Time-sensitive Weighting for Microblog Retrieval

Hao Wu and Hui Fang
University of Delaware, Newark DE 19716, USA
haowu@ece.udel.edu, hfang@ece.udel.edu

Abstract

We report our system and experiments for the realtime Adhoc task in the 2011 MicroBlog track. Our goal is to develop effective technique to retrieve relevant tweets that have been posted recently. In particular, we propose a time-sensitive term weighting strategy that can favor tweets in hot-discussed time and a document length related weighting method that can favor long tweets which are more likely to be interesting. Query expansion technique is also used to further improve the retrieval performance.

1 Introduction:

In this year's realtime Adhoc task, Tweets2011 corpus is used. It contains about 16 millions tweets over the period from Jan. 24th, 2011 to Feb. 8th, 2011. Every topic contains not only a query but also the time when the query is submitted. Retrieval Systems are required to return tweets that are posted before the query was submitted, and the tweets need to be sorted based on the post time.

The problem set up and the characteristics of tweets make it necessary to consider the following three components: relevance, interestingness and the recency. Intuitively, relevant tweets should satisfy the following three properties. (1) tweets containing more query terms are more likely to be relevant. (2) tweets containing query terms with higher IDF values are more likely to be relevant. (3) tweets appearing in hot-discussed time period are more likely to be relevant and interesting. The first two properties are similar to traditional TF-IDF method and they lead to our baseline directly. The third property is a unique feature of microblog search and it leads to our time-sensitive weighting approach. Moreover, we also propose a document length-based weighting strategy to favor long tweets that contain more important content. Finally, we explore a query expansion method based on Web search results to better infer user intention.

The rest of the paper is organized as follows. We first explained the pre-processing procedure in Section 2, and then describe our methods as well as the submitted runs in Section 3. Experiment results are showed in Section 4 and we conclude in Section 5.

2 Preprocessing and Index Building

Document Downloading

We used "official twitter-corpus-tools corpus downloader" to download tweets with html output form. We totally crawl 15052875 available tweets (13938553 with response code 200 and 1114322 with response code 302).

Preprocessing

Twitter is widely used all over the world and a large number of twitters are written in or include non-English characters, e.g. Chinese, Japanese, French and Arabic. Filtering out such non-English tweets will help us improve the retrieval performance and increase retrieval speed.

Most non-English text will apply characters which ASCII code is greater than 127 (128-255), e.g. Chinese, Japanese, Korean and Arabic. Thus simply removing tweets with characters which ASCII code over 127 will help filtering out non-English content. We apply this strategy and successfully remove most of non-English tweets from the database. However this strategy cannot guarantee all non-English content removed and it may mistakenly delete a few English tweets with special expression characters.

Another choice may be applying some comprehensive English dictionary to remove tweets with term not in the term list. However, regarding that micro blog is casual and informal, it will be a difficult task to category widely appeared misspelled terms from non-English characters. And more important, applying dictionary will against the constraint “not to use external resource”.

Index Building

We built the index with the Lemur toolkit from UMass. Porter stammer is applied and none stop words are removed. Tweets are treated as documents and they are sorted by their post time from old to new.

3 Method

Overview

Microblog search is a special kind of text search. It has some similarity with traditional text search, but it also has some features that are different from normal text search. For example: a tweet often contains at most 140 characters and it mostly only contains one aspect. As a consequence, traditional text-search method may not be the best choice for microblog search. Specifically, term frequency and document length normalization are important term weighting strategies in traditional retrieval models. However, they may not be very useful for microblog search because tweets are usually very short and contain only one aspect. On the other hand, IDF should still contribute to filter out non-sense terms. Based on this analysis, we simply design the following retrieval function:

$$s = \sum_{t \in Q} W(t)$$

where W(t) is the time-sensitive term weighting of term t.

Time-sensitive term weighting

Query term weight is also important in tweets search since tweets are also constituted with common non-sense terms and rare but important terms. Traditional methods calculate term weights by their occur frequency in documents which is widely known as inverse document frequency (IDF). However in our opinion, microBlog search is a dynamic rather than a static process. Thus term weight is determined by not only the background language model (IDF), but also tweet post time. In our assumption, terms occur in their popular discussed time should receive higher term weight than those in rare-discussed period. Take query "Kubica crash" as an example: term "Kubica", "crash" around Feb.6 when they are hot-discussed should receive higher term weight than "Kubica", "crash" which appear in other time.

