ICTNET at Trec 2019 Decision Track

Wanqing Cui[12*], **Yan Jiang**[12*], **Shuchang Tao**[12*], **Hanzhang Guo**[12*]
[1]CAS Key Lab of Network Data Science and Technology
Institute of Computing Technology, Chinese Academy of Sciences, China
[2]University of Chinese Academy of Sciences, China
*{cuiwanqing18z, jiangyan18s, taoshuchang18z, guohanzhang18s}@ict.ac.cn

Abstract

In this paper we report on our participation in the Trec 2019 Decision Track[1] which aims to provide a venue for research on retrieval methods that promote better decision making with search engines and develop new online and offline evaluation methods to predict the decision making quality induced by search results. We convert this task into a standard information retrieval task and use traditional IR model. Finally we give a summary for the solution of our work.

1 Introduction

Search engine results underpin many consequential decision making tasks. Such as seeking health advice online, deciding the best treatment/diagnosis/test for a patient. However, when the search occurs within uncontrolled data collections, such as the web, where information can be unreliable, generally misleading, too technical, and can lead to unfounded escalations. Information from search engine results can significantly influence decisions, and research shows that increasing the amount of incorrect information about a topic presented in a SERP can impel users to take incorrect decisions.

So this track focus on encouraging participants to design search techniques that promote correct information over incorrect information. Given a data collection and a set of topics, participants should return relevant and credible information that will help searchers make correct decisions. Besides, more than simply define what is relevant, there are 3 types of results: correct and relevant, incorrect, and non-relevant. And the goal is to return relevant and correct information.

2 Model

This track is the retrieval task. Given the topic and narrative as query, we are supposed to return the most relevant and credible document. This is essentially a text matching problem. The higher the matching degree between the query and the document, the higher its ranking should be.

2.1 Terrier

Terrier is a highly flexible, efficient, and effective open source search engine, readily deployable on large-scale collections of documents. Terrier implements state-of-the-art indexing and retrieval functionalities, and provides an ideal platform for the rapid development and evaluation of large-scale retrieval applications.

With the Terrier tool, we can easily create an index. The first step is to collect documents from the entire document set, and get a list. We can enter the command under the project directory:

```
./bin/trec_setup.sh "path/to/docset"
```

The second step is index creation. Enter "./bin/trec_terrier.sh -i", we will get the index of the document set, but this need a long time. Finally, we search for the result of topics. Enter "./bin/trec_terrier.sh -r", and then we will get the results under the directory "./var/results".

In the meantime, before the second step, we can configure the index, such as the selected retrieval method, which tags in the document we indexed, and so on. These are achieved by configuring the file "terrier.properties", which can be referred to in its official documents.

2.2 BM25

Our retrieval method uses BM25, which is an algorithm used to evaluate the correlation between search terms and documents. It is based on the probability retrieval model. BM25 algorithm expands the scoring function of binary independent model by adding document weights and query weights. This extension is based on probability theory and experimental verification, and is not a formal model. On the basis of binary independent model, BM25 model takes into account the weight of words in query and the weight of words in documents, fits the formula of synthesizing the above considerations, and introduces empirical parameters through experiments.

Compared with tf-idf, BM25 makes better use of the fact that the relationship between word frequency and correlation is non-linear. Its general formula is as follows:

$$Score\left(Q,d\right)=\sum_{n}^{i}W_iR\left(q_i,d\right) \tag{1}$$

Among them, Q is the query, d is the document, q_i is the word in the query, and W_i is the word weight. Word weights can actually be calculated using idf.

2.3 DSSM

Deep Structured Semantic Models (DSSM)[2] uses a deep neural network to rank a set of documents for a given query as follows. First, a non-linear projection is performed to map the query and the documents to a common semantic space. Then, the relevance of each document given the query is calculated as the cosine similarity between their vectors in that semantic space. The neural network models are discriminatively trained using the clickthrough data such that the conditional likelihood of the clicked document given the query is maximized. Furthermore, to deal with large vocabularies, a word hashing method is used, through which the high-dimensional term vectors of queries or documents are projected to low-dimensional letter based n-gram vectors with little information loss.

The semantic relevance score between a query Q and a document D is then measured as:

$$R(Q,D)=cosine(y_Q,y_D)=\frac{y_Q^Ty_D}{||y_Q||||y_D||} \tag{2}$$

Where y_Q,y_D are the concept vectors of the query and the document, respectively. They are the output of DNN.

So the posterior probability of a document given a query from the semantic relevance score between them is:

$$p(D^+|Q)=\frac{exp(\gamma R(Q,D^+))}{\sum exp(\gamma R(Q,D))} \tag{3}$$

2.4 CDSSM

CDSSM [3] is an improvement over DSSM. Compared to DSSM, it uses CNN to learn the lower-dimensional semantic vectors of queries and documents. By using the convolution-max pooling operation, local contextual information at the word n-gram level is modeled first. Then, salient local features in a word sequence are combined to form a global feature vector. Finally, the high-level semantic information of the word sequence is extracted to form a global vector representation.

2.5 DRMM

Deep relevance matching model (DRMM) [4] employs a joint deep architecture at the query term level for relevance matching. It first builds local interactions between each pair of terms from a query and a document based on term embeddings. For each query term, DRMM maps the variable-length local interactions into a fixed-length matching histogram. Based on this fixed-length matching histogram, then a feed forward matching network is employed to learn hierarchical matching patterns and produce a matching score. Finally, the overall matching score is generated by aggregating the scores from each query term with a term gating network computing the aggregation weights. We show how our major model designs, including matching histogram mapping, a feed forward matching network, and a term gating network, address the three key factors in relevance matching for ad-hoc retrieval.

Suppose both query and document are represented as a set of term vectors denoted by $q=\{w_1^{(q)},...,w_M^{(q)}\}$ and $d=\{w_1^{(d)},...,w_N^{(d)}\}$, the final relevance score s is:

$$z_i^{(0)}=h(w_i^{(q)}\otimes d) \tag{4}$$

$$z_i^{(l)}=tanh(W^{(l)}z^{(l-1)_i}+b^{(l)}) \tag{5}$$

$$s=\sum_{i=1}^{M}g_iz_i^{(L)} \tag{6}$$

Where h denotes the mapping function from local interactions to matching histogram, $z_i^{(l)},l=0,...,L$ denotes the intermediate hidden layers for the i-th query term, and $g_i,i=1,...,M$ denotes the aggregation w eight produced by the term gating network. $W^{(l)}$ denotes the l-th weight matrix and $b^{(l)}$ denotes the l-th bias term, which are shared across different query terms.

2.6 MatchPyramid

MatchPyramid [5] models text matching as the problem of image recognition. Firstly, a matching matrix whose entries represent the similarities between words is constructed and viewed as an image. Then a convolutional neural network is utilized to capture rich matching patterns in a layer-by-layer way.

For two texts W and V, first represent the input of text matching as a matching matrix M, with each element M_{ij} standing for the basic interaction, i.e. similarity between word w_i and v_j:

$$M_{ij} = w_i \otimes v_j \tag{7}$$

In this way, we can view the matching matrix M as an image, where each entry (i.e. the similarity between two words) stands for the corresponding pixel value. And different kinds of $\otimes$ can be adopted to model the interactions between two words, leading to different kinds of raw images.

Then M can be input to CNN to extract different levels of feature maps z from different layers. And a MLP (Multi-Layer Perception) is used to produce the final matching score:

$$(s0, s1)^T = W_2\sigma(W_1 z + b1) + b_2 \tag{8}$$

3 Experiments

In this section, we first describe the datasets and then the methods we used.

3.1 Datasets

ClueWeb12-B13 was utilised for the decision track. It was created to support research on information retrieval and related human language technologies. The dataset consists of 733,019,372 English web pages, collected between February 10, 2012 and May 10, 2012. ClueWeb12 is a companion or successor to the ClueWeb09 web dataset. Unlike previous tracks, the assessors will be provided the topic query and narrative. And the topics will be provided as XML files.

