BJUT at TREC 2014 Microblog Track

Guangyuan Zhang, Zhen Yang, Shuyong Si,*
yangzhen@bjut.edu.cn
College of Computer Science,
Beijing University of Technology, Beijing 100124, China

Abstract—This paper describes the second participation of BJUT in the TREC Micro-blog Track. Two tasks are proposed in this year, including ad hoc search task and tweet timeline generation task. We attended the first task and focused on the method for re-ranking of the returned search results. Specifically, a graph-based method is proposed to re-rank the twitters returned by the official API, namely we re-rank the results by detecting whether some given characteristics are existing or not. Also, we introduce the details of our system, which consists of data preprocessing, system structure, and rank method & results analysis module.

***Index Terms* — Micro-blog retrieval, graph-based method, sentence similarity, re-ranking.**

I. INTRODUCTION

This paper describes the second participation of BJUT in the TREC Micro-blog Track[1]. This year's track focuses on two tasks that are Ad Hoc Search Task and Tweet Timeline Generation Task. We attended the first task and submitted three results. The task is the same as the last year, that all participants should answer a query by providing a list of relevant tweets ranked in decreasing order by predicted relevance score.

Since the whole data set is unknown in this year, we can only receive the 10 thousand tweets at most for each topic through the official API. Therefore in this track, we focus on the re-ranking methods for the received tweets through the API, but we only make use of 1500 tweets that are searching results to re-rank. We perform the experiments on the 2014 TREC Micro-blog data using the same re-ranking method with three sentence similarity computing methods as the value of linking.

II. CORPUS AND SYSTEM

A. Corpus

This year the corpus is the same as the data of 2013 Micro-blog track that is more than an order of magnitude larger than the previously use tweets 2011 collection. Approximately, the corpus consists of 259,057,269 tweets over a two-month period: 1 February, 2013-31 March, 2013 (inclusive). We cannot obtain the whole collection through the official API and only receive 10 thousand relevant tweets for each topic through the official search API[2]. The 10 thousand tweets are encoded by json and composed of tweet id, user id, text and so on. But we only deal with 1500 tweets of one topic with our system to get the result, and the number is an experience point.

B. Pre-Processing

There are four tasks to be done when we deal with one tweet of each topic in our system: 1) extracting the tweet id and text, 2) judge features, including whether or not have http links and re-tweet, 3) removing the http links, and removing the no-English, 4) converting the tweet text to lowercase letters

Firstly, we extract the tweet id and its text of each tweet from the corpus file that contains only a single topic. If a tweet with the label "RT" in the tweets text is found, we will give it a tag such as 1, if not found it would be 0. And the "http" feature is the dame. Then remove the http links, the retweeted_status object and the no-English char. Finally we convert the processed tweet text to lowercase letters. And write the content into a new file. For example we process the tweet that is "id: 0001, text: RT my name is rose, http://www.trec.cn, …" into "id: 0001, text: my name is rose, RT: 1, http: 1".

C. System Construction

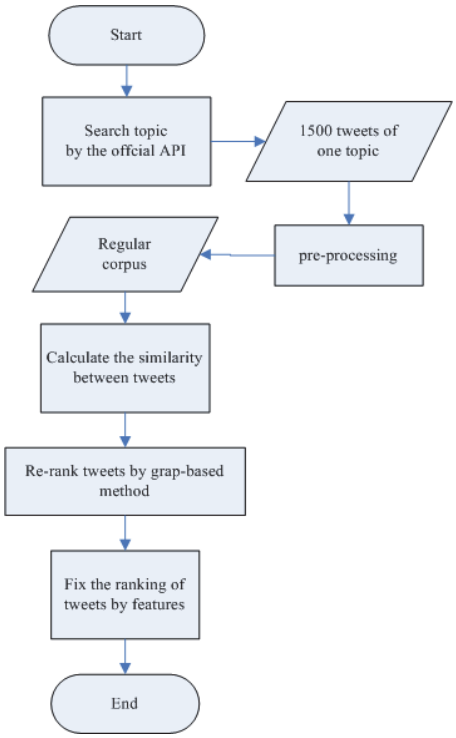

FIG. 1 The Framework of System

As shown in FIG. 1, we performed the experiments on the 2013 TREC Micro-blog data using the official data API and my re-ranking system which consists of data preprocessing, re-ranking of searching results and results analysis module.

As we cannot get the whole collection, we only deal with each topic one by one through our system. When we get the final results, the tweets of different topics are independent of each other. And our system mainly contains three parts as the Fig.1 shows.

- The first chapter is the corpus and preprocessing. We already detailed introduce this part in previous chapter.
- The second part is the re-ranking of search results. It is th e most important part. First of all, we make use of the top 1500 tweets in the new corpus file of one topic to first re-rank through graph-based method, and then we make use of the features form the preprocessing tweet to fix the ord er of the results.
- The final part is the results. The number of result tweets i s 1000 by one topic, and one result contains the topic nu mber, an unused column, a tweet id, the rank, the score an d the run tag such as 'MB171 Q0 307353530413490177 1 2.946289 OSIM'.

III. RE-RANKING

In our system, there are two reorder on tweets. The first time of re-ranking is sorting the whole tweets that have been pre-processed by the graph-based method, that is we treat every tweet as a node and treat the similarity between two tweets as an edge, so they constitute a graph; and the next time is ranking by the order of the features, that is the ranking of tweets that contain the most important characteristic is high, and the ranking of tweets that contain the second important characteristic is following and so on, in addition, the tweets belong to one characteristic is ranked by the score from the first time order.