To implement such idea, for each query term in each tweet, beside calculate its IDF in the whole collection, we also calculate its IDFs in three window sizes: 6 hours, 1 day and 3.5 days. For example, to get the term weight of a tweet posted at 12:00pm on Feb. 6. We will calculate IDFs based on tweets posted "9:00am – 3:00pm Feb. 6", "12:00am Feb.6 – 12:00am Feb.7" and "6:00pm Feb.4 – 6:00am Feb. 8".

Then we design the following term weight function:

$$IDF(w,t) = \log\left(\frac{docN(w)+1}{DF(w,t)}\right)$$

In which docN(w) is total number of documents in time window w while DF is number of documents contain term t in time window w.

$$W(t) = IDF(t) - \frac{1}{4}IDF(6hr,t) - \frac{1}{16}IDF(1d,t) - \frac{1}{64}IDF(3.5d,t)$$

To simplify the computational process, we choose window size by documents number rather than accurate time with negligible difference. (1/4 docN as 3.5 days, 1/16 docN as 1 day, 1/64 docN as 6 hours given the whole collection across two weeks)

Document length

Since micro blog may contain much spam or senseless information, filtering out these spam tweets may help us improve both the interestingness and relevant of our results. In our assumption, such spam tweets are more likely to be short and contain many low IDF terms. Hence we design the following document length favor strategy to favor long tweets with high IDF terms.

$$S = s + \beta \sum_{t \in D} IDF(t)$$

In experiments, we set $\beta = 0.5$.

Pseudo Feedback and query expansion with external search results

Query term match alone is not enough and in order to further improve system performance, latent information should be applied to expanse original queries.We designed two methods to dig out such latent information from two different sources: internal and external.

For internal source, pseudo feedback is applied: we chose all the terms of top 10 tweets as candidate and sum their TF-IDF scores in the top 10 tweets. Then 30 candidate terms with highest scores are chosen to expanse the original query with normalized weight calculated by their scores.

For external source, top search results from Yahoo are used and the expansion process is similar to the pseudo feedback we mentioned before.

4 Experiment and Results

Based on the methods we mentioned before, we submit the following 4 runs:
UDMicroIDF: only apply time-sensitive IDF on baseline
UDMicroIDFD: add document length favor strategy to UDMicroIDF
UDMicroComb1: apply pseudo feedback to UDMicroIDFD
UDMicroComb2: external resource query expansion based on UDMicroIDFD

To make a compare, we also do the following unofficial runs:
Static-IDF: apply static IDF rather time-sensitive IDF strategy
Comb1-no-length: apply pseudo feedback to UDMicroIDF without considering document length
Comb2-no-length: external resource query expansion based on UDMicroIDF without considering document length.

The results are shown at following table:

Runs	MAP	r-precision
UDMicroIDF	0.1857	0.2365
UDMicroIDFD	0.1532	0.2059
UDMicroComb1	0.1581	0.2027
UDMicroComb2	0.1700	0.2206
Static-IDF	0.1592	0.2075
Comb1-no-length	0.1738	0.2208
Comb2-no-length	0.2030	0.2206

Runs	MAP	r-precision
UDMicroIDF	0.2280	0.2521
UDMicroIDFD	0.1697	0.2051
UDMicroComb1	0.1700	0.1939
UDMicroComb2	0.1868	0.2173
Static-IDF	0.1726	0.1951
Comb1-no-length	0.2052	0.2216

Comb2-no-length	0.2162	0.2173

5 Discussion and Conclusion:

Comparing UDMicroIDF with Static-IDF, we notice a significant improvement on both MAP and r-precise scores. It shows that our time-sensitive strategy can help us improve the system performance for most queries especially some time-sensitive queries. For example, for query MB011 "Kubica crash" which is very time sensitive, the MAP score of our time-sensitive system is 0.77 comparing to 0.28 of static-IDF. It will be very interesting to do more research about the time-sensitive approach and other application of it.

Document-length-favor seems not to be a good strategy in this year's micro blog track and it hurts the performance a lot. Our unofficial runs "Comb1-no-length" and "Comb2-no-length" show some significant improvement to runs "UDMicroComb1" and "UDMicroComb2" