Microblog Search and Filtering with Time Sensitive Feedback and Thresholding based on BM25

Wei Gao
Qatar Computing Research Institute
Qatar Foundation
Doha, Qatar
wgao@qf.org.qa

Zhongyu Wei
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
zywei@se.cuhk.edu.hk

Kam-Fai Wong
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
kfwong@se.cuhk.edu.hk

ABSTRACT

Microblogs such as Twitter are considered faster first-hand sources of information with many real-time fashions. We report our work in the real-time adhoc search and filtering tasks of TREC 2012 microblog track. Our system is built based on the traditional BM25 relevance model, in which specific techniques are tried out to respond to the need of finding relevant tweets. In the real-time adhoc task, we applied a peak detection algorithm for the process of blind feedback. We also tried to automatically combine the search results of multiple retrieval techniques. In the real-time filtering pilot task, we examine the effectiveness of some typical filtering methods previously used in TREC filtering track.

1. INTRODUCTION

This year comes the second edition of the TREC Microblog track following the evaluation for real-time tasks on Tweets2011 corpus. Like the real-time adhoc task in 2011's edition, the system is required to return the most recent relevant tweets for 60 newly created topics, posted earlier than the time each query was issued. In addition, Microblog track 2012 introduced a real-time filtering pilot task for the first time thought of as orthogonal to the real-time adhoc task. In this task, the goal is to select relevant tweets that are subsequently posted after a query issued at a particular time. This caters for the need of a user to monitor a developing topic on Twitter.

We participated in both of the tasks and submitted 4 runs for the adhoc and 3 runs for the filtering, all of which used the traditional BM25 [5] as the core relevance model. The reason to use BM25 is that its score function performs more effectively for IR in general. By this participation, we are hoping to create some baseline methods and then try to improve upon them.

- Real-time adhoc task: The 4 runs we submitted for this task are BM25, BM25PRF, BM25TRF, and MergedRun, which correspond to the baseline using BM25 scoring function, BM25 plus blind (pseudo-relevance) feedback, BM25 plus blind feedback favoring the tweets fell into the temporally detected peaks, and the weighted merge of result lists from the previous 3 runs, respectively.

- Real-time filtering task: We chose a typical adaptive filtering algorithm YFilter [8] as the baseline, which used BM25 with Rocchio feedback [6] as the basic scoring function. The reason to choose it is it's a greedy and purely real-time algorithm without the need of any offline training. Then its three variants are used to Tweets2011 corpus for experiments, resulting in QFilRun1, QFilRun2 and QFilRun3 for submission.

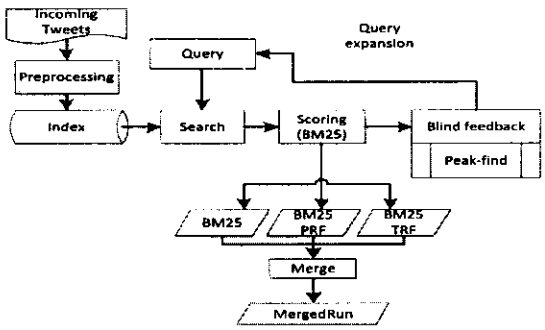

Figure 1: The work flow of Twitter search system for real-time adhoc task. We output 4 runs: BM25, BM25PRF, BM25TRF and MergedRun.

2. SYSTEMS OVERVIEW

2.1 Real-time adhoc search system

The architecture of search system is described in Figure 1.

The incoming tweets are firstly preprocessed with some basic normalization steps before indexing. All the English tweets are automatically identified using a language detection toolkit. All words were converted to lowercase and stemmed using Krovetz stemmer, the punctuations were removed, the words in the hastag '#' were duplicated, but we didn't removed the stop words.

To simulate the requirement of not using future information, we indexed only the tweets posted before the querytweet time for each query for avoiding the influence from future tweets on IDF.

