PKUICST at TREC 2013 Microblog Track

Runwei Qiang Yue Fei Yihong Hong Jianwu Yang*
{qiangrw, feiyue, hongyihong, yangjw}@pku.edu.cn
Institute of Computer Science and Technology
Peking University,Beijing 100871,China

ABSTRACT

This paper describes PKUICST's entry into the TREC 2013 Microblog track. In this year of microblog track, we designed and conducted a series of experiments based on both our local crawled collection and the official track API. For runs with local crawled dataset, we exploited different retrieval models, such as TFIDF, Okapi BM25 and statistic language model and tuned optimal parameters for all these models with the dataset in TREC 2012. Furthermore, we attempted to combine these models to gain a better performance with the help of learning to rank framework. For runs with the official track API, we employed language model with two-stage pseudo feedback query expansion. In addition, a filtering component was adopted to refine the results retrieved by the expanded query. Experimental results demonstrate that our approach obtains convincing performances.

1. INTRODUCTION

Information retrieval in microblogging environment has attracted a lot of attention with the growing popularity of microblog. To explore the search behavior and boost the retrieval performance in the real-time environment, TREC first introduced Real-Time Search task in 2011 [8], where a user's information need is represented by a query at a specific point in time. Systems should favor relevant and highly informative tweets about the query topic, which makes this task skin to ad-hoc search on Twitter. The primary difference this year from the 2011-2012 lies in the tweet collection and the way the participants interact with it. For TREC 2011-2012, participants are requested to acquire local copy of a canonical corpus and do all the experiments on the local corpus, while for TREC 2013 participants can also interact with a tweet collection stored remotely via a search API. The motivation for this track-as-a-service design is to increase the size of the collection while adhering to Twitter's terms of services. However, this restricts a highly-customized preprocessing of the corpus but only allows a small number of operations such as query modification and re-ranking of results obtained from the baseline retrieval. Thus we also crawled a local copy of the tweets2013 collection with the help of provided official streaming tool [1] and attempted both interactive ways in our experiments.

For runs with our local corpus, we first retrieved top 20,000 relevant tweets as candidates with Real-time Tweet Ranking (RTR) model, and then re-rank the candidate tweets to gain a better performance with the help of learning to rank framework. For runs with the official track API, we employed language model with a two-stage pseudo feedback query expansion. In addition, a filtering component was adopted to refine the results retrieved by the expanded query.

*Corresponding author.
[1]https://github.com/lintool/twitter-tools

2. SYSTEM OVERVIEW

Our system mainly contains two steps : (1) initial search using RTR model, which is based on three classical IR models(e.g. language model), and (2) re-ranking the initial result sets with Ranking SVM [6]. The architecture of our system is shown in Figure 1. Note that for our runs with local copy of the canonical corpus, a preprocessing component is utilized to gain a better retrieval performance.

We first apply a few preprocessings such as stemming, stopwords elimination and non-English detection on both initial corpus and query. The shortened URLs within tweets are very informative as they are always aimed at tracking breaking news stories, recommending interesting video clips and brand marketing [5]. Thus, we extract topic information from the web pages we crawl through these shortened URLs, and form a new corpus called **TopicInfo** Corpus. Another usage of the topic information is directly replacing the URLs in the tweet and generating a new **DE** (i.e. Document Expansion) Corpus.

In the initial search stage, RTR (i.e. Real-Time Ranking) models are utilized to return the top 20,000 candidate tweets for each query. In the re-ranking stage, effective features (i.e. semantic features and quality features) of these candidate tweets are generated so that we could employ the learning to rank framework to re-rank the candidate tweets. Finally, the re-ranked top 1,000 relevant tweets are selected as the final results of our system.

2.1 Preprocessing

The official collection of TREC 2013 consists of approximately 240 million tweets, while our local copy of the canonical corpus contains about 259 million tweets in total, which is really a huge size compared to TREC 2012 collection (about 10 million tweets over two weeks). Both of the corpora are collected via the Twitter streaming API over a two-month period: 1 February, 2013 - 31 March, 2013 (inclusive).