For every topic we import the top 1500 tweets which are pre-processed through our system into one file which we consider as an initial whole text. All of our three results are based on above hypothesis.

A. Graph-based ranking method

A social network is a mapping of relationships between interacting entities (e.g. people, organizations, and computers). Social networks are represented as graphs, where the nodes represent the entities and the links represent the relations between the nodes. For one topic of micro-blog TREC, the 1500 tweets got through the official API can be viewed as a network of tweets that are related to each other. Some tweets are more similar to each other while some others may share only a little information with the rest of the tweets. Inspired by the work of ref. [3][4], we think that the tweets that are similar to many of the other tweets in the searching results are more central to the topic. So there are two points to clarify in the task. First is how to define similarity between two tweets, and second is how to compute the overall centrality of a tweet given its similarity to other tweets.

a. Define similarity

To define similarity, we use three methods. The first measure of similarity between two tweets is cosine similarity. This given two vectors of attributes, A and B, and the attribute in a tweet is a word which is not English stop word, the cosine similarity[5], cos(θ), is represented using a dot product and magnitude as

$$cos(\theta) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{1}$$

The second measure of similarity between two tweets is dice coefficient[6]. Formula is as follows

$$sim(A, B) = \frac{2C}{A + B} = \frac{2|A \cap B|}{|A| + |B|} \tag{2}$$

where A and B are the number of words in tweets A and B, respectively, and C is the number of words shared by the two tweets.

The third one is improved dice coefficient, as follows

$$msim(A, B) = \frac{C}{B} = \frac{|A \cap B|}{|B|} \tag{3}$$

This method can build the whole tweets into a directed graph.

All three of these can also use the tf-idf of word to create the vector, but this we can use it because we can't acquire it.

b. Compute the overall centrality

A simple way of assessing tweet centrality is to count the number of similar tweets for each tweet. When computing tweet centrality, we have treated each edge as a vote to determine the overall centrality value of each tweet. So a tweet centrality can be computed one time as follow

$$TR(\mu) = \sum_{v \in adj[\mu]} \frac{TR[v]}{\deg[v]} \tag{4}$$

where $TR(\mu)$ is the centrality of tweet μ, $adj[v]$ is the set of tweets that are adjacent to μ, and deg[v] is the degree of the tweet v.

If we assign a uniform probability for jumping to any node in the graph, we are left with the following modified version of Equation (4), which is known as PageRank[7]:

$$TR(\mu) = \frac{\alpha}{N} + (1 - \alpha) \sum_{v \in adj[\mu]} \frac{TR[v]}{\deg[v]} \tag{5}$$

Various studies have tested different damping factors, but it is generally assumed that the damping factor will be set around 0.85.

B. Re-ranking by features

When computing centrality, we have treated each edge as a vote to determine the overall centrality value of each tweet. This is a totally democratic method where each vote counts the same. However, in many types of social networks, not all of the relationships are considered equally important. Centrality may have a negative effect in the quality of the ranking in some cases where several uncorrelated tweets vote for each other and

raise their centrality, and make the correlated tweets get a low order. So we will make use of the features to modify it.

In our system, we get some features, such as whether to forwarding, whether to have http and the similarity between tweet and query. Then we make the ranking of tweets that contain the most important characteristic be high order, and the ranking of tweets that contain the second important characteristic is following and so on, in addition, the tweets belong to one characteristic is ranked by the score from the first time order.

In here, we make the similarity between tweets and query to be the most important characteristic. If a tweet contain the whole words of query, we will mark it a high order, and these tweets that contain the whole words of query are ranked by the score computed through the first method, if some of these tweets have the same score, we will rank them by the characteristic, whether to forwarding and whether to have http.

In this part, we can also make use of more features to improve the ranking. We will do it in the future work.

IV. RESULTS

In this year's TREC Microblog Track, we submit 3 versions of runs which are shown in the Tab. 1.

TABLE 1. RESULTS OF OUR TEAM

Run id	MAP	R-Prec	bpref	P@10	P@20
OSIM	0.1708	0.2207	0.2673	0.4182	0.3682
NSIM	0.1690	0.2169	0.2655	0.3982	0.3536
NCOS	0.1659	0.2198	0.2667	0.3673	0.3255

V. CONCLUSION

In this paper, we describe the details of our methods and system structure. In our system the first run have received the best performance, and this year's performance is much better than last year because the emphasis of the two years is different, because last year focus on query expansion and this year is re-ranking of searching results. This year, our results that are good or bad depends on the search results by the official API. But we also improve the results through other method, including calculating the similarity value of two tweets by tf-idf, and distinguish the linking term of two tweets by features. Then in the coming time, we will make use of these methods to improve the results.

REFERENCES

[1] 2014 Micro-blog Track Guidelines, https://github.com/lintool/twitter-tools/wiki/TREC-2014-Track-Guidelines, 2013

[2] TREC 2013 API Specifications, https://github.com/lintool/twitter-tools/wiki/TREC-2013-API-Specifications, 2013.

[3] G. Erkan, D. R. Radev, LexRank: Graph-based lexical centrality as salience in text summarization, 2004

[4] T. H. Haveliwala, Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search, IEEE Transactions on Knowledge and Data Engineering, 2003

[5] Cosine similarity, http://en.wikipedia.org/wiki/Cosine_similarity

[6] Sørensen–Dice coefficient, http://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient

[7] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web, 1999