We tried two types of blind feedback for query expansion. One is the traditional pseudo-relevance feedback (PRF) which assumes the top-n tweets in the initial search result as relevant, and the other is the temporal relevance feedback (TRF) which assumed the tweets falling into the largest peak detected from the initial results as rel-

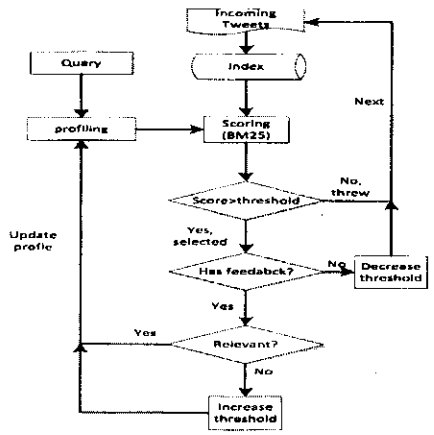

Figure 2: The work flow of Twitter filtering system for real-time filtering task.

evant. We applied the peak-finding algorithm [2] to automatically discover the main peaks from the results.

Based on the results of BM25, PRF and TRF, we merged the three result lists into a single list for the MergedRun using a weighted score combination method [7]. We utilized the queries and relevance judgement of Microblog track 2011 to train the weights of the lists for maximizing the MAP measure as did in [7].

The detailed method of peak-find-based blind feedback is presented in Section 4.1 and the result combination is given in Section 4.2.

2.2 Real-time filtering system

The workflow of the filtering system is described in Figure 2.

The procedure was an variant of YFilter [8]. The algorithm did not need any prior training data. Instead, it continuously updated the threshold of relevancy and the profile of queries with the incoming tweet streams. Initially, each query is profiled only based on the query words, and then the profile is updated automatically using Rocchio feedback [6] according to the decision of selection and the relevancy of the current tweet. The scoring function is based on BM25.

We output 3 runs. QFil1 used the tweets in the corpus that are out of the time range between the lowest querytweettime and the highest querynewesttweet of *all* queries for calculating prior IDF values, considered as using limited future information. QFil2 used only tweets that are up to the current incoming tweet for computing prior IDF values. QFil3 is based on QFil2 and placed more strict threshold update rules.

The details of the algorithm are described in Section 4.3.

3. DATASET AND STATISTICS

The Tweet2011 corpus contains tweets of two weeks from Jan. 23rd to Feb. 8th, 2011. According to the policy of Twitter, tweets corpus is not allowed to redistribute. Therefore, TREC Microblog dataset only contains the ids of over 16 million tweets, and the participants are required to crawl tweets content separately. However, the availability of past tweets is uncertain because twitter authors can delete their posts. Besides, the tweets downloaded by different groups may not be identical due to different quality of network connection. The statistics of our obtained dataset are shown in Table 1 and Table 2.

HTTP Response Code	# of tweets downloaded
200 (correct)	14,451,657
301	763
302	1,146,533
403	146,696
404 (connection error)	396,163

Table 1: Statistics of tweets download in the dataset.

	# with freq.	Unique #	# of Tweets
Mention	3,845,290	1,933,002	2,968,353
Hashtag	1,483,349	356,172	1,073,085
URL	1,560,068	-	1,537,608
Retweet (RT)	-	-	304,391
Total	80,009,364	385,742	6,246,970

Table 2: Statistics of downloaded English tweets in the dataset.

As shown in Table 1, among total 16,141,812 HTTP requests, we successfully downloaded 15,598,190 valid tweets. Since the task focuses on English tweets only, we eliminated all the non-English tweets first using the language identifier tool provided by Nutch[1], resulting in 6,246,970 English tweets in our corpus.

Table 2 indicates that in our English tweets, there are 3,845,290 mentions (with preceding symbol '@', which is used to directly refer to other users' tweets), 1,483,349 hash tags and 1,560,068 URLs. Totally, there are 80,009,364 words.

4. TECHNICAL DETAILS

4.1 Peak-Find-based blind feedback

Microblog streams are in the form of continuous incoming data flows, which is analogous to the network data transmission where care must be take for controlling conjestion. Peak-finding algorithm [2] was inspired by the similar problem encountered in TCP's conjection control mechanism. The goal is to determine whether a window (a fixed period of time) contains an usually large number of tweets in it. Unlike the pseudo-relevance feedback, we assume that only those tweets in the peaks of the initial search result list are relevant which can be used for guiding the blind feedback process. The rationale is that it is more likely those relevant terms to be found in the peaks of tweets for doing query expansion.

The peak-finding algorithm first groups the tweets into a histogram by time. We used hour as the basic unit and a whole day (24 hours) as a bin in our implementation. We counted the tweet-arrival rate in each bin starting from the querytweettime to the time of the oldest retrieved tweet. Then the detection process goes as follows (see [2] for details):

- When the algorithm encounters a significant increase in bin count relative to the historical mean, it starts a new window and follows the increase to its maximum.
- The algorithm ends the peak's window once the bin count returns to the same level it started at, or when it encounters another significant increase.

In Figure 3, we illustrate the correlation between the peaks of event and the relevant tweets according to the relevance judgement.

[1] https://issues.apache.org/jira/browse/NUTCH-623

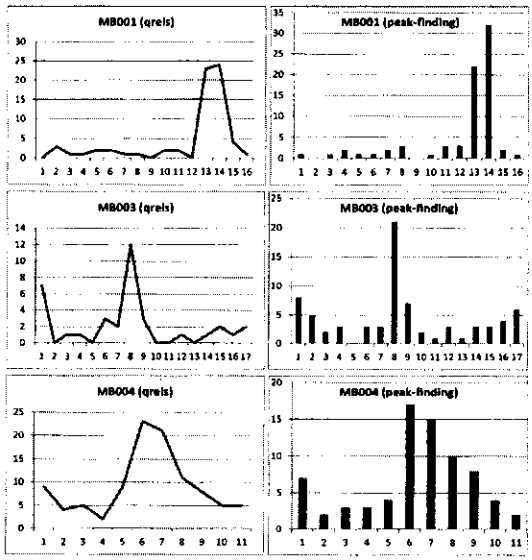

Figure 3: The close correlation of burst patterns between the retrieved tweets about the query and the relevant tweets from relevance judgement.

Figure 3 shows three real-time adhoc task topics in Microblog track 2011 when we used it to develop the system, where the curves at the left side are the count of relevant tweets (including highly relevant ones) in each day after the query is issued, and the histograms at the right side are the bursts of tweets of each day automatically identified by peak-finding from the initial search results. We can clearly observe such correlation between the bursty patterns of two sides.

After the initial retrieval, we applied peak-finding to the top-1000 search results and extracted those tweets falling into the first and second largest peaks, which are deemed as feedback documents used for query expansion. We output the results of the second-round retrieval using the expanded queries, referred to as BM25TRF.

4.2 Merge of different result lists

To improve the ranking of search result, many retrieval systems used score combination approach to merge the result lists produced by different retrieval methods. The basic assumption of improving ranking accuracy by combining ranked lists is that relevant documents are generally retrieved by multiple retrieval algorithms while different retrieval algorithms tend to retrieve different irrelevant documents [7]. Similar idea has been used in Metasearch, federate search and multilingual information retrieval (MLIR). Si and Callan [7] introduced a learning based merge algorithm for MLIR, where they used training data to learn the weights of each result list by directly optimizing the mean average precision (MAP) measure of the combined ranking result. We applied this approach to merge the results of our first three runs, i.e., BM25, BM25PRF, and BM25TRF.

Suppose there are N ranked lists to combine, we first normalized the retrieval scores of the each ranked list using Min-Max algorithm. Then the final combined scores for each tweet t is calculated as follows:

$$score(t) = \frac{1}{N}\sum_{i=1}^{N} w_i * score_i(t)^{r_i} \quad (1)$$

where $score(t)$ is the final combined score, $score_i(t)$ is the normalized score of t in the i-th ranked list, $\{w_i\}$ and $\{r_i\}$ are the model parameters.

The parameters $\{w_i\}$ and $\{r_i\}$ are estimated by maximizing MAP criterion which is interpolated with two regularization terms for the parameters to avoid overfitting:

$$(w_i, r_i)^* = \underset{(w_i,r_i)}{\text{argmax}} \left\{ \log \text{MAP} - \sum_{i=1}^{N} \left[\frac{(w_i-1)^2}{2a} - \frac{(r_i-1)^2}{2b} \right] \right\}$$

where $(w_i, r_i^*$ is the optimal model parameters and (a, b) are two regularization factors. In this work, we set $a = b = 3.0$ empirically, and the model parameters are estimated using the Powell's search method [3, 4].

We trained the model parameters using the 50 topics and the relevance judgement from the data of Microblog track 2011. Then the ranking scores of tweets in MergedRun on the 2012 topics was calculated using Eq. 1.

4.3 Greedy algorithm for online filtering

We implemented the profile updating algorithm of YFilter [8], which includes the process of indexing, profiling, relevance scoring, Ricchio's feedback and threshold updating as shown in Figure 2.

Unlike information retrieval, indexing for filtering is simpler in a sense that only certain statistics on document frequency (DF) are needed, and complex index structure can be ignored. We used hash map to store word DF. Note that the DF table should be updated once an incoming tweet is received.

The profiling process creates and maintains the information of a topic and its expansion, which is initialized with the original query words and is updated with the expansion words extracted from feedback tweets.

The scoring function for relevance is based on BM25 formula [5] with regard to profile words and tweet words, where we set the common BM25 parameters as $k1 = 1.2$, $k2 = 0$, $k3 = 8$ and $b = 0.75$. An incoming tweet is selected or filtered out depending on whether its relevance score is greater than the threshold.

Ricchio's feedback is used to update the profile. Whenever a relevant tweets is selected, all words in the tweets are added to the profile's candidate word list and then weighted using the incremental Rocchio formula (see [8] for details). We set the common Ricchio parameters as $\alpha = 1.0$, $\beta = 0.5$, and $\gamma = 0.25$.

The threshold is dynamically updated depending on the nature of a selected tweet. If the selected tweet is not relevant, we increase the threshold to make it more strict. If the selected tweet has no feedback at all, we decrease the threshold in a small scale to allow for a higher chance to see the tweets with feedback information (see [8] for details).

5. EXPERIMENTS AND RESULTS

5.1 Topics and Statistics

The test dataset of real-time adhoc task (qrels) was established by pooling all the runs from the participating groups. There are 60 new topics in total this year. Like last year, three levels of relevancy are defined–highly relevant, relevant and non-relevant. Retweets without further information were removed from the pool as they were assumed non-relevant.

The evaluation was done by considering only the highly relevant as relevant tweets. However, topic 53, 69 and 105 did not contain any highly relevant tweets in the pool, thus was discarded from the

	Full size corpus	Our missing
Total tweets number	73,073	6,416
Highly relevant tweets number	2,572	52
Relevant tweets number	6,286	107
Non-relevant tweets number	66,787	6,309

Table 3: Statistics of the test dataset (qrels) in real-time adhoc task with 60 new topics.

	Full size corpus	Our missing
Total tweets number	40,855	4,302
Highly relevant tweets number	558	13
Relevant tweets number	2,864	73
Non-relevant tweets number	37,991	4,229

Table 4: Statistics of the test dataset (qrels) in real-time filtering task with 50 topics.

evaluation resulting in 57 topics. We retrieve 10,000 tweets per query.

The statistics of the adhoc task test dataset are shown in Table 3, where the missing column reports the number of tweets missed in our corpus due to download failure.

The test dataset in real-time filtering task consists of the 50 topics in the 2011 adhoc task and the same relevance judgement. The statistics of the filtering task test dataset are shown in Table 4.

5.2 Results

5.2.1 Result of real-time adhoc task

BM25 is the baseline run. The results of all 4 runs are shown in Table 5 in terms of Precison@30, MAP and R-Prec. The ROC curves [1] of the 4 runs are given in Figure 4.

As we can observe, the performance of all our best runs (the bolded) is higher than the median. In terms of MAP and ROC curve, MergedRun outperformed other runs. T-test showed that its better performance over BM25 and BM25TRF is statistically significant (95% confidence level). But according to P@30 and R-Prec, BM25PRF performed the best. However, it's not significantly better than others except for BM25. This may suggest that the MergedRun has obvious advantage than other approaches.

Although BM25PRF outperformed BM25TRF in terms of the three measures, we didn't find it statistically significant. This indicates that the relevance feedback based on the bursty event detection using peak-finding algorithm can achieve comparable effectiveness as the commonly used pseudo-relevance feedback. Also as we can see, BM25TRF returned the most number of relevant tweets.

5.2.2 Result of real-time filtering task

We submitted 3 similar runs in this task, in which QFil1 used limited amount of future information, QFil2 didn't use any future tweets, and QFil3 imposed some extra constraints on threshold updating to make it more strict than the original rule of YFilter. The results of all runs are shown in Table 6.

We found our results are generally poor in terms of all measures except for the excellent performance on the recall. This is actually caused by the large number of retrieved results we returned due to the conservative threshold setting, which let pass most of the relevant tweets but also much more irrelevant ones. For example, both QFil1 and QFil2 returned over 200K tweets. QFil3 returned

Run	Rel_Ret (2,572)	P@30	MAP	R-Prec
BM25	1,652	0.150	0.133	0.152
BM25PRF	1,875	**0.182**	0.154	**0.188**
BM25TRF	**1,907**	0.171	0.150	0.183
MergedRun	1,902	0.178	**0.157**	0.176
Best (est.)	-	0.392	0.414	0.430
Median (est.)	-	0.181	0.149	0.187

Table 5: The adhoc task performance of our submitted runs compared to the results of the estimated best and median. The bold numbers indicate our best runs.

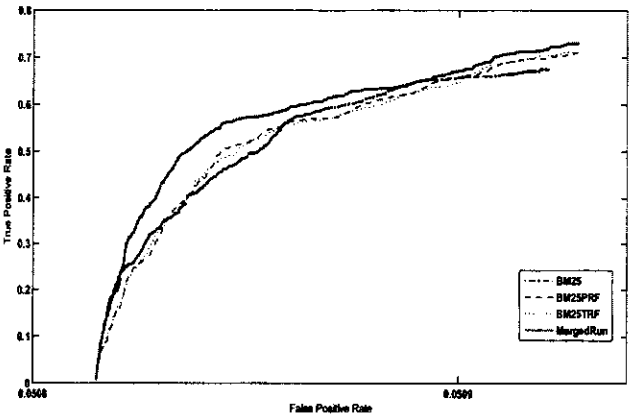

Figure 4: The ROC curves of the 4 adhoc runs. The curve closest to the upper left corner is deemed the best performance.

over 60K due to its relatively strict threshold setting strategy, and this obviously improved the result a lot.

We did an additional experiment by simply aggressively increasing the initial threshold value by 60% after the official result was released. The numbers in the row of "Post" in Table 6 indicate some significant improvement on the three measures over our best result with wide margin. There also could be other factors adjustable for further improvement, such as the magnitude of increase or decrease of the threshold when feedback is obtained. This suggests a large room for us to improve the thresholding in the future.

6. CONCLUSIONS

We describe our microblog search and filtering systems for participating in microblog track 2011 which were built based on the traditional BM25 model. For the real-time adhoc task, we applied a peak detection algorithm for the process of blind feedback. We also tried to automatically combine the search results of multiple retrieval techniques. Results suggested that the result merging per-

Run	Ret	Rel_Ret	T11SU	F_0.5	Prec	Recl
QFil1	202,283	1,620	0	0.037	0.030	0.726
QFil2	208,760	1,611	0	0.040	0.033	0.723
QFil3	66,416	1,421	0.013	0.072	0.062	0.610
Post	23,415	752	0.078	0.116	0.104	0.381
Best (est.)	-	-	0.596	0.607	0.922	0.946
Median (est.)	-	-	0.207	0.149	0.176	0.334

Table 6: The filtering performance of our submitted runs compared to the results of the estimated best and median.

formed the best, and the temporally detected bursts can be helpful to the relevance feedback, archiving comparable effectiveness as the commonly used pseudo-relevance feedback. In the real-time filtering pilot task, we examine the effectiveness of some typical filtering methods previously used in TREC filtering track. The method didn't work well due to our lenient threshold updating strategy. However, more aggressive update on threshold did demonstrate some improvement. We will continue to tune our systems.

7. REFERENCES

[1] C. D. Manning, P. Raghavan, and H. Schütze. Introduction to Information Retrieval. Combridge University Press, 2008.

[2] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller. TwitInfo: Aggregating and Visualizing Microblogs for Event Exploration. In Proceedings of CHI 2011.

[3] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. Computer Journal, 7(2): 155-162, 1964.

[4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical Recipes in C: The Art of Scientific Computing (10.5) Cambridge University Press, Cambridge, 1992.

[5] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In Prceedings of the Third Text REtreival Conference, 1994.

[6] G. Salton. The SMART retrieval system: experiments in automatic document processing. pages 313-323. Prentice-Hall Inc., 1971.

[7] L. Si and J. Callan. CLEF 2005: Multilingual Retrieval by Combining Multiple Multilingual Ranked Lists. In Proceedings of CLEF 2005.

[8] Y. Zhang and J. Callan. YFilter at TREC-9. In Prceedings of the Ninth Text REtreival Conference, 2000.