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.

2.1 System Overview

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

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
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.
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-indepenent, the ranking formula can be rewritten as follows:

$$P(Q|D,t) \propto P(t|Q,D) \cdot P(Q|D)$$

$$\propto P(t|Q,D) \cdot Score(Q,D)$$

$$= P(t|Q,D) \cdot \sum_{w \in V} P(w|\theta_Q) \cdot \log P(w|\theta_D) \quad (2)$$

With the ranking formula, the retrieval task is reduced to three subtasks, i.e. the estimation of query model $\theta_Q$, the estimation of document model $\theta_D$ 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:

$$P(w|\theta_Q) = (1 - \alpha) \cdot P(w|\theta_Q) + \alpha \cdot P(w|\theta_{PRF}) \quad (3)$$

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:

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

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 1: System Architecture**

**Table 2: Summary statistics of TopicInfo corpus**

<table>
<thead>
<tr>
<th>Html Code</th>
<th>Status</th>
<th>Tweets in 2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>1,225,947</td>
</tr>
<tr>
<td>302</td>
<td>Found</td>
<td>688</td>
</tr>
<tr>
<td>403</td>
<td>Not Found</td>
<td>5,050</td>
</tr>
<tr>
<td>404</td>
<td>Forbidden</td>
<td>92,378</td>
</tr>
<tr>
<td>Null</td>
<td>Null</td>
<td>265,468</td>
</tr>
<tr>
<td>Searchable</td>
<td></td>
<td>1,226,635</td>
</tr>
</tbody>
</table>

(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
Learning to rank framework is MAP. 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>
<thead>
<tr>
<th>Run ID</th>
<th>P@30</th>
<th>MAP</th>
<th>R-Prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKUICST1</td>
<td>0.2164</td>
<td>0.1699</td>
<td>0.2176</td>
</tr>
<tr>
<td>PKUICST2</td>
<td>0.2068</td>
<td>0.1561</td>
<td>0.2120</td>
</tr>
<tr>
<td>PKUICST3</td>
<td>0.2113</td>
<td>0.1868</td>
<td>0.2174</td>
</tr>
<tr>
<td>PKUICST4</td>
<td>0.2333</td>
<td>0.2263</td>
<td>0.2174</td>
</tr>
</tbody>
</table>

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 high-rel 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

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Candidate Set</th>
<th>Feature Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrgBase</td>
<td>Origin Corpus</td>
<td>OrgTweetScore, Tweet Related Features</td>
</tr>
<tr>
<td>OrgTime</td>
<td>Origin Corpus</td>
<td>OrgTweetScore, Tweet Related Features, Temporal Feature</td>
</tr>
<tr>
<td>DEBase</td>
<td>DE Corpus</td>
<td>OrgTweetScore, Tweet Related Features</td>
</tr>
<tr>
<td>DEIssueTitle</td>
<td>DE Corpus</td>
<td>OrgTweetScore, OrgTitleScore, IssueTweetScore, IssueTitleScore, Tweet Related Features</td>
</tr>
</tbody>
</table>

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 SVMrank [6], we re-rank the candidate tweets and generate the final results.

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 evaluation 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

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 evaluation tool for TREC2012 is published, we’ll do more experiments for further conclusions.

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 evaluation tool for TREC2012 is published, we’ll do more experiments for further conclusions.

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.

\[ \vec{T}_i = (w_{1i}, w_{2i}, w_{3i}, \ldots w_{mi}) \]
\[ \vec{Q}_i = (w_{1q}, w_{2q}, w_{3q}, \ldots w_{nq}) \]

The T F IDF 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.

\[
\text{Sim} = \cos \theta = \frac{\vec{T}_i \cdot \vec{Q}_i}{||\vec{T}_i|| \cdot ||\vec{Q}_i||} \tag{8}
\]

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.

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

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

![Figure 4: Two-Stage Filtering Model](image-url)

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 hashtag. 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 evolution of topic. And we apply two relevance feedback models
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


<table>
<thead>
<tr>
<th>Run_ID</th>
<th>T11SU</th>
<th>F(beta=0.5)</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKUICSTF1</td>
<td>0.3424</td>
<td>0.2722</td>
<td>0.3963</td>
<td>0.2300</td>
</tr>
<tr>
<td>PKUICSTF2</td>
<td>0.3244</td>
<td>0.2525</td>
<td>0.3701</td>
<td>0.2809</td>
</tr>
<tr>
<td>PKUICSTF3</td>
<td>0.3233</td>
<td>0.2556</td>
<td>0.3857</td>
<td>0.2272</td>
</tr>
<tr>
<td>PKUICSTF4</td>
<td>0.3341</td>
<td>0.2629</td>
<td>0.3766</td>
<td>0.2936</td>
</tr>
</tbody>
</table>

Table 7: Performance of submitted runs


