UCAS at TREC-2014 Microblog Track

Qingli Ma, Ben He, Dongxing Li
School of Computer and Control Engineering
University of Chinese Academy of Sciences
{maqiingli13, lidongxing12}@mails.ucas.ac.cn, benhe@ucas.ac.cn

Abstract

University of Chinese Academy of Sciences (UCAS) participated in the first task, namely temporally-anchored ad hoc retrieval in Microblog track, aiming to efficiently and effectively retrieve tweets. Based on the conventional application of learning to rank, we incorporated a machine learning approach, such as logistic regression for selecting high-quality training data for improving the effectiveness. Except for the tweets' content features, we also used the features of the web information, external evidence, which is related with the URLS to improve the effectiveness.

1 Introduction

The TREC 2014 Microblog track is the fourth iteration of the track. In addition to temporally-anchored ad hoc retrieval (same as last year), this year's track will consist of an additional new task: tweet timeline generation (TTG). Similar to last year's track, in the real-time ad-hoc task the systems are requested to produce a list of recent and relevant tweets starting from the query was issued. UCAS only participated in the first task. The TREC 2014 Microblog track used the same collection of tweets as last year (the Tweets2013 collection). It contains 243 million tweets gathered from the (sampled) public Twitter stream from February 1, 2013 to March 31, 2013 (inclusive, UTC time). Just like last year, participants can adopt the "evaluation as a service" (EaaS) model. And participants can interact with the tweet collection via a search API.

In the previous years, quite a few researches have attempted to apply learning to rank to Twitter search[2]. By using learning to rank, multiple intrinsic features of Twitter, such as user authority, mentions, retweets, hashtags and recency can be combined to learn a ranking model [1].

We, University of Chinese Academy of Sciences (UCAS), submitted four runs for ad hoc search task, which were configured differently by selecting training data to train the learning to rank models and by using the web information to re-rank the search results. The four runs are named as UCASRun1, UCASRun2, UCASRun3 and UCASRun4 whose configurations are described as follows:

- UCASRun1 used the whole 2013 data to train the ranking model without the web information. We used the training model to experiment on the Tweets14 data collection.
- UCASRun2 used the whole 2013 data to train the ranking model with the web information. We experiment on the Tweets14 data collection using the training

model.

- UCASRun3 selected the part of the 2013 training data, namely high-quality training data by machine learning algorithm, without the web information, to train the model which is used to experiment on the Tweets14 data collection.
- UCASRun4 selected the part of the 2013 training data, namely high-quality training data by machine learning algorithm, with the web information, to train the model which is used to experiment on the Tweets14 data collection.

The rest of the paper is organized as follows. Section 2 introduces the data pre-processing, indexing strategy and the language filter. Sections 3 gives a detailed introduction of the experimental process. Section 4 presents the experimental results and analysis. Finally, Section 5 concludes our experiments.

2 Pre-processing and Indexing

The corpora used in our experiments is in the format of HTML. We experiment on the Tweets14 data collection, which has been introduced in the introduction. All fields are marked as Store. In the index, allowing users to access data from retrieved documents. Some fields are present in all statuses, while others only contain a value if the source JSON object contained a non-null entry in that slot. The details are id, screen name, epoch, text, retweeted count, followers count, statuses count, lang, in reply to status id, in reply to user id, retweeted status id, retweeted user id.

Before using it, we first should remove the retweets, repeated tweets and non-English tweets. In the reprocessing, if the retweets were started with RT and there was not any comments, then we filtered the tweets. Some tweet had both English and Non-English words, as long as the English part is relevant then the tweet is relevant, so we reserve the tweet.

Secondly, we should convert the corpora to the TREC format. In particular, in TREC-formatted files, documents are delimited by<DOC></DOC> tags, as in the following example:

```
<DOC>
<DOCNO> 298244286468726788 </DOCNO>
<AUTHOR> TimInThe419 </AUTHOR>
<TIME> Sun February 3 02:08:32 +0000 2013 </TIME>
<AT> </AT>
<BODY> The water has caused a shortage </BODY>
<RTAT> </RTAT>
<RT> </RT>
</DOC>
```

In the above example, DOCNO is the tweet id; AUTHOR is the author of the tweet; TIME is the posted time of the tweet; AT contains all the mentioned users in the tweet, except those occurring in RT tweet; RT is the reposted tweet; RTAT indicates the author from which the tweet is retweeted; BODY means the remaining tweet content after removing AT, RTAT, RT.

In our experiments, we build an individual index for each query using an in-house version of Terrier [5]. Both direct index and inverted index are built to support retrieval and query expansion. Standard stop-words removal and Porter's stemmer are applied in our experiments.

The web information should be preprocessed in the same way of the tweets.

3 Experimental Process

In this section, we will give an introduction of the training data selection approach for learning to rank [8] based on the estimation of the retrieval performance. The basic idea of our approach is to directly estimate the retrieval performance gain by learning weak ranking models using the individual training queries. A linear relationship of a set of query features and the estimated retrieval performance gain is estimated to learn a query quality classifier that selects training queries out of many.

