimming the weight of query word and the expansion words. Word-embedding based on the given NYTimes can enhance the Top@10 hit rates, but it costs much more time for build the word embedding model and its scalability is poor when adding new document.

We would like to thank all organizers and assessors of TREC and NIST. This work is sponsored by Taishan Scholars Program of Shandong Province，China（No.ts201511082）, This work is also supported by National Key Research and Development Program of China under grant 2016YFB1000902, and NSF Foundation of China under grants 61572473 and 61772501.

5. REFERENCE

[1]. https://en.wikipedia.org/wiki/Information_retrieval

[2]. https://www.google.com.hk/

[3]. https://www.baidu.com/

[4]. Robertson S, Zaragoza H. The probabilistic relevance framework: BM25 and beyond[J]. Foundations and Trends® in Information Retrieval, 2009, 3(4): 333-389.

[5]. http://lucene.apache.org/solr/

[6]. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. ICLR Workshop, 2013.

[7]. https://en.wikipedia.org/wiki/Word2vec

[8]. http://web.stanford.edu/class/cs224n/lecture_notes/cs224n-2017-notes1.pdf

[9]. Roy D, Paul D, Mitra M, et al. Using word embeddings for automatic query expansion[J]. arXiv preprint arXiv:1606.07608, 2016.

[10]. R. Mihalcea, P. Tarau. 2004. TextRank: Bringing Order into Texts. In Proceedings of EMNLP2004.

[11]. Page L. The PageRank citation ranking: Bringing order to the web[J]. Stanford Digital Libraries Working Paper, 1998, 9(1):1-14.