PKUICST at TREC 2012 Microblog Track

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

ABSTRACT

This paper describes the PKUICST's entry into the TREC 2012 Microblog track. In this year of microblog track, we participate in both the Real-time Adhoc Task and Real-time Filtering Task. In the Real-time Adhoc Task, we designed and conducted a series of experiments based on different retrieval models, namely Real-time Tweet Ranking (RTR) model and learning to rank framework. In the Real-time Filtering Task, we adopted various strategies to determine the filtering threshold. Official results demonstrate that our approach obtains convincing performances and more unofficial runs lead to some further conclusions.

1. INTRODUCTION

The popularity of microblog has significantly increased information seeking behaviors in the microblogging environments. To explore the search behavior and boost the search performance in the real-time environment, TREC introduced a novel pilot track named microblog track last year. In this year of microblog track, two tasks are introduced, namely Real-time Adhoc Task and Real-time Filtering Task, whereby a user's information need is represented by a query at a specific time. In the real-time adhoc task, the user wishes to see the most recent but relevant information to the query. However, different from last year, participants are required to return top 10,000 tweets prior to the query time per topic according to their relevance score. Hence, systems should favor relevant and highly informative tweets about the query topic, which makes this task skin to ad-hoc search on Twitter. In this task, we adopt both RTR model and state-of-art learning to rank framework to improve the retrieval effectiveness.

The real-time filtering task aims at deciding if subsequently posted tweets are relevant for a query entered at a particular point in time. In this task, the user is interested in new relevant tweets, thus to keep up to date about a developing topic. The topics used for the real-time filtering task are the same as last year, which provides a way to use supervised methodology. Hence, we train different models from training data and try various strategies to determine the filtering threshold which is of vital importance in the adaptive filtering task.

2. REAL-TIME ADHOC TASK

In this section, we describe our approach for the Real-time Adhoc Task in detail.

*Corresponding author.

Table 1: Summary statistics of Tweet corpus

Html Code	Status	Tweets in 2011	Tweets in 2012
200	OK	13,839,083	8,084,724
302	Found	1,106,999	815,794
403	Not Found	284,225	817,273
404	Forbidden	844,494	868,667
Null	Null	67,011	67,011
Searchable		14,946,082	8,900,518

2.1 System Overview

Our system contains two major steps: initial search using RTR Model [7], which is based on statistic language model and re-ranking the initial result sets with Ranking SVM [5]. The architecture of our system is shown in Figure 1. Corpus and query are processed in parallel.

Our system does some necessary text preprocessing to the corpus and query, such as stemming and stopwords elimination. The links within tweets are very informative as they are always aimed at tracking breaking news stories, recommending interesting video clips and brand marketing [3]. Thus, we crawl all the links and extract topic information from the pages we get[7], forming a new corpus called TopicInfo Corpus. Another way to use topic information is directly replacing the links in the original tweet and generate a new DE(Document Expansion) Corpus. After the retrieval step with the help of RTR Model, the system produces top 20,000 scoring candidate tweets. Then we collect features for these candidate tweets, such as semantic features, tweet-related features and temporal features, then use learning to rank framework to re-rank the candidate tweets. Lastly, we choose the top 10,000 relevant tweets as the final results.

2.2 Preprocessing

Tweet11 corpus was obtained using a donation of the unique identifiers of a sample of tweets from Twitter [10]. We crawled the HTML version copy of the corpus with the provided tools.Table 1 shows basic statistics of our HTML version acquisition on June 23, 2011. Given the corpus and topic set, we do the following preprocessings.

- **Corpus status update:** For the sake of fairness, organizers re-crawled the Tweet11 corpus at the beginning of this year's track, and offered a list of valid IDs for corpus update. We filter all the invalid IDs to generate the Tweet12 corpus.
- **Crawled TopicInfo corpus:** To expand the document representation, we collect all the external URLs

