ICTNET at TREC2017 Complex Answer Retrieval Track

Rui Cheng[123], Xiaomin Zhuang[123], Hao Yan[123], Yuanhai Xue[12], Zhihua Yu[12], Yue Liu[12], Xueqi Cheng[12]
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
{chengrui,zhuangxiaomin,yanhao}@software.ict.ac.cn; {xueyuanhai,yzh,liuyue,cxq}@ict.ac.cn

1.Introduction:

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers) [1]. The most common and popular information retrieval application is web search engine such as Google[2], Baidu[3], Bing[4] and Sogou[5]. These application will return top-N best retrieval result to users. Information Retrieval systems for the Web, i.e., web search engines, are mainly devoted to finding relevant web documents in response to a user's query[6].

Current retrieval systems performance well in phrase-level retrieval tasks which provide simple fact and entity-centric needs. Complex Answer Retrieval Track is a new track in 2017, which requests a more complex and longer retrieval result to answer a query. It focuses on developing systems that are capable of answering complex information needs by collating relevant information from an entire corpus.

Given an article stub Q, retrieval for each of its sections Hi, a ranking of relevant entity-passage tuples (E, P). Tow tasks are offered: passage ranking and entity ranking. This paper introduces an algorithm and a system for passage ranking.

The retrieval queries are outlines which consist of titles and section titles of articles. The retrieval collection consists of paragraphs which are come from Wikipedia articles. We use the BM25 algorithm and develop a system to retrieval the top-100 most relevant paragraphs.

2.Data analyzing

The task provides passage corpus, stubs, and example articles in CBOR-encoded archives to preserve hierarchical structure and entity links. All passages that are cut out of Wikipedia articles are assigned a unique (SHA256 hashed) paragraphId that will not reveal the originating article. Pairs of paragraphId and paragraphContent are provided as passage corpus. Each extracted page outline, provides the pageName (e.g., Wikipedia title) and pageId, and a nested list of children. Each section is represented by a heading, a headingId, and a list of its children in order. Children can be sections or paragraphs.

Training data contains outlines, and they are associated paragraphs and entities. These paragraphs are subordinate to the specified headings, such as the paragraphs belong to heading1.1, and headings1.1 is related to the paragraphs. In training set, queries are the outlines and retrieval set is made up of paragraphs which are come from Wikipedia articles. Testing data are outlines which can be used to extract queries.

3.Implementation principle

The task is required to find the corresponding paragraph content (or entity), given a specified wiki paragraph title or article title. We chose one of the tasks, that is given a "hierarchical qrels" (hierarchical query), to find the corresponding paragraph content.

First of all, we preprocessed the query. Here we only removed the identifier "/" and we did not remove the stop words because the stop words will also have useful information. And then we used Solr (which we will introduce below) to set up an inverted index on all the documents in the test set to speed up the query. Finally, we used the BM25[7] model to calculate the score for all the documents and sorted them. BM25 is shown below:

For a query, the score for a document is as follows：

$$score(Q, d) = \sum_{t \in Q} w(t, d)$$

$$w(t,d) = \frac{qtf}{k_3 + qtf} \cdot \frac{k_1 \cdot tf}{tf + k_1(1 - b + b \cdot l_d / avg\_l)} \cdot w^{(1)} \qquad w^{(1)} = \log_2 \frac{N - df + 0.5}{df + 0.5}$$

The parameters are explained as follows:
qtf: term frequency of a query; **tf**: term frequency of a document;**ld**: length of a document
avg_l: The average length of all documents;**N**: the number of documents;**df**: term frequency across all documents
b, k1, k3: hyper-parameters

4.Use solr to retrieve the documents

Solr[8] is a high-performance, Lucene-based full-text search server. Solr has extended Lucene to provide a richer query language than Lucene, which is configurable, extensible and optimized for query performance. There is no doubt that it is a very good full-text search engine.

Here we use Solr to complete the task which we have described above. Firstly, we splitted the data into many copies, each with one thousand text data, because Solr only supports indexing one thousand text data at a time. Then we set up inverted index on each copies. Since the default search model of Solr is BM25, we did not have to modify the configuration file. Finally we use http to retrieve documents and we will receive an XML/JSON response which contained 1000 results for a query.

Our model is BM25, BM25 is not a learning model based on data, but to retrieve TOP N related documents given queries. So here we think about the problem from another perspective, directly to analyze the results of the test set. Now we have retrieved TOP 100 documents for each query, and for each document, Solr will gives the relevance score. So we separate different documents into groups by their corresponding queries and then calculate mean, minimum, maximum, average and standard deviation respectively. We discover:

1) For each query, the mean of relevance scores is very different, ranging from 12.356 to 102.254.
2) The relevance scores of documents with the length of the queries is positively correlated. It can be deduced from the figure 1.

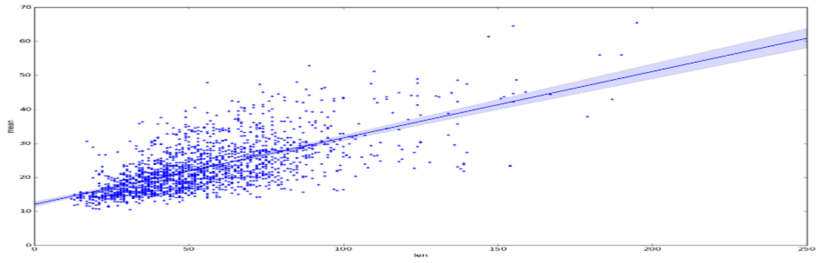

Figure 1. The relevance of query length and score

3) For each query, the variance of the relevance scores is very small.75% are less than 3.474 and only 25% range from 3.474 to 13.784.
4) For each query, the mean of relevance scores and the median of relevance scores is very close, indicating that the distribution of relevance scores are relatively uniform.

5. Acknowledgements

We would like to thank all organizers and assessors of TREC and NIST. This work is sponsored by the National Basic Research Program of China (the 973 program) under grant numbers 2014CB340401 and 2014CB340406, 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.

6.Reference

[1]. Manning, C. D., Raghavan, P., & Schütze, H. (2008). An introduction to information retrieval. Journal of the American Society for Information Science & Technology, 43(3), 824-825.
[2]. https://www.google.com.hk/
[3]. https://www.baidu.com/
[4]. https://www.bing.com/
[5]. https://www.sogou.com/
[6]. Delbru, R., Campinas, S., & Tummarello, G. (2012). Searching web data: an entity retrieval and high-performance indexing model. Web Semantics Science Services & Agents on the World Wide Web, 10(2), 33-58.
[7]. https://en.wikipedia.org/wiki/Okapi_BM25
[8]. http://lucene.apache.org/solr