The preprocessings we adopted on the corpus and topic set (i.e. queries) are described as follows:

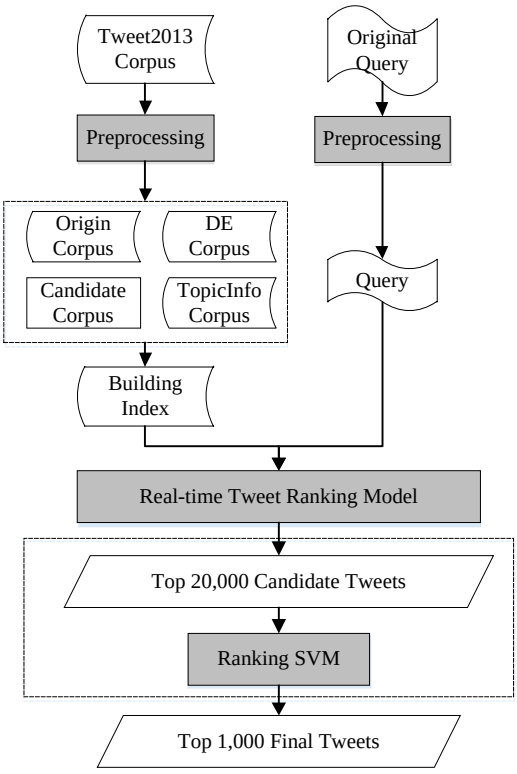

Figure 1: System Architecture

- **Non-English Filtering:** We discarded the non-English tweets by using a language detector with infinity-gram, named ldig[2].
- **Simple Retweet Elimination:** We eliminate tweets that begin with 'RT' with the consideration that these tweets are simple retweets without any additional information.
- **Hashtag Segmentation:** Many tweets are marked with so-called hashtags. A hashtag is a character string preceded by a '#' sign. Hashtags often signal aspects of a tweet's meaning such as its topic or its intended audience [3]. As no space is allowed in a hashtag, we segment them into tokens using an English dictionary generated with the tweets2013 corpus and the MaxMatch algorithm. The segmented words are then added to the original tweet.
- **TopicInfo based Document Expansion:** As mentioned above, we collected all the external URLs (i.e. TopicInfo corpus) contained in TREC 2013 corpus and extracted their title information for our document expansion process. Note that web pages might be deleted as time elapsed, we have only crawled a portion of the external URL set. When adding the topic information to the original corpus, we name the newly generated corpus DE corpus. This expansion is optional as we can get more independent features when computing similarity scores using Origin Corpus or TopicInfo Corpus respectively.

[2]https://github.com/shuyo/ldig

2.2 Retrieval Models

2.2.1 Real-Time Tweet Ranking Model

Given a real-time search problem, the ideal system should consider: (1) build a dynamic dataset for each query to avoid using the future resources; (2) use expansion techniques to enrich the representation of both queries and documents; (3) make a tradeoff between relevance and recentness. To solve these challenges, Liang *et al.* [7] proposed a RTR model, which highlights the following aspects: (1) describe a two-stage pseudo-relevance feedback query expansion to estimate a query language model. (2) propose two ways to expand documents with the shortened URL's information to enrich the representation of the documents and (3) suggest several temporal re-ranking functions and two representations of temporal profile to evaluate the temporal aspect of documents.

In our systems, we implemented different RTR models with three classical IR algorithms for comparison:

- **Vector Space Model** Vector space model is an algebraic model for representing text documents as vectors of identifiers. We express the tweet and query as vectors.

$$\overrightarrow{T_i} = (w_{1i}, w_{2i}, w_{3i}, \cdots w_{ni})$$

$$\overrightarrow{Q_i} = (w_{1q}, w_{2q}, w_{3q}, \cdots w_{nq})$$

  The TFIDF weighting scheme is adopted as the term weight and the Cosine Similarity Metric is used to evaluate the relevance between tweets and query. The Cosine Similarity Metric is defined as Eq.1.

$$Sim = \frac{\overrightarrow{T_i} \cdot \overrightarrow{Q}}{\|\overrightarrow{T_i}\| \cdot \|\overrightarrow{Q}\|} \quad (1)$$

- **Okapi BM25 Model** Okapi BM25 model is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document (e.g., their relative proximity). The similarity of a document D to query Q is defined as Eq.2.

$$Sim = \sum_{q_i \in Q} IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \quad (2)$$

  Where $f(q_i, D)$ is q_i's term frequency in the document D, |D| is the length of the document D in words, and avgdl is the average document length in the text collection from which documents are drawn. k_1 and b are free parameters.

- **Language Model** A statistical language model assigns a probability to a sequence of m words $P(w_1, \ldots, w_m)$ by means of a probability distribution. With query Q as input, retrieved documents are ranked based on the probability that the document's language model would generate the terms of the query.

Next we describe the language model based RTR model in detail. For the other two approaches, we use the same query expansion and document expansion techniques. To rank tweets for a given topic, RTR model estimates the

probability of generating a query Q given the content D and timestamp t of the tweet as follows:

$$P(Q|D,t) = \frac{P(t|Q,D) \cdot P(Q|D)}{P(t|D)} \quad (3)$$

Assume that $P(Q|D) \propto Score(Q,D)$ which can be calculated using Kullback-Leibler retrieval model [10], and that $P(t|D)$ can be assumed as a constant because it is query-independent, the ranking formula can be rewritten as follows:

$$\begin{aligned} P(Q|D,t) &\propto P(t|Q,D) \cdot P(Q|D) \\ &\propto P(t|Q,D) \cdot Score(Q,D) \\ &= P(t|Q,D) \cdot \sum_{w \in V} P(w|\hat{\theta}_Q) \cdot \log P(w|\hat{\theta}_D) \quad (4) \end{aligned}$$

With the ranking formula, the retrieval task is reduced to three subtasks, i.e. the estimation of query model $\hat{\theta}_Q$, the estimation of document model $\hat{\theta}_D$ and the temporal re-ranking component $P(t|Q,D)$, respectively. Considering that this year's task doesn't require participants to rank returned tweets by timestamp, we just implement the estimation of query model and the estimation of document model.

For the estimation of query model, RTR model adopts a two-stage pseudo-relevance feedback query expansion as follows: (1) In the first stage, a single tweet is picked up to generate topical words using the maximum likelihood estimator. (2) In the second stage, a group pseudo-relevant tweets are used to distill the relevant content by implementing the model-based feedback approach [11].

It is important to point out that the single tweet (i.e. *issue tweet*), which is generated in the first stage query expansion can be used to calculate another score with both original tweets and topic information for further semantic representation. Overall, the estimation of query model can be represented as:

$$P(w|\hat{\theta}_{Q'}) = (1-\alpha) \cdot P(w|\hat{\theta}_Q) + \alpha \cdot P(w|\hat{\theta}_{PRF_1}) \quad (5)$$

For the estimation of the document model, RTR model presents two ways to utilize the external resource, i.e. TopicInfo corpus and we use it as *local context* [7]. We merge the original tweet T and topic information I if exists to form a new document and estimate the document language model using Dirichlet Smoothing [10] as follows:

$$P(w|\hat{\theta}_D) = \frac{c(w,D) + \mu P(w|C)}{|D| + \mu} \quad (6)$$

2.2.2 Optimizing classical IR models

Microblogs are extremely short compared to traditional IR documents due to a 140 characters limit, thus it seems intuitive that standard term weighting approaches may not be effective for short documents like microblogs.

Paul *et al.* [4] examined the applicability of term frequency statistics and document length normalization to microblog retrieval. They found that document length normalization always harms performance, and benefit from incorporating term frequency statistics is minor, where term frequency statistics denotes parameter k_1 and document length normalization denotes parameter b in Eq.2. Our experiments also show that the optimal parameters for BM25 model are quite different from typical parameters(k_1 = 1.2, b = 0.75). We get the optimal parameters in TREC 2012 corpus where $k_1 = 0.3$ and $b = 0.05$, indicating that document length normalization and term frequency statistics have little contributions in short text retrieval. For the language model approaches, we use the best empirical parameters reported in [7].

2.2.3 Track as a Service

Since the corpus of TREC 2013 is more than an order of magnitude larger than the previously used TREC 2012 collection, we are provided an official search API to interact with the tweet collection. The search API returns us a ranked list of tweets retrieved by a state-of-the-art IR model (e.g., language modeling). Besides, participants can also access the text, API-supplied metadata from retrieved data and corpus-level statistics.

