UCAS at TREC-2012 Microblog Track

Xin Zhang, Sha Lu, Ben He, Jungang Xu, and Tiejian Luo

School of Computer and Control Engineering
University of Chinese Academy of Sciences
{zhangxin510, lusha12}@mails.ucas.ac.cn, {benhe, xujg, tjluo}@ucas.ac.cn

Abstract. Two search tasks, i.e. real-time adhoc and real-time filtering are addressed in this year's Microblog track. The participation of (Graduate) University of Chinese Academy of Sciences (UCAS) in this track aims to evaluate the effectiveness of the query-biased learning to rank model, which was proposed in our previous work. To futher 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, we learn a query-biased ranking model with a semi-supervised transductive learning algorithm so that the query-specific features, e.g. the unique expansion terms, are utilized to capture the characteristics of the target query. This query-biased ranking mode is combined with the general ranking model to produce the final ranked list of tweets in response to the given target query. Experiments on the standard TREC Tweets11 collection show that the query-biased learning to rank approach outperforms strong baselines when evaluated under MAP, namely the conventional application of the state-of-the-art learning to rank algorithm.

1 Introduction

This year is the second year of the Microblog track and two search tasks, namely Real-time Adhoc and Real-time Filtering are addressed, 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 issused. Different from the real-time adhoc task, the real-time filtering task which can be thought of as orthogonal to the real-time adhoc task, aims to decide whether the subsequently posted tweets are response to the query entered at the particular point in time. In the filtering task, the user is interested in the tweets after the querytweettime in order to keep up to date about a specific topic on Twitter.

Recently, quite a few research has attempted to apply learning to rank to Twitter search [2], including supervised [3,4] and semi-supervised methods [6,7,8]. 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, 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 microblog queries by the conventional learning to rank approach. In addition, a query-biased ranking model is learned by a transductive learning algorithm based on the pseudo relevance information. 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 Tweets11 data collection, which spans over a period of two weeks from 24th January 2011 to 8th February 2011. We successfully download 13,401,964 tweets from Twitter.com using the feeds distributed by the track organizers in 2011. Our crawl of Tweets11 consists 10,443,773 tweets after removing the defacto irrelevant tweets that are not fetchable as of 7th May 2012, as the tweets may be deleted or protected after posting. 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> 28968126875963392 <DOCNO>
<AUTHOR> TonyFranceOH </AUTHOR>
<TIME> Sun Jan 23 00:11:54 +0000 2011 </TIME>
<AT> </AT>
<BODY> Oh, my GOD </BODY>
<RTAT> </RTAT>
<RT> if today was a boring slop day </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 indivdual 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 2011 microblog queries and the query-biased model is learned from the query-specific features by a semi-supervised learning algorithm. 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 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]. For the content-based relevance features, we just use DFRee [10], DirichletKL [9] with and without query expansion.

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 arbitrarily reassign the target values of the relevant tweets with an interval of 0.5 [6], according to the temporal distance in days between the timestamps of the tweet and the query.

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.

Transduction for Query-biased Modeling In [8], a query-biased transductive learning algorithm is devised to boost the retrieval performance. Transductive learning [13] is a semi-supervised method for classification by utilizing limited data Using transduction, it is not necessary to generate a model to predict the label of any unobserved point during the process of learning. In our experiments, the query-biased transduction learning algorithm and retrieval are integrated into a learning to rank paradigm. A general description of the proposed method is given in Figure 2. The underlying idea of our proposed method is similar to that of pseudo relevance feedback [14], which assumes a high degree of relevance of the top-ranked documents. Before training, it is only required to predict the labels of a given test set examples [15]. After the initial retrieval using

```
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.

the content-based model, we label, the top-ranked and bottom-ranked tweets are selected as the positive and negative examples, respectively, on the hypothesis that the top-ranked tweets are highly relevant to the given query topic. In contrast, the bottom-ranked are believed to some extent off-topic in their contents. A model is learned to predict the ranking of the remaining unlabeled tweets, from which the very top- ranked and bottom-ranked tweets are appended to the labeled data set. Such a process repeats until the number of iterations exceeds a predefined threshold. Finally, a ranking of all the retrieved tweets are produced by the query-biased model learning by the above transduction process. All the parameters in the proposed query-biased transductive learning algorithm, for example the number of iterations (i.e. N in Figure 2, are obtained by tuning on training queries.

We only apply RankSVM for learning a query-biased model in this paper. This is mainly due to the small amount of pseudo training data available, where the advantage of the listwise approaches such as LambdaMART is negated.

4 Experimental Results

4.1 Real-time Adhoc Task

We submitted four official runs as follows:

- *gucasBasic*: A baseline run using DFRee with query expansion.
- *gucasGen*: A run using the general learning to rank approach, namely RankSVM [11].
- *gucasGenReg*: A run using the general learning to rank approach and simple text matching.
- *gucasQuery*: A run using the query-biased learning to rank approach [8].