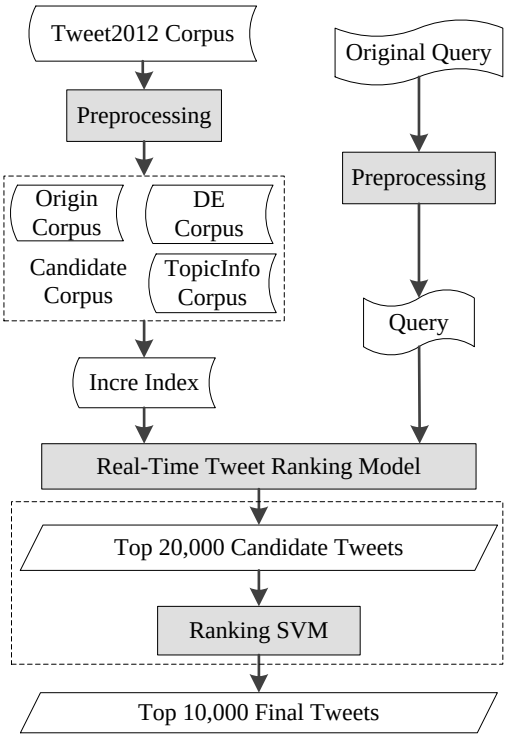

Figure 1: System Architecture

Table 2: Summary statistics of TopicInfo corpus

Html Code	Status	Tweets in 2011
200	OK	1,225,947
302	Found	688
403	Not Found	5,050
404	Forbidden	92,378
Null	Null	265,468
Searchable		1,226,635

(i.e. TopicInfo corpus) contained in Tweet11 corpus and extract their topic information for our document expansion process in early December, 2011.Note that web pages might be deleted as time elapsed, we have only crawled a portion of the external URL set. Summary statistics of TopicInfo corpus is present in Table 2.

- **Non-English filter:** We filter out all tweets that have words encoded with non-ASCII code.
- **Simple retweet elimination:** We eliminate tweets that begin with 'RT' with the consideration that these tweets are simple retweets without any other additional information.

2.3 Retrieval Models

2.3.1 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 query and document; 3) make a tradeoff between relevance and recentness. To solve these challenges, Feng et al. [7] propose a Real-time Tweet Ranking (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 document with the shortened URL's information to enrich the representation of document. 3) suggest several temporal re-ranking functions and two representations of temporal profile to evaluate the temporal aspect of documents.

To rank tweets for a given topic, RTR model is to estimate the probability of generating a query Q given the content D and timestamp t of the tweet as follows:

$$P(Q|D,t) = \frac{P(t|Q,D) \cdot P(Q|D)}{P(t|D)} \quad (1)$$

Assuming that $P(Q|D) \propto Score(Q,D)$ which can be calculated using Kullback-Leibler retrieval model [14], and that $P(t|D)$ can be assumed as a constant because it is query-independent, the ranking formula can be rewritten as follows:

$$\begin{aligned} P(Q|D,t) &\propto P(t|Q,D) \cdot P(Q|D) \\ &\propto P(t|Q,D) \cdot Score(Q,D) \\ &= P(t|Q,D) \cdot \sum_{w \in V} P(w|\hat{\theta}_Q) \cdot \log P(w|\hat{\theta}_D) \quad (2) \end{aligned}$$

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 [15].

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:

$$P(w|\hat{\theta}_{Q'}) = (1-\alpha) \cdot P(w|\hat{\theta}_Q) + \alpha \cdot P(w|\hat{\theta}_{PRF_1}) \quad (3)$$

For the estimation of document model, RTR model presents two ways to utilize the external resource, i.e. TopicInfo corpus. One is to 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 [14] as follows:

$$P(w|\hat{\theta}_D) = \frac{c(w,D) + \mu P(w|C)}{|D| + \mu} \quad (4)$$

Another approach is to smooth the original document model using linear incorporation with the topic information language model estimated based on TopicInfo corpus, and each model is smoothed using Dirichlet method as well. The doc-

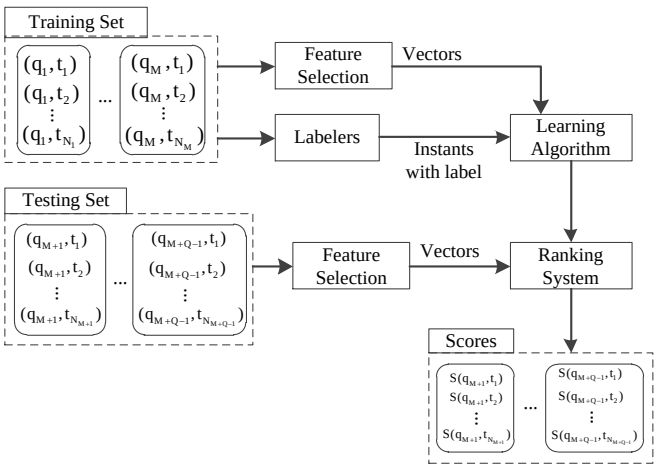

Figure 2: Learning to rank framework

Table 3: Training Data Relevance Distribution

Category	TREC11 Labeled Tweets	Crawled
Minimally Relevant	2306	2075
Highly Relevant	558	511
Non Relevant	37900	33803
Total	40764	36389

