Exploring Tweets Normalization and Query Time Sensitivity for Twitter Search

Zhongyu Wei[1], Wei Gao[2], Lanjun Zhou[1], Binyang Li[1], and Kam-Fai Wong[13]

[1]Department of System Engineering & Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China

{zywei, ljzhou, byli, kfwong}@se.cuhk.edu.hk

[2]Qatar Computing Research Institute, Qatar Foundation, Doha, Qatar

wgao@qf.org.qa

[3]Key Laboratory of High Confidence Software Technologies, Ministry of Education, China

Abstract

This paper presents our work for the Realtime Adhoc Task of TREC 2011 Microblog Track. Microblog texts like tweets are generally characterized by the inclusion of a large proportion of irregular expressions, such as ill-formed words, which can lead to significant mismatch between query terms and tweets. In addition, Twitter queries are distinguished from Web queries with many unique characteristics, one of which reflects the clearly distinct temporal aspects of Twitter search behavior. In this study, we deal with the first problem by normalizing tweet texts and the second by capturing the temporal characteristics of a topic. We divided topics into two categories: time-sensitive and time-insensitive. For the time-sensitive ones, we introduce a decay factor to adjust the relevance score of results according to the expected date of the topical event to happen, and then re-rank the search results. Experiments demonstrate that our methods are significantly better than baseline and outperform the medium of all runs.

1. Introduction

The 2011 edition of the TREC Microblog track is the commencement of evaluation for Twitter search task, where it introduced Realtime Adhoc Task for the first time. In this task, the system is required to return the most recent relevant tweets for 50 specific queries. We, from the Laboratory of Web Intelligence Systems Engineering at the Chinese University of Hong Kong (WISE@CUHK), participated in the evaluation task and submitted the results on four different runs, namely, Wise2ndRun, WiseThirdRun[1], WiseFouthRun and WiseFifthRun.

Our approach stems from two different angles: one aims at ill-formed content processing, and the other is focused on time sensitivity of Twitter queries, both corresponding to the distinctive genres of microblog context.

- **Content processing.** Unlike traditional news content, tweets contain a large number of irregular expressions, including typos, phonetic substitutions, adhoc morphological terms, emoticons, and specialized syntax, and so on. These ill-formed elements can lead to significant mismatches between query terms and microblog content, thus worsening the search effectiveness. Therefore, the most straightforward idea is to automatically regularize all ill-formed tweets content and recover them to the original forms. In our work, we proposed to alleviate the mismatch problem following this perspective.
- **Query processing.** Twitter queries are typically very short in length [4]. In order to obtain more relevant search results, pseudo relevance feedback (PRF) is used to expand the original queries. More importantly, recent studies showed that people often search Twitter to find temporally relevant information [4], such as current events, trending topics and real-time information. Therefore, we divided Twitter queries into two categories, namely time sensitive and time insensitive queries. The former typically reflects the information

[1] WiseThirdRun was found wrong afterwards and the results of the run is invalid (see Section 2&6)

need on emerging events or relevant tweets burst in a specific period of time (e.g., one day or even an hour). For the latter case, we generally cannot observe from them clear time-specific information need and there is not temporal characteristic in the result tweets list either. For time sensitive queries, we re-rank the retrieved tweets by introducing a temporal decay factor (see Section 5).

This paper is organized as follows. Section 2 describes our framework in general; Section 3 introduces the data and preprocessing; Section 4 presents the details of tweets content normalization; Section 5 gives the details of capturing temporally related search results and the tweets re-ranking method; Section 6 discusses experimental results; finally, we draw some conclusions and point out the future work in Section 7.

2. Our framework

Our approach is proposed to examine the effectiveness of the two perspectives with tweets content processing and query processing (see Section 1) and their combination. We employed Indri toolkit[2] for indexing and retrieval [3]. The framework of our approach is shown in Figure 1.