When applying RTR models with the search API, we only employ the two-stage pseudo relevance query expansion component while ignoring highly-customized preprocessing and document expansion as we are not allowed to access the official corpus directly. On the other hand, a filtering component was adopted to refine the tweet list to compensate for the different approach in preprocessing. Our filtering strategies are described as follows:

- Discard non-English tweets with the help of language detector ldig.
- Remove the simple retweeted tweets beginning with 'RT' based on the assumption that such tweets has no extra information beyond the original ones.

2.3 Learning to Rank Framework

Learning to rank is a data-driven approach which integrates a bag of features in the model effectively [2]. Our system adopts the same framework that Duan *et al* [2] proposed except that we employ different features for the learning algorithm. The basic learning to rank framework is shown in Figure 2.

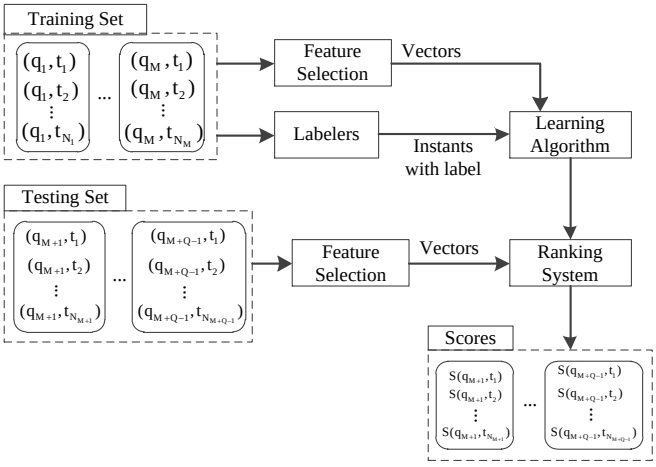

Figure 2: Learning to rank framework

In order to train an effective model, adequate training data and useful feature set are required. The candidate tweets are produced by the language model based RTR

Table 1: Training Data Relevance Distribution

Category	# of Tweets	Crawled
Minimally-Relevant	6,286	6,223
Highly-Relevant	2,572	2,549
Non-Relevant	66,787	56,838
Total	73,073	63,061

model described in section 2.2.1. Our training set is generated from official result set of TREC 2012 Microblog Track, the relevance distribution in the result set is listed in Table 1. Our system trains Ranking SVM [6] as learning to rank model.

Several features have been proved effective in the prior work [2]. In our experiments, two groups of features are utilized in our learning to rank approach: semantic features and quality features.

Semantic features

Semantic features refer to the features that describe the relevance between the query and tweets, such as the Kullback-Leilbler divergence between query model and document model. RTR model with different similarity algorithms can generate different semantic features, such as Kullback-Leilbler divergence and BM25 similarity. Using different query or document model can generate different features that may reflect different aspects of query-document similarity. For example, using TopicInfo Corpus, we may get the relevance between the tweet link and user's query while using Origin Corpus, we can get the content relevance between the query and the tweet text.

In our approaches, we propose four semantic features.

- **OriginTweetScore** score generated by the RTR Model with the original query and Origin Corpus
- **OriginTitleScore** score generated by the RTR Model with the original query and TopicInfo Corpus
- **IssueTweetScore** issue tweet is the highest-scored tweet retrieved in the first stage of RTR Model that is described in section 2.2.1, IssueTweetScore is generated by the RTR Model with this issue tweet, as a new query and Origin Corpus.
- **IssueTitleScore** like IssueTweetScore, generated by the RTR Model with the issue tweet and TopicInfo Corpus.

Quality Features

Many of the microblogs are not informative or have very little content due to their personal and ephemeral nature. Providing effective retrieval in a microblog service will require addressing the challenge of distinguishing the high-quality, informative documents from the others [1].

Recent work has focused on finding features that indicate the quality of microblog documents. For example, tweets containing a URL indicate that more information can be referred on the linked page. Besides, a tweet usually includes a group of well-defined symbols which indicate the social interactive behaviors between different users [9]. Those symbols are described as follow:

- 'RT' symbol denotes re-tweet. A tweet with symbol RT reflects that more users are interested in the topic talked about in this tweet.
- '@' symbol followed with a user's screen name stands for mentions and replies. A tweet with more '@' means this tweet may attract more persons' attention.
- '#' symbol (i.e. hashtag) is used for organizing tweets into a particular topic. A symbol '#' marks the tweet as belonging to a particular topic.

Thus, in our system we use retweet count, mention count and hashtag count as our quality features.