ument model is present as follows:

$$P(w|\hat{\theta}_D) = (1-\lambda)\cdot P(w|\hat{\theta}_T) + \lambda \cdot P(w|\hat{\theta}_I) \quad (5)$$

$$P(w|\hat{\theta}_T) = \frac{c(w,T)+\mu_T P(w|C_T)}{|T|+\mu_T} \quad (6)$$

$$thickP(w|\hat{\theta}_I) = \frac{c(w,I)+\mu_I P(w|C_I)}{|I|+\mu_I} \quad (7)$$

2.3.2 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 select different features for the learning algorithm. The basic learning to rank framework is shown in Figure 2.

In order to train an effective model, adequate training data and useful feature set are required. The candidate tweets is produced by the RTR model described in section 2.3.1. Our training set is generated from official result set of TREC'11 Microblog Track, the relevance distribution in the result set is listed in Table 3.Our system trains Ranking SVM [5] as learning to rank model.

Three major types of features are used in our model: semantic features, tweet related features and temporal 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. 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.

- **OrgTweetScore** score generated by the RTR Model with original query and Origin Corpus
- **OrgTitleScore** score generated by the RTR Model with original query and TopicInfo Corpus
- **IssueTweetScore** Issue tweet is highly relevant tweet retrieved in the first stage of RTR Model that is described in section 2.3.1, 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.

Temporal Feature

Recentness is also an important aspect in the microblogosphere. In TREC'11 Microblog Track, participants are required to produce a ranked list of tweets from the latest timestamp to the earliest. Thus, tweets posted an hour ago are often more worthy than those were updated a day before. So we use the normalized time difference between the time when the query issued and the time when the tweet published as temporal feature.

2.4 Result Analysis

In this section, we analyze the results of our approaches. In this year's track, all submitted runs were pooled to depth 100 (while in last year the pooling depth is 30) according to the retrieval scores indicated in each run.

2.4.1 Analysis of Official Runs

Table 4 show the performance values 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 learning to rank framework is MAP.

Table 4: Performance of our submitted runs

Run_ID	P@30	MAP	R-Prec
PKUICST1	0.2164	0.1639	**0.2176**
PKUICST2	0.2068	0.1561	0.2120
PKUICST3	0.2113	0.1686	0.2174
PKUICST4	**0.2333**	**0.2263**	0.2174

PKUICST3 only uses RTR model with the DE Corpus and cuts the top 10,000 tweets from the candidate tweet set. Except PKUICST3, other runs all adopt learning to rank framework. PKUICST1 and PKUICST2's candidate tweet sets are both generated with the DE Corpus, while PKUICST4's candidate tweet set is generated with the Origin Corpus. The difference between PKUICST1 and PKUICST2 is that they train in different training set. The former trains on the 49 allrel topics while the latter trains on the 33 highrel topics. All ranking SVM models use all the semantic features and tweet related features. PKUICST4 doesn't use the OrgTitleScore and IssueTitleScore features as it doesn't use any external resources.

Table 5: Model Description in unofficial runs

Model Name	Candidate Set	Feature Set
OrgBase	Origin Corpus	OrgTweetScore, Tweet Related Features
OrgTime	Origin Corpus	OrgTweetScore, Tweet Related Features, Temporal Feature
DEBase	DE Corpus	OrgTweetScore, Tweet Related Features
DEIssueTitle	DE Corpus	OrgTweetScore, OrgTitleScore, IssueTweetScore, IssueTitleScore, Tweet Related Features

From the evaluation result, we can see that training on allrel topics is better than training on highrel topics. Compared with PKUICST2, PKUICST1 achieves 4.64% and 5.00% further increases in P@30 and MAP, respectively. As we known, the official evaluation used only highly relevant tweets as relevant, however we didn't gain any improvements by training on highrel topics, further investigation is needed for this issue. Origin candidate is even better than the DE candidate according to the official evaluation.