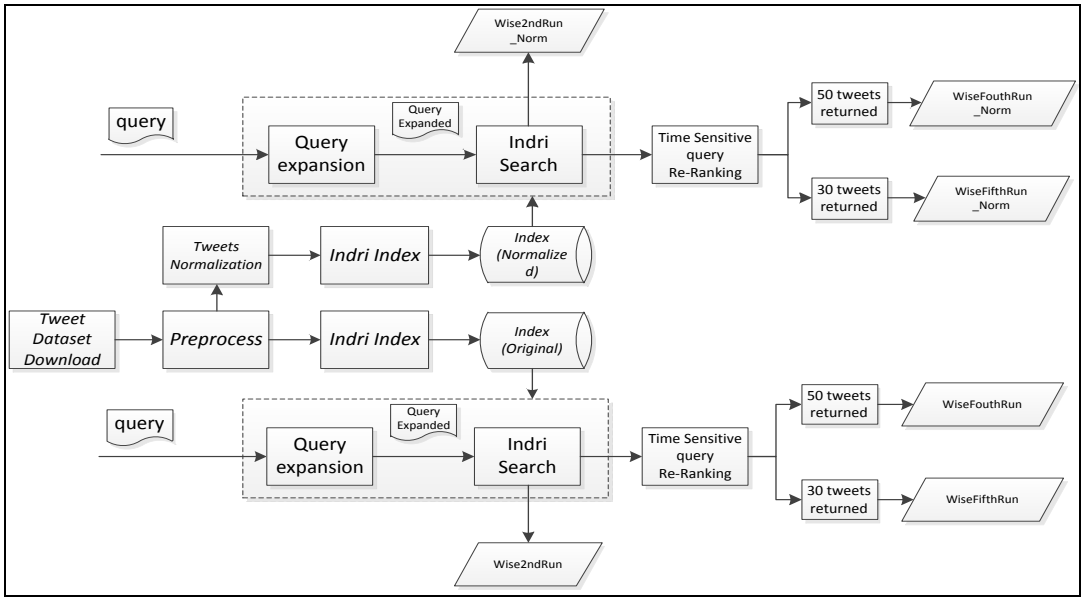

Figure 1. The framework of our approach for the Realtime Adhoc Task (Wise2ndRun: PRF w/o tweets normalization; WiseFouthRun: 50 search results with decay factor; WiseFifthRun: 30 search results with decay factor; the other three runs with "_Norm" tag are their counterparts based on normalized tweets)

We performed three runs for searching normalized tweets, namely Wise2ndRun_Norm, WiseFouthRun_Norm and WiseFifthRun_Norm. Wise2ndRun_Norm is the basic run using PRF without the particular consideration of time-sensitive queries. The other two runs both dealt with such kind of queries by time-sensitive query detection and tweets re-ranking. Based on the initial ranking results returned by Indri, we distinguish queries with temporal characteristics from others and then re-rank their results by introducing a decay factor to adjust the relevance of each tweet. The difference between WiseFouthRun_Norm and WiseFifthRun_Norm is that they returned different number of tweets in the result lists. This is because all the submitted results are subject to a final re-ranking purely by timestamps by the organizer. Therefore, different number of returned tweets will result in different performance.

For the raw tweets not normalized, we conducted the corresponding three runs named Wise2ndRun, WiseFouthRun and WiseFifthRun for comparisons. Unfortunately, by the time of result submission, we didn't

[2] http://lemurproject.org/indri/

complete all three normalized runs. Therefore, we only submitted one normalized run (named as WiseThirdRun) plus the three non-normalized runs. What was even more unfortunate is that WiseThirdRun was found a serious bug afterwards, and thus the original submitted result of normalized run (i.e., WiseThirdRun) is invalid. In this report, we rectified the problem and present additionally all the results of the normalized runs.

3. Dataset and Preprocessing

The microblog track 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 the dataset obtained by our group are shown in Table 1 and Table 2.

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[3], 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 (with preceding symbol '#', which is used to describe a topic many tweets may relate to), and 1,560,068 URLs. Additionally, there are 80,009,364 words and more than 14.4% of them (11,575,609) are ill-formed words which cannot be found in a common English dictionary (see Section 5). These results are obtained if we count the frequency. If frequency is not counted, we ended up with 385,742 unique terms in the corpus, and 329,562 of them are ill-formed.

Stemmer is used in preprocessing. Moreover, the tweets are preprocessed by replacing some commonly occurred components in tweets to the corresponding uniform symbols. The transformation rules are as follows: @username → ATMENTION, #topic → HASHTAGTOPIC, URLs → HTTPURL, punctuations → PUN, numbers → NUM. All the introduced symbols are considered as stop words during indexing. However, they should be kept during normalization as contextual information for OOV words (see Section 4).

Table 1. Statistics of tweets download in the dataset

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 2. Statistics of downloaded English tweets in the dataset

	# with frequency	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
Ill-formed word	11,575,609	329,562	1,311,448
Retweet (RT)	-	-	304,391
Total	80,009,364	385,742	6,246,970

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

4. Tweets Normalization

Normalization of tweets is a challenging problem due to the complexity of irregular formations of ill-formed words, rendering their detection and correction difficult. We implemented the recent lexical normalization technique for short text messages proposed by [1], which includes four steps: (1) Out-of-Vocabulary (OOV) word detection, where we detect OOV word in tweets based on a common English dictionary. (2) Slang words detection, where slang words are detected and translated into the corresponding standard form based on a slang word dictionary. (3) Candidate set generation, where we generate the candidates of standard forms for each OOV word. (4) Candidate selection, where we choose the standard word from the candidate set. For simplicity, we didn't adopt the classifier to further differentiate true ill-formed words from OOV words as [1] did, because we are lack of a large set of tweets for model training. Below describes our implementation of the method. More details of the original method can be found in [1].

4.1 OOV word detection

Similar as [1], we searched each token in tweets text in the common English dictionary, i.e., GNU aspell dictionary v0.60.6, which contains 124,589 words in total, and treated words not found as OOV words. All the one character tokens except 'a' and 'I' are removed. We didn't treat "RT" as in vocabulary (IV) word as [1] did, and we simply removed all 'RT' from tweets for efficiency. Also, "RT" always occurs at the beginning of retweet contents and is not useful as context of OOV words.

4.2 Slang word translation

Slang words are usually OOV of a common English dictionary, but could be covered by a slang word dictionary, for which we converted them into standard form using the Internet slang word dictionary[4] suggested in [1], which contains 5,255 entries in total.

4.3 Candidate set generation

For each OOV word, we generated a candidate set of standard words based on lexical edit distance and phonemic edit distance between the OOV word and the candidate. The process basically followed [1]. Firstly, any repetition of more than 3 letters within an OOV word is reduced to 2 letters (e.g., coooooooool is reduced to cool). Note that this is different from [1] where they reduced 3+ repetitive letters to 3. This considered the fact that there are rarely standard words containing 2+ repetitions of the same letter. Secondly, IV words within a threshold T_c character edit distance of the given OOV word were treated as candidates. Thirdly, IV words within a threshold T_p phonemic edit distance of the given OOV word were treated as candidates. Following [1], T_c was set as 2 and T_p was set as 1.

4.4 Candidate selection

Here we identify the most likely IV word from the candidate set as the standard form of an OOV word. Two kinds of features are used to rank the candidates including word similarity and context inference. For capturing word similarity, we incorporated lexical edit distance, phonemic edit distance, prefix substring, suffix substring and longest common subsequence (LCS) to measure the overall similarity. Besides, language modeling and dependency-based frequency were utilized as context inference features. It is straightforward to calculate the word similarity features. However, context inference features need more elaborations.

- **Dependency-based frequency feature:** We replaced the target OOV word in tweets with the candidate standard form iv_i and computed the dependency frequency based on a dependency bank generated from the New York Times (NYT)[5] for iv_i [1]. The dependency similarity between iv_i and oov is computed by

[4] www.noslang.com
[5] 48 million sentences from English Gigaword (www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2008T19)

$$Sim_{dependency}(oov, iv_i) = \frac{dependency(iv_i) + \mu}{\sum_i dependency(iv_i) + \mu * k}$$

where μ is a smoothing factor, which is empirically set to 1, and k is the number of candidates in the candidate set. Following [1], the dependency count of iv_i was the sum of dependency frequency of each word pair that contains iv_i and a contextual word in tweet context. Three words before and after the target OOV word were taken into consideration as the contextual words. There were two steps to extract dependency relation: (1) The Stanford parser [2] was used to extract dependencies from NYT corpus; (2) The dependency feature of each word pair was calculated by combining the counts of its corresponding dependency forms to obtain a confidence score.

- **Language model**: The context inference features based on language model was simplified based on [1]. We replaced the target OOV word in tweets with the candidate standard form iv_i and computed similarity between oov and iv_i based on the counts of tri-grams collected from the Web 1T 5-gram Corpus[6]. In our implementation, WebEasy[7] toolkit was used to process the corpus. The similarity between oov and iv_i based on a single tri-gram containing oov was computed by

$$Sim_{lm}(oov, iv_i) = \frac{\sum_j count(Tri_j(iv_i))}{\sum_i \sum_j count(Tri_j(iv_i))}$$

where $Tri_j(iv_i)$ is the *j*-th tri-gram containing iv_i given the set of tri-grams in the corpus. Since there may be multiple tri-grams containing oov in a single tweet, we sum up the similarity values above based on all its corresponding tri-grams and then normalize the similarity.

The language model computation was time consuming where a single inquiry needs several seconds. So, we divided the selection process into two steps based on different features. We first ranked all the candidates by the combination of six features listed above except for language mode feature, and then we computed the language model for top-*m* candidates and chose the candidate with the highest language model score as the standard form of the target OOV. In our implementation, *m* was set as 10. The six features were combined linearly with equal weights following [1].

5. Time Sensitive Query Detection

By observing the topics in test corpus, we found that there are two kinds of queries, namely time-sensitive and time insensitive topics. Time sensitive topics are almost always related to new events and the relevant tweets may burst out in some specific period of time. For this kind of topics, we should assign higher weight to the relevant tweets published in the target date (tweets bursting date). For example, "Jintao visit US" presents the event that Chinese President Hu Jintao visited USA during Jan. 18th and Jan. 20th. In contrast, time insensitive topics reveal some general information need without particular temporal concerns.

We explored time distribution of the search results for each topic to determine if a given topic is time sensitive or not as well as the bursting time of the event. A simple approach was proposed. If there are more than half of top *n* retrieved tweets posted in a single day, the query was determined as time sensitive, otherwise not. Also, the date with the highest frequency appearing in the top *n* tweets was chosen as the target date, where *n* was set to 3 empirically. For the identified time sensitive topics, we adjusted the relevance scores of the retrieved tweets by adding a temporal decay factor shown as follows:

$$Relevance(Tweet_i) = Relevance(Tweet_i) * e^{(-k)*abs(T_i - T_{core})}$$

where $Tweet_i$ stands for the *i*-th tweet returned, k is an adjustment factor, T_i is the time of the tweet being

[6] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13
[7] http://sourceforge.net/projects/webeasy/

published and T_{core} is the predicted target time of the event. We empirically set k as 0.5. Our method is liable to push the relevant tweets with the target date to the top of the result list.

6. Experiments and Results

Our group submitted four runs, namely the Wise2ndRun, WiseThirdRun, WiseFouthRun and WiseFifthRun. However, WiseThirdRun, which was the only normalized run completed by the time of submission, was found invalid afterwards due to a major bug. For this report, we rectified the problem and finished the three runs on normalized tweets corpus, namely the Wise2ndRun_Norm, WiseFouthRun_Norm and WiseFifthRun_Norm. Comparative experiments were conducted to study the effectiveness of the normalization, the temporal decay factor and the number of tweets returned.

6.1 Test Dataset and Setup

The test dataset of microblog track 2011 was established by pooling all the 184 runs from 58 participating groups. The top 30 retrieved tweets from all the submissions were pooled according to the retrieval scores provided in each run. Retweets without further information were removed from the pool as they were assumed non-relevant. There are 50 topics in total this year, however, topic 50 did not contain any relevant tweets in the pool, thus was discarded from the evaluation. Eventually, only 33 topics contain tweets judged to be highly relevant. The evaluation results would present two separate sets of scores. The first set of scores result from considering all relevant and highly relevant tweets as relevant, in which there are 49 topics in total. The second set of scores should be obtained by considering only highly relevant tweets as relevant, and there are 33 topics in this case. We present results based on the former set of scores.

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

Table 3. Statistics of the test dataset

	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

5.2 Results and Discussions

Four sets of experiments were carried out for comparison: (1) we examined the effectiveness of normalization; (2) we investigated the influence of temporal decay factor to time sensitive queries; (3) we studied the impact of the number of returned tweets; (4) finally we studied the performance on each of the 50 queries in the our best run. The performance of baseline is provided by the taskforce based on Lucene[8] search engine. The results are shown in Table 4. Note that we also provided the performance based on Precision@5,10 and 20 (i.e., P@5, P@10 and P@20) in addition to the three standard measurements, i.e. MAP, R-Prec and P@30 required by the Task. This is because we believe that the precision achieved at top few positions in the result list is more important than the overall accuracy since Twitter searchers tend less likely to read a number of returned tweets.

Overall, our results are obviously better than the medium level of performance among all submissions, but obviously worse than the best results. It seems to us that using the very simple techniques proposed here may

[8] http://lucene.apache.org/java/docs/index.html

result in fairly decent effectiveness. After all, there are lots of rooms for us to make further improvements based on these techniques.

Table 4. Performance of our submitted runs and additional runs compared to the results of best run, medium and baseline. The numbers in bold are the best performance of our runs

Run	Ret_Number	MAP	R-Prec	P@5	P@10	P@20	P@30
Wise2ndRun	50	0.2060	**0.2660**	0.3755	0.3571	0.3582	0.3429
WiseFouthRun	50	0.1949	0.2541	0.4082	0.3857	0.3622	0.3265
WiseFifthRun	30	0.1710	0.2191	0.4449	0.3939	0.3582	0.3578
Wise2ndRun_Norm	50	**0.2080**	0.2659	0.4163	0.3796	**0.3663**	0.3469
WiseFouthRun_Norm	50	0.1909	0.2527	0.4204	0.3959	0.3612	0.3265
WiseFifthRun_Norm	30	0.1734	0.2220	**0.4612**	**0.4061**	0.3653	**0.3612**
Best	-	0.5126	0.6149	**-**	**-**	**-**	0.6115
Medium	-	0.1433	0.1944	**-**	**-**	**-**	0.2591
Baseline	-	0.1411	0.1486	0.2082	0.1612	0.1143	0.0986

- **Normalization**

As shown in Table 4, the three additional runs on normalized tweets outperform their corresponding original runs without normalization. This indicates that our normalization, although not so comprehensive as its full version [1], is effective as we had expected. For example, in Topic 5 ("NIST computer security"), tweets containing "computerrr", "komputer", etc. were normalized to "computer" and were successfully retrieved; in Topic 15 ("Thorpe return 2012 Olympics"), tweets containing "thorp", "thorpedo" and "thorpey" were normalized to "Thorpe". By retrieving tweets with relevant tweets containing ill-formed terms, the performance was improved. However, normalization also led to many noises. For instance, "Socceroos" was treated as ill-formed in the tweet "`Socceroos Win! Now you can win - FIFA Soccers Balls @todayscatch - clickhere for more details: http://bit.ly/hjNfjm GO #socceroos #kewell`" and normalized as "soccer". The normalized version of this tweet was retrieved by Topic 2 ("2022 FIFA soccer"), but the non-normalized version was not. Since the tweet itself is not relevant to the 2022 FIFA game, normalization seems detrimental here.

We further examined the performance of normalization. Actually, it's impossible to estimate how many query words were ill-formed in tweets. The recall of the ill-formed words detection therefore cannot be obtained. However, we can estimate how many identified ill-formed tweet words correspond to our query words. There were 2,189 ill-form tweet words which were normalized into our query terms. Among 500 ill-formed words chosen randomly from tweets, we found that only 104 were correctly normalized (around 20% accuracy). Among the rest of 396 words, 198 of them were simply OOV words in correct form, such as "iphone", "Caetano", etc. not in the common dictionary, and 46 of them were abbreviations such as "what2cook" (what to cook), which were not appropriately handled in our approach.

In addition, normalization seems to be more useful when we concentrate on the accuracy in the top few positions in result list. Normalized runs demonstrated clearly higher effectiveness than the non-normalized runs in terms of P@5 and P@10. However, normalization tends to be useless or even detrimental when we consider P@20 and P@30, which means a lot more noises could be brought in at the lower ranked positions since the ill-formed words may be turned into a query term incorrectly.

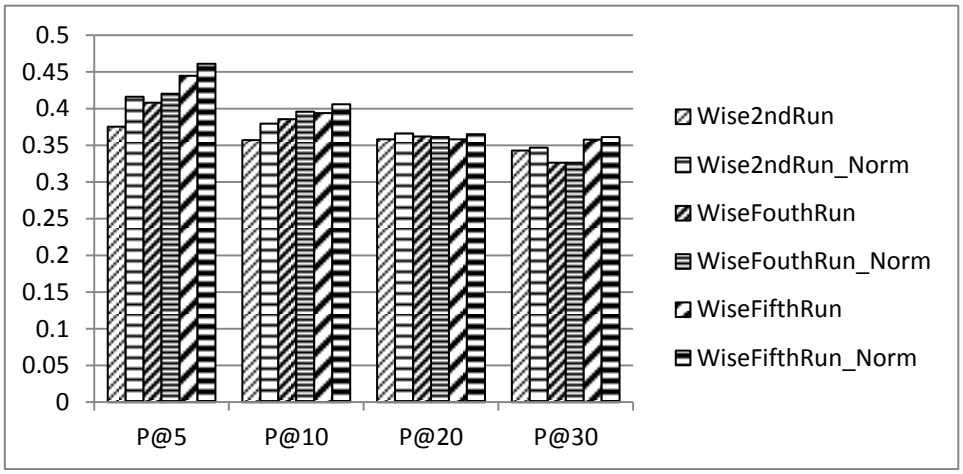

Figure 2. P@N of our six runs, which shows that our methods (by normalization and temporal considerations) are obviously more effective for ranking at top *n* positions in the result list

- **Temporal property of queries**

In our approach, temporal decay factor encourages to capture the relevant tweets (post time close to the occurrence of the event) instead of new tweets. We compared WiseFouthRun to Wise2ndRun to show the effectiveness of our method dealing with time-sensitive queries. As shown in Figure 2, WiseFouthRun clearly outperformed Wise2ndRun for P@5, P@10 and P@20. However, the P@30 of WiseFouthRun became 0.0164 lower than Wise2ndRun. Similar observation can be found in normalized runs.

Further analysis showed that there are two different event burst patterns in tweets characterized as single burst point and multiple burst points. The latter typically represents the pattern of an event that bursts in different non-consecutive times. For example, Topic 9 ("Toyota Recall") contains two bursts at Jan. 26th and Feb. 8th. Our approach however can only catch the first burst point. Besides, we designated a single date as the target date. In the case of single burst point, some topics could keep lively for several consecutive days, such as topic 20 ("Taco Bell filling lawsuit") from Jan. 26th to Jan. 29th. Since we cannot capture tweets published in other bursting days which are not treated as target date, the performance drops when relevant tweets are distributed in different dates. And it will also bring more noise leading to lower recall in that case. The worse MAP and R-Prec values for WiseFouthRun (or WiseFouthRun_Norm) than Wise2ndRun (or Wise2ndRun_Norm) may be due to this reason. Therefore, significant rooms exist for improvements.

Furthermore, we argue that P@30 may not be a good metric in this Task. We found that the number of relevant tweets of 19 queries in the test set is less than 30, and 15 out of these 19 queries have no more than 20 relevant tweets.

- **Number of tweets returned**

Since the submitted results were subject to a final re-ranking purely based on time by the organizer, the different number of tweets returned will lead to different performance. Our preliminary experiments showed that P@30 could monotonously increase with the decrease of the number of returned tweets. Therefore, we set returned number to 30 and obtained the WiseFifthRun and WiseFifthRun_Norm. As the precisions shown in Figure 2, WiseFifthRun outperforms WiseFouthRun except for P@20, and WiseFifthRun_Norm outperforms WiseFouthRun_Norm for all tempted positions. However, the MAP and R-Prec values of Wise5thRun are both lower because the recall of relevant tweets is lowered due to the small number of returns.

- **Performance on individual queries**

We studied the effectiveness on individual queries based on the P@30 values of WiseFifthRun_Norm for

better understanding specific cases, which was shown as Figure 3. The P@30 values varied significantly over all the 49 topics. We found that the precisions are zero on Topics 15 ("Thorpe return in 2012 Olympics"), 16 ("release of 'Known and Unknown'"), 32 ("State of the Union and jobs") and 33 ("Dog Whisperer Cesar Millan's techniques"). Further examinations revealed that Topic 15 and 16 only have 2 relevant tweets according to relevance judgment, and for Topic 33, the relevant information containing query words is hidden in the webpages pointed by the outward links of tweets. Our method using PRF introduced lots of noises to the feedback process of the former three topics with only a few relevant results, and we also did not consider external webpage information, thus missing the relevant tweets for Topic 33.