3. RESULT ANALYSIS

Table 2 shows the retrieval performance of our submitted four runs. The primary evaluation measures for this year's task are still P@30 (Precision at 30), MAP (Mean Average Precision) and R-Prec(R-Precision). Our training metric in optimizing the RTR model and learning to rank framework is P@30.

Table 2: Performance of our submitted runs

Run_ID	P@30	MAP	R-Prec
PKUICST1	0.5478	0.3351	0.3721
PKUICST2	0.5311	0.3268	0.3637
PKUICST3	**0.5567**	**0.3486**	**0.3827**
PKUICST4	0.5017	0.2768	0.3260

PKUICST1 uses learning to rank framework, adopting RTR scores as semantic features and all the quality features. PKUICST2 and PKUICST3 are both from the top 1,000 candidate tweets from the RTR model on local DE corpus, the former employs BM25 model while the latter is based on language model. PKUICST4 is the only run that relies strictly on data obtained from the official track API.

From the experimental results, we can first see that PKUICST4 is slightly worse than local corpus runs, which points out the significance of preprocessing and document expansion. Compared with PKUICST2, PKUICST3 achieves 4.43% and 6.67% further increase in P@30 and MAP, respectively. RTR model based on language model still performs better than the one based on BM25 model, perhaps BM25 model becomes average on short text for the weakening of parameters.

However, we didn't gain any improvements with the learning-to-rank approach, which turned out to be the best one in training. This might be caused by the different corpus we used for training and testing, and further investigation is still needed for this issue.

4. CONCLUSION AND FUTURE WORK

In this paper, we present our system for TREC 2013 Microblog Track. For our local corpus, we adopt Real-time Tweet Ranking (RTR) model to rank the tweets to the given topic, and meanwhile the RTR model provides candidate tweets to learning to rank framework for the further ranking process. For data obtained from the official search API, we employed a two-stage pseudo feedback for to query expansion. Then a filtering component was adopted to refine the results retrieved by the expanded query. Many studies

remain for the future work. One of the most interesting directions is to improve learning to rank framework for better results. Moreover, we are also interested in how to adapt classical IR models to short text retrieval.

5. ACKNOWLEDGMENTS

We thank Feifan Fan and Chao Lv for technical assistance in this year's microblog track. The work reported in this paper was supported by the National Natural science Foundation of China Grant 61370116.

6. REFERENCES

[1] Jaeho Choi, W Bruce Croft, and Jin Young Kim. Quality models for microblog retrieval. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1834–1838. ACM, 2012.

[2] Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and Heung-Yeung Shum. An empirical study on learning to rank of tweets. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING*, pages 295–303. Tsinghua University Press, 2010.

[3] Miles Efron. Hashtag retrieval in a microblogging environment. In Fabio Crestani, Stéphane Marchand-Maillet, Hsin-Hsi Chen, Efthimis N. Efthimiadis, and Jacques Savoy, editors, *SIGIR*, pages 787–788. ACM, 2010.

[4] Paul Ferguson, Neil OHare, James Lanagan, Owen Phelan, and Kevin McCarthy. An investigation of term weighting approaches for microblog retrieval. In *Advances in Information Retrieval*, pages 552–555. Springer, 2012.

[5] Bernard J. Jansen, Mimi Zhang, Kate Sobel, and Abdur Chowdury. Micro-blogging as online word of mouth branding. In Dan R. Olsen Jr., Richard B. Arthur, Ken Hinckley, Meredith Ringel Morris, Scott E. Hudson, and Saul Greenberg, editors, *CHI Extended Abstracts*, pages 3859–3864. ACM, 2009.

[6] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[7] Feng Liang, Runwei Qiang, and Jianwu Yang. Exploiting real-time information retrieval in the microblogosphere. In *JCDL*, pages 267–276, 2012.

[8] Iadh Ounis, Craigand Macdonald, Jimmy Lin, and Ian Soboroff. Overview of the TREC-2011 Microblog Track. In *Proceedings of TREC 2011*, 2012.

[9] Runwei Qiang, Feng Liang, and Jianwu Yang. Exploiting ranking factorization machines for microblog retrieval. CIKM '13, pages 1783–1788. ACM, 2013.

[10] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.

[11] ChengXiang Zhai and John D. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM*, pages 403–410. ACM, 2001.