2.4.2 Analysis of Unofficial Runs

In addition to the submitted runs, we also do some complementary experiments on TREC2011 data for comparison. These experiments aim at comparing the selection of candidate tweet sets, semantic features and the effectiveness of temporal feature. All models in the experiments adopt learning to rank algorithm and apply repeated random subsampling validation. The metric used in our learning algorithm is MAP, which is one of the major evaluation measures in TREC'11 microblog track.

The models we compare adopt different candidates or feature sets. The model description is shown in Table 5. The second column describes the corpus used in the RTR Model to generate the candidate tweet set. With the optimum parameters C (trade-off between training errors) in SVM^{rank} [6], we re-rank the candidate tweets and generate the final results.

Table 6: Performance of unofficial runs

Model Name	P@30	MAP	R-Prec
OrgBase	0.4623	0.2914	0.3317
OrgTime	0.4532	0.2902	0.3314
DEBase	**0.4654**	0.2921	0.3319
DEIssueTitle	0.4567	**0.2989**	**0.3355**

The performance of each run is shown in Table 6. The performance of DEBase and OrgBase is basically the same, so is OrgBase and OrgTime. DEIssueTitle gains about 2.3% improvements in MAP score compared with DEBase.

According to these experiments, we conclude that:

- When feature set are determined, candidate set influence a little in the unofficial runs.
- Temporal feature may not be effective in the current framework.
- More semantic features can improve the MAP score to some extent.

The conclusion may not be the same with the ones from the official runs, as the unofficial runs are all tested on the TREC2011 Tweet Corpus. As now the evalation tool for TREC2012 is published, we'll do more experiments for further conclusions.

3. REAL-TIME FILTERING PILOT TASK

This section describes our approach for Real-time Filtering Pilot Task. Filtering differs from searching in that documents arrive sequentially over time. The Real-time filtering task aims at simulating online time-critical tweet filtering applications, which means that potentially relevant tweet must be presented immediately to the user.

3.1 System Overview

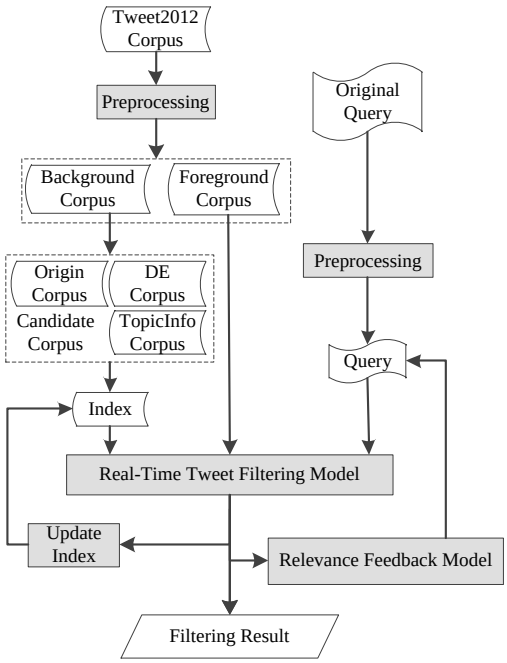

Figure 3: FIltering System Architecture

Figure 3 shows the architecture of our filtering system. In our filtering system, we do the same preprocessing as the Real-time Adhoc Task for the corpus at first. The corpus index is still built with the help of the Lemur IR toolkit[1]. The Filtering Model and the Relevance Feedback Model are the core parts of our system. For each tweet, we use the Filtering Model to estimate its relevance score with respect to the current query and generate the filtering result according to the relationship between the relevance score and threshold obtained in the training phase. We vary the decision threshold to get the best evaluation result based on P@30 evaluation metric. The Relevance Feedback Model aims at expanding the keywords of the query and it can be regarded as an evolvement of topic. To achieve the goal of filtering task, we try different filtering models and feedback

[1]http://www.lemurproject.org/lemur.php

algorithms which will be discussed in detail next. When the filtering action is done, new tweets from the foreground corpus will be added to the background corpus. Thus we can update the index dynamically with the time [8].

3.2 Real-Time Tweet Filtering Model

The filtering model we used in the filtering model will be introduced in this section. For each model, we train the decision thresholds to make evaluation result best based on P@30 evaluation metric in the training stage except the SVM model.

3.2.1 Baseline Models

In the Filtering Model, several traditional retrieval models such as Boolean Model, Language Model and Vector Space Model are applied as the scoring method[9].

Boolean Model

The Boolean Retrieval Model was used by the earliest search engines and is still in use today. And Boolean Retrieval System achieves its goal by judging whether the document contains the keywords of query.

Language Model

Language Model described in the Adhoc task is also applied to the filtering task while we still use the Kullback-Leibler divergence as the relevance score.

Vector Space Model

Vector space model (VSM) is an algebraic model for representing text documents as vectors of identifiers. We express the tweet and the query as vector.

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

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

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.8.

$$Sim = \cos\theta = \frac{\overrightarrow{T_i} \cdot \overrightarrow{Q}}{\|\overrightarrow{T_i}\| \cdot \|\overrightarrow{Q}\|} \tag{8}$$

3.2.2 Two-Stage Filtering Model Combined VSM and Improved Boolean Model

We propose an efficient but simple combination model in the filtering task. Figure 4 describes a Two-Stage Filtering Model which combines Vector Space Model and Improved Boolean Model. The two score thresholds t_c and t_b are obtained in the training phase.

For each tweet, we calculate its relevant score based on the Improved Boolean Model if it survives in the Vector Space Model, which depends on the Cosine Similarity Metric. Here, we use an improved Boolean Model instead of the traditional Boolean Model. Since each tweet contains no more than 140 words, the tweet is likely to be more relevant to the query if it contains a high proportion of keywords. Thus we define its relevance score as Eq.9.

$$Sim(T, Q) = \frac{|\{t|t \in (T \cap Q)\}|}{|Q|} \tag{9}$$

where T and Q denote the term set of the tweet and query.

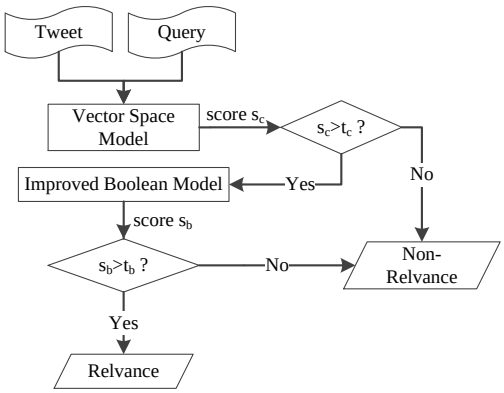

Figure 4: Two-Stage Filtering Model

3.2.3 SVM-based Model

Support Vector Machine (SVM) is a robust machine learning methodology which has been shown to yield state-of-the-art performance for text classification [4]. D. Sculley et al [13] also demonstrates that online SVMs do indeed provide good performance for online spam filtering. Thus we try to combine SVM in our approach to gain a more robust performance in tweet filtering task. In our experiment, libSVM tool developed by Chih-Chung Chang and Chih-Jen Lin [1] is used as our core SVM solver.

Generating machine learning features from text could be done in a variety of ways, especially when the text may include hyper-content and meta-content such as tweet link and hash tag. All the tweet related features mentioned in section 2.3.2 are used in our algorithm. The score generated by the language model described in section 3.2.1 is used as semantic feature. In addition, the query words' average inverse document frequency and the score generated by the Boolean model which is described in Eq.9 are both candidate features.

The SVM tradeoff parameter C must be tuned to obtain the optimal performance. The filtering task provides us 10 topics whose related tweets are classified as high relevant, minimally relevant and non-relevant. To tune our system parameters, five-fold cross validation was used in our experiment to determine the optimum parameter C of SVM.

With the optimum parameter, we classify the query documents pair of the remaining 39 topics as 3 levels: 3(highly relevant), 2(minimally relevant) and 1(non-relevant). These label info can help us filter tweets. Our relevance judging strategy is described as follows: (1) if a tweet is labeled by SVM as minimally relevant or highly relevant, we output yes for this tweet. (2) Else we will judge the tweet by the score calculated from the language model. if the score is higher than the static threshold tuned using the training set, then system outputs yes. (3) Otherwise, outputs no.

To conclude, we consider SVM as a high performance classifier that may merely miss relevant tweets, and these missing tweets are expected to be judged correctly by static score threshold method.