More specially, the general ranking model is learned from the 2013 microblog queries. To rank the relevance, we use the learning to rank technique, which was successfully used in last several tracks. The setting of the parameters are obtained by training on the official queries of 2013 Microblog track.

3.1 General Ranking Model

The common features used to represent the tweets and the learning to rank algorithm will be described in this section.

It is of great importance to select the appropriate feature set to generate a good ranking function in the learning to rank systems. In our experiments, the features are organized around the basic entities for each query-tweet pair to distinguish between the relevant and irrelevant messages. More specially, five types of features are exploited, namely content-based relevance, content richness, authority, recency and Twitter specific features, which were used in our previous work [6].

Many learning to rank approaches have been proposed in the literature, which can be applied for learning the general ranking model. In the experiments, we adopt the pair-wise learning to rank algorithm RankSVM [11,12], which applies the traditional formulation of the SVM optimization problem by taking the document pairs and their preferences as the learning instances.

In the learning process, after the positive and negative examples are appended to the labeled set by making use of the relevance assessments information, we empirically assign preference values according to the temporal distance between the timestamps of the tweet and the query. The larger the preference value is, the higher the tweet is relevant to the given query. This labeling strategy is mainly due to the fact that recency is a crucial factor of relevance in real-time Twitter search. The fresh tweets are favored over those outdated.

The target values of RankSVM define the order of the examples of each query. We reassign the target values of the relevant tweets with an interval of 0.5 which is obtained by training on the official queries of 2013 Microblog track , according to the temporal distance in days between the timestamps of the tweet and the query.

3.2 Description of Experiment

The UCASRun1 is our baseline in our experiment. We use the RankSVM, a learning to rank approach to re-rank the results.

In UCASRun2, we add the information of web, and choose a linear interpolation between the score of tweets and the related web information.

In UCASRun3, we learn a linear relationship of a set of queries with the quality of a training query for learning to rank. We use the machine learning algorithm to train a query classifier to automatically select high quality queries, so the selected queries are used as the training data for learning to rank.

UCASRun4 are added web information based on the UCASRun3. So it's easy to understand.

4 Experimental Results

We submitted four official runs which had been introduced.

In our submitted runs, we issue a retrieval score for each returned tweet to represent the probability of relevance to the query. We examine to selecting high-quality training data to learn to rank approach is able to improve the retrieval effectiveness in Table1.

Metrics	UCASRun1	UCASRun2	UCASRun3	UCASRun4
MAP	0.4384	0.4509	0.4703	0.4515
P@30	0.6303	0.6364	0.6697	0.6400

According to the results, the two runs with the training query selection (Runs 3 & 4) outperform the two baseline runs, indicating the benefit brought by this new approach.

5 Conclusions and Future Work

We incorporate a machine learning approach for selecting high-quality training data for improving the effectiveness of learning to rank for the real-time Twitter search. Such a ranking model is combined with a general ranking model given by the conventional learning to rank approach to produce the final ranking of the Twitter messages, namely the tweets, in response to the user information need. Our preliminary experiments on Tweets14 show that our proposed training data selection approach is able to outperform the conventional application of learning to rank algorithm. Besides, the web information contributes to the retrieval effectiveness.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (61103131/61472391), Beijing Natural Science Foundation (4142050), and SRF for ROCS, SEM.

References

1. M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring user influence in twitter: The million follower fallacy. In ICWSM, 2010.

2. I. Ounis, C. Macdonald, J. Lin, and I. Soboroff. Overview of the TREC 2011 microblog track. In TREC, 2011

3. D. Metzler and C. Cai. Usc/isi at trec 2011: Microblog track. In TREC, 2011.

4. T. Miyanishi, N. Okamura, X. Liu, K. Seki, and K. Uehara. Trec 2011 microblog track experiments at kobe university. In TREC, 2011.

5. I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A high performance and scalable information retrieval platform. In SIGIR OSIR, 2006.

6. X. Zhang, B. He, and T. Luo. Transductive learning for real-time Twitter search. In ICWSM, 2012.

7. K. Duh and K. Kirchhoff. Learning to rank with partially-labeled data. In SIGIR, pages 251–258, 2008.

8. X. Zhang, B. He, T. Luo, and B. Li. Query-biased learning to rank for real-time twitter search. In CIKM, pages 1915–1919. ACM, 2012.

9. C. Zhai and J. D. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In CIKM, pages 403–410, 2001.

10. G. Amati, G. Amodeo, M. Bianchi, A. Celi, C. D. Nicola, M. Flammini, C. Gaibisso, G. Gambosi, and G. Marcone. Fub, iasi-cnr, UNIVAQ at TREC 2011. In TREC, 2011.

11. T. Joachims. Optimizing search engines using clickthrough data. In KDD, pages 133–142. ACM, 2002.

12. W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. Journal of Artificial Intelligence Research, 10:243–270, 1998.

13. V. N. Vapnik. An overview of statistical learning theory. In IEEE Transactions on Neural Networks, pages 988–999. 1999.

14. J. Rocchio. Relevance feedback in information retrieval. In Prentice-Hall Englewood Cliffs, 1971.

15. R. El-yaniv and D. Pechyony. Stable transductive learning. In COLT, pages 35 – 49,
2006.

16. http://olivo.net/software/lc4j/