```
Input
    D: initial retrieved tweets returned by content-based
    retrieval models for a batch of target queries
    U: a set of unlabeled tweets
    I: a set of labeled tweets, if any
    F: a set of term features representing the tweets
    k: the number of top-ranked and bottom-ranked
    tweets to be initially added to I, if I is empty
    IN: the number of iterations

Output
    L: a ranked list for each query in the target queries

Method
    If I is empty, add the top and bottom-k tweets in D
    into I as positive and negative examples, respectively
   1. Do the following for IN iterations
        (1) Learn a ranking model M from I using F to
        represent the tweets
        (2) Use M to rank tweets in U represented by F
        (3) For all the topics in the training dataset, select
        the very top-ranked and bottom-ranked tweets T
        from U
        (4) Add T to I and remove T from U
   2. Get the best effectiveness and get the
    number of iterations for the batch of queries or per-
    query in the test set
   3. Gradually add the labeled examples for the query
    in the test set
   4. Use the generated data to train a model M
   5. Return a ranked list for each of the target queries by
    applying M
```

Fig. 2. The query-biased transductive learning algorithm.

In our submitted runs, we issuse a retrieval score for each returned tweet to represent the probability of relevance to the query. In the following, we will evaluate the runs at two levels.

- *Comparison of the general learning to rank (gucasGen) with the content-based retrieval models (gucasGen).*
  In our experiments, we carry out a grid seach to select the optimal query expansion setting by adding different numbers of expansion terms from the diverse top-k tweets to the original query.
  We examine if the general learning to rank model (**gucasGen**) is able to improve the retrieval effectiveness over the classical content-based weighting models. From Table 1, we can see that, gucasGen have relatively weak performance when evaluated under MAP, while provide better precision at 30. We suggest that the minor error of the feature extraction of recency and the regularization parameter C for RankSVM, lead to the interesting phenomenon.
- *Comparison of the query-biased learning to rank (gucasQuery) with the the general learning to rank (gucasGen).* We examine to which extent the query-biased learning to rank approach is able to improve the retrieval effectiveness

Table 1. Comparison of gucasGen with gucasBasic.

Metrics.	gucasBasic	gucasGen
MAP	0.1476	0.1344, -9.82%
P@30	0.1763	0.1876, +6.41%

by taking query differences into consideration in Table 2. It turns out that the query-biased learning to rank outperforms the general learning to rank approach when evaluating under MAP, while the precision at 30 makes no difference. It is possibly due to the fact that the large size of the test set while conducting transduction learning is limited to the top-ranked tweets.

Table 2. Comparison of gucasQuery with gucasGen.

Metrics.	gucasGen	gucasQuery
MAP	0.1344	0.1503, +11.83%
P@30	0.1876	0.1876, 0%

4.2 Real-time Filtering Task

Three official runs are submitted as follows:

– *gucasB*: A baseline run using DFRee with query expansion.
– *gucasL1*: A run using the query-biased learning to rank approach [8].
– *gucasL2*: A run using the query-biased learning to rank approach [8] with different parameter settings.

Surprisingly, the effectiveness of the above three runs shows no difference with F=0.1009, Presion=0.0848 and Recall=0.6656, which we will investigate the reasons in the next steps.

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. In particular, a query-biased ranking model is learned by a semi-supervised transductive learning algorithm in order to better capture the characteristics of the given queries. 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. Extensive experiments have been conducted on the standard Tweets11 dataset to evaluate the effectiveness of our proposed approach. Results show that the our proposed combined learning to rank approach is able to outperform strong baseline, namely the state-of-the-art learning to rank algorithms.

Acknowledgements

This work is supported in part by the NSFC Project 61103131/F020511, the President Fund of UCAS (Y15101FY00), 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/`