3.3 Relevance Feedback Model

The Relevance Feedback Model aims at expanding the keywords of the query and it can be regarded as an evolvement of topic. And we apply two relevance feedback models

Table 7: Performace of submitted runs

Run_ID	T11SU	F(beta=0.5)	Precision	Recall
PKUICSTF1	**0.3424**	**0.2722**	**0.3963**	0.2300
PKUICSTF2	0.3244	0.2525	0.3701	0.2809
PKUICSTF3	0.3233	0.2556	0.3857	0.2272
PKUICSTF4	0.3341	0.2629	0.3766	**0.2936**

in our filtering system.

3.3.1 Rocchio Algorithm

The Rocchio algorithm [12] is based on a method of relevance feedback found in information retrieval systems which stemmed from the SMART Information Retrieval System. And the Rocchio feedback approach is developed with the help of Vector Space Model. The algorithm is based on the assumption that most users have a general conception of which documents should be denoted as relevant or non-relevant. Therefore, the user's search query is revised to include an arbitrary percentage of relevant and non-relevant documents as a means of increasing the search engine's recall, and possibly the precision as well.

The Rocchio Feedback Algorithm is described as Eq.10.

$$\overrightarrow{Q_m} = \alpha\overrightarrow{Q_0} + \beta\frac{\sum_{T_j \in T_R}\overrightarrow{T_j}}{|T_R|} + \gamma\frac{\sum_{T_l \in T_{NR}}\overrightarrow{T_k}}{|T_{NR}|} \quad (10)$$

where T_R denotes the set of relevant tweets while T_{NR} denotes the set of non-relevant tweets, and Q_m represents the feature vector after m tweets. We set γ as zero since we can only use the non-relevant tweets as background dataset.

3.3.2 Iteration Algorithm

The Iteration Algorithm used in the filtering task is similar to the Rocchio Algorithm. It's also a method of relevance feedback and developed with the help of Vector Space Model. In the feedback stage, terms with low weight will be discarded. The Iteration Algorithm is described as Eq.11.

$$\overrightarrow{Q_m} = \alpha\overrightarrow{Q_{m-1}} + \beta\overrightarrow{T_{m-1}} \quad (11)$$

where α and β are the tuning parameters and the sum of them equals 1.

3.4 Result Analysis

Table 7 show the performance values of our submitted four runs. The primary evaluation measures for the filtering task are T11SU [11], F-score (beta=0.5), P-score (Precision) and R-score (Recall). Our training metric is based on the P30 evaluation metric.

Run PKUICSTF1 uses the two-stage filtering model that combines the VSM and improved Boolean Model. And the Rocchio Algorithm is denoted to Run PKUICSTF1 as the relevace feedback model. Run PKUICSTF3 uses the language model only with a static threshold set as -6. PKUICSTF2 and PKUICSTF4 both apply another two-stage strategy which include SVM classifier and Language Model. The difference between two runs is that Run PKUICSTF4 took the external link resources into consideration and the parameters of these two runs are different.

We'll do more experiments for further conclusions after the publication of evalation tool.

4. CONCLUSION AND FUTURE WORK

In this paper, we present our system for TREC'12 Microblog Track. For the real-time search task, 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 the real-time filtering pilot task, we compare different baseline models and propose the two-stage filtering model which combines VSM model and Boolean model. In addition, we apply two relevance feedback models to improve the filtering results. Many studies remain for the future work. One of the most interesting directions is to improve learning to rank/filter framework for better results. Moreover, we also interested in the unified methodology of how to determine a decision threshold for different topics.

5. ACKNOWLEDGMENTS

The work reported in this paper was supported by the National Natural science Foundation of China Grant 60875033.

6. REFERENCES

[1] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[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] 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.

[4] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *ECML*, pages 137–142, 1998.

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

[6] Thorsten Joachims. Training linear svms in linear time. In *KDD*, pages 217–226, 2006.

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

[8] Jimmy Lin, Rion Snow, and William Morgan. Smoothing techniques for adaptive online language models: topic tracking in tweet streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 422–429, New York, NY, USA, 2011. ACM.

[9] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, New York, 2008.

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

[11] Stephen Robertson and Ian Soboroff. The TREC 2002

filtering track report. In *TEXT RETRIEVAL CONFERENCE*, 2002.

[12] J. J. Rocchio. Relevance feedback in information retrieval. 1971.

[13] D. Sculley and Gabriel Wachman. Relaxed online svms for spam filtering. In *SIGIR*, pages 415–422, 2007.

[14] 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.

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