UCAS at TREC-2013 Microblog Track

Dongxing Li, Ben He, Xin Zhang, and Tiejian Luo

School of Computer and Control Engineering
University of Chinese Academy of Sciences
{lidongxing12, zhangxin510}@mails.ucas.ac.cn, {benhe, tjluo}@ucas.ac.cn

Abstract. The participation of University of Chinese Academy of Sciences (UCAS) in the real-time adhoc task in Microblog track aims to evaluate the effectiveness of the query-biased learning to rank model, which was proposed in our previous work. To further improve the retrieval effectiveness of learning to rank, we construct the query-biased learning to rank framework by taking the difference between queries into consideration. In particular, a query-biased ranking model is learned by a cluster classification learning algorithm in order to better capture the characteristics of the given queries. This query-biased ranking mode is combined with the general ranking model (BM25 etc.) to produce the final ranked list of tweets in response to the given target query. We were also planning to incorporate a machine learning approach for selecting high-quality training data for improving the effectiveness of learning to rank. However, due to interruption caused by lab move, we only managed to experiment with the query-biased approach using partial features.

1 Introduction

This year is the third year of the Microblog track, whereby a user's information need is represented by a query at a specific time. Similar to last year's track, in the real-time adhoc task the systems are requested to produce a list of recent and relevant tweets starting from the query was issued. As in this year, it is not possible to acquire the whole collection, our experiments are based on the tweets obtained using the TREC 2013 Official API.

Recently, quite a few research has 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].

In our experiments about Tweets2011 dataset, we adopt a query-biased learning to rank approach by integrating a general ranking model with the query-biased model that takes the query differences into account [8]. In the combined framework, the general ranking model is learned from the 2011 and 2012 microblog queries by the conventional learning to rank approach. Finally, the query-biased model is combined with the general model to produce the final tweet ranking for the target queries.

The rest of the paper is organized as follows. Section 2 introduces the data pre-processing, indexing strategy and the language filter. Sections 3 givens a detail introduction of the query-biased learning to rank framework. Section 4 presents the experimental results and analysis. Finally, Section 5 concludes our experiments and suggests future research directions.

2 Pre-processing and Indexing

The corpora used in our experiments is in the format of HTML. We experiment on the Tweets13 data collection, which spans over a period of two months from 1th February 2013 to 31th March 2013. We successfully collected 259,057,269 tweets (statuses) via the Twitter streaming API using the feeds distributed by the track organizers in 2013. All fields are marked as Store.Yes 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 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 stopword removal and Porter's stemmer are applied.

For the language filter, the LC4j package is used to detect whether a tweet is in English or not. It is a language categorization library designed for the Java programming language. It has been designed to be a compact, fast and scalable Java library that implements the algorithms to identify languages using n-grams [16]. In our runs, the detected non-English tweets are removed.

3 Query-biased Learning to Rank

In this section, we will give a detail introduction to the query-biased learning to rank approach [8] that utilizes both the common features of Twitter messages and the query-specific aspects that differentiate between queries. More specially, the general ranking model is learned from the 2013 microblog queries and the query-biased model is learned from the query-specific features. Then the two models with a learning rate are linear combined to produce a final tweet list for each given topic.

$$Score_{final}(d, Q) = Score_{LTR}(d, Q) + \beta \cdot Score_{QLTR}(d, Q) \tag{1}$$

where $Score_{final}(d, Q)$ is the final score of tweet d for the given query Q; $Score_{LTR}(d, Q)$ is the score given by the general ranking model; $Score_{QLTR}(d, Q)$ is the score given by the query-biased model. The setting of the parameter β is obtained by training on the official queries of 2011 and 2012 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 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 2011 and 2012 Microblog track , according to the temporal distance in days between the timestamps of the tweet and the query.

```
Input
    D: initial retrieved tweets returned by a content-based
    retrieval model for a batch of query topics
    N: the maximum number of terms to be selected
    to represent a document

Output
    W: the selected term set

Method
    Do the following for each query:
        (1) Get R, the top-k ranked tweets from D
        (2) Compute the KL divergence weight for each
        unique term in R
        (3) Extract the N terms with the highest weights as
        the term features
        (4) Represent each tweet retrieved for the given query
        with the selected terms and their KL divergence
        weights
```

Fig. 1. The tweet representation algorithm.

3.2 Query-biased Ranking Model

Query-specific Tweet Representation Since the purpose of the query-biased modeling is to utilize the query-specific characteristics to boost the retrieval performance, it is a challenging issue to select the appropriate features that are unique to the given queries to represent the tweets. We choose to represent the tweets by the most informative terms in the pseudo relevance set, namely the top-ranked tweets in the initial retrieval. As queries are different to each other in their topical concepts, it is a natural choice to represent the query-specific aspects by the most weighted terms in the pseudo relevance set, which are usually assumed to be highly related to the query topics.

Figure 1 provides the algorithm used for extracting the term features for the query-specific tweet representation. In particular, all the unique terms in the top-30 tweets are taken as candidate terms, and the 10 terms with highest KL divergence weights are chosen as the query-specific features. Thus, the selected words and their corresponding KL divergence weights are used as attributes and values to represent the given tweets. Our arbitrary choice of selecting the top-10 terms from the top-30 tweets is mainly due to the fact that this setting was found to provide the best query expansion effectiveness in the TREC 2011 Microblog track, as reported in [10]. The KL divergence weight of a candidate term t in the top-k ranked tweets in the initial retrieval is computed as follows:

$$w(t, R_k) = P(t|R_k) \log_2 \frac{P(t|R_k)}{P(t|C)} \tag{2}$$

where $P(t|R_k)$ is the probability of generating the candidate term t from the set of top-k ranked tweets R_k, and $P(t|C)$ is the probability of generating t from the entire collection C.

The algorithm for document representation is shown in Figure 1.

4 Experimental Results

We submitted two official runs as follows:

- *UCASqe*: A run using the general learning to rank approach, namely RankSVM [11].
- *UCASgem*: A run using the query-biased learning to rank approach [8].

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 which extent the query-biased learning to rank approach is able to improve the retrieval effectiveness by taking query differences into consideration in Table 1. It turns out that the query-biased learning to rank outperforms the general learning to rank approach when evaluating under Top 30, while there is only minor difference between the MAPs obtained by the two runs.

Table 1. Comparison of UCASqe with UCASgem.

Metrics.	UCASqe	UCASgem
MAP	0.1276	0.1285, +0.71%
P@30	0.2217	0.2844, +28.28%

5 Conclusions and Future Work

We adopt a query-biased learning learning to rank approach that utilizes both the general and query-specific evidence of relevance for the real-time Twitter search. Such a query-biased 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 Tweets13 show that the our proposed combined learning to rank approach is able to outperform the conventional application of learning to rank algorithm.

We were also planning to incorporate a machine learning approach for selecting high-quality training data for improving the effectiveness of learning to rank. However, due to unforeseen interruption caused by the lab relocation, we were only able to experiment with the query-biased approach using partial features. We plan to continue with this line of research in the future.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (61103131/F020511), the President Fund of UCAS (Y15101FY00/Y25102HN00), and the National Key Technology R&D Program of China (2012BAH23B03).

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/`