In contrast, the precisions are as high as over 80% for Topics 7 ("Pakistan diplomat arrest murder"), 9 ("Toyota Recall"), 20 ("Taco Bell filling lawsuit") and 37 ("Giffords' recovery"). The reason is that these are easy queries each of which has over 80 relevant tweets according to relevance judgment.

Exceptions are Topics 8 ("phone hacking British politicians"), 17 ("White Stripes breakup") and 45 ("political campaigns and social media"). Their precisions are still very low even though there are many relevant tweets for these queries (i.e., 99, 64 and 85 relevant tweets, respectively). We found that for the former two topics, our normalization failed, bringing in many noisy tweets. And for the latter, the terms "social media" and "campaign" occurred in a large number of irrelevant tweets, which were inevitably retrieved out.

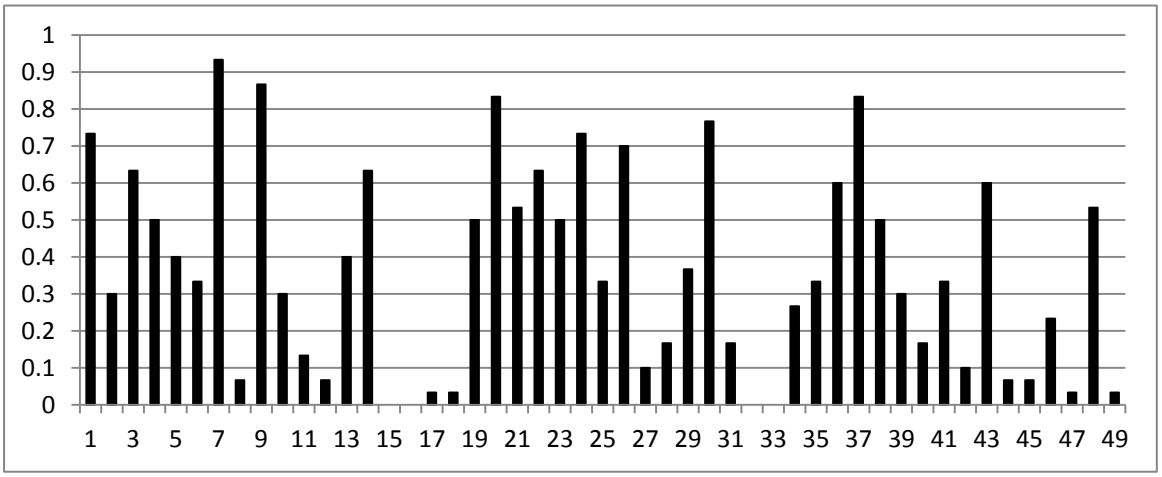

Figure 3. P@30 value of each topic based on our run of WiseFifthRun_Norm

7. Conclusion and Future Work

In this work, we explored two obvious characteristics of search in microblog contexts, i.e., the ill-formed tweet words and the time-sensitivity of queries. By incorporating tweets normalization and time-sensitive query detection, our approach to the Realtime Adhoc Task of TREC 2011 Microblog Track performed significantly better than the baseline and obviously outperformed the medium of all runs. Both of the two features considered are effective, but there still exists much room for further improvements.

For normalization, differentiating new words emerging in microblog and the truly ill-formed words should be studied more carefully. Multiple ill forms for a single word or a single ill form for multiple words is especially confusing and difficult to handle. For time-sensitive queries, more sophisticated method should be explored to deal with multi-bursting points or multiple target times. Besides, our model focused on utilizing temporal properties to obtain relevant tweets instead of new tweets, which is only partially compatible with the new-tweets-first strategy. In the future, we will give the recency more priority in ranking.

References

[1] Bo Han and Timothy Baldwin. Lexical Normalisation of Short Text Messages: Makn Sens a #twitter. In Proceedings of ACL 2011, pages 368-378.

[2] Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In Proceedings of NIPS 2002, pages 3-10.

[3] D. Metzler and W.B. Croft. Combining the Language Model and Inference Network Approaches to Retrieval. Special Issue on Bayesian Networks and Information Retrieval, Information Processing and Management, 40(5):735-750, 2004

[4] Jaime Teevan, Daniel Ramage, and Meredith Ringel Morris. #TwitterSearch: A Comparison of Microblog Search and Web Search. In Proceedings of WSDM 2011, pages 35-44.