The dataset are shown as the following examples:

```
WARC/1.0
WARC-Type: response
WARC-Date: 2012-02-10T21:51:20Z
WARC-TREC-ID: clueweb12-0000tw-00-
00013WARC-Target-URI: http://cheap
costhealth insurance.com/2012/01/2
5/what-is-hiv-aids/
WARC-Payload-Digest: sha1:YZUOJNSU
MFG3JVUKM6LBHMRMMHWLVNQ4
WARC-IP-Address: 100.42.59.15
WARC-Record-ID: <urn:uuid:74edc71e
-a881-4942-81fc-a40db4bf1fb9>
Content-Type: application/http; ms
gtype=response
Content-Length: 71726

HTTP/1.1 200 OK
Date: Fri, 10 Feb 2012 21:51:22
Server: Apache/2.2.21 (Unix) mod_s
sl/2.2.21 OpenSSL/0.9.8e-fips-rhel5
mod_auth_passthrough/2.1 mod_bwlimi
ted/1.4 Front Page/5.0.2.2635 mod_
jk/1.2.32
X-Powered-By: PHP/5.2.17
X-Pingback: http://cheapcosthealth
insurance.com/xmlrpc.php
Link: <http://cheapcosthealthinsur
ance.com/?p=711>; rel=shortlink
Connection: close
Content-Type: text/html; charset=
UTF-8
<!DOCTYPE html PUBLIC "-//W3C//DTD
XHTML 1.0 Transitional//EN" "http:
//www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/199
9/xhtml" dir="ltr" lang="en-US">
<title>What is HIV Aids | Health
and Insurance</title>
...
</html>
```

3.2 Preprocessing

3.2.1 Some attempts

First, we read data with warcat directly. However, the web page enters the folder named after the URL, divided by / after decompression. It is hard to index them with ”ClueWeb12_B13_DocID_To_URL.txt.bz2”. Because the file names are often modified when unzipping. For example, the ”//” is replaced by”/_index_da39a3/_index_da39a3”, ”/?” is replaced by ”/_index_da39a3_”, ”=http://” is replaced by ”=http___”, and so on. What’s worse, the modification doesn’t happen all the time. Without the knowledge of decompression coding, we can only change the name by trial and error.

The size of ClueWeb12-B13 is 382GB after compressed. After uncompressed the size of it is 1.95TB, with a subset about 50 million pages. Since the data is extremely large, we try to use other methods.

3.2.2 Decompressing

We use gunzip[6] command together with warcio.archiveiterator[7] toolbox instead .

We firstly extract[8] the xxxx.warc.gz file to a xxxx.warc file. Then we use the following python code to extract the warc file into an .html file.

```
from warcio.archiveiterator import
ArchiveIterator
for record in ArchiveIterator
(stream):
    filename =record.rec_headers.
    get_header('WARC-TREC-ID')
    tr = record.content_stream()
    text = str(tr.read())
```

3.2.3 Convert

We need to convert the HTML document we extracted to XML. Here we use Python to write scripts for processing. HTML documents can be read by calling the library BeautifulSoup, but the data format is confusing and not the text of the web page. So we call html2text to extract the text from the previous cluttered string, and the data is in markdown format. In order to reduce the impact of hyperlinks on the retrieval results, we use the following regular expressions to remove the hyperlinks:

```
text = re.sub(r'\(http[\s\S]*?\)',
'', text)
```

Ultimately, our XML document retains the head and body content of the original html, which is implemented by calling minidom.

3.3 Processing

In this part, we use Terrier as a search tool. Firstly, the document is preprocessed, the HTML document is converted into the required XML format, and then the index is established. Finally, we get the results about the given topics. BM25 is used as our retrieval method.

4 Conclusion

We also tried other models, such as DRMM, Match Pyramid. We didn’t adopt them in the end due to the difficulty of data processing and the large amount of data. Because of the time to build the index and process the data, we finally submitted the retrieval results only on some of the datasets. In the future, we will try the PageRank method to take advantage of a large number of web links in the data. We will also improve the performance of the model to better adapt to large-scale data.

References

[1] TREC group. Trec decision track, 2019. https://trec-decision.github.io/.

[2] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338. ACM, 2013.

[3] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire 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.

[4] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce 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.

[5] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[6] Free Software Foundation. Gnu free codumentation, 2019. https://www.gnu.org/software/gzip/manual/gzip.html.

[7] Webrecorder. Warcio: Warc (and arc) streaming library, 2017. https://github.com/webrecorder/warcio.

[8] Python Software Foundation. Warcat, 2019. https://pypi.org/project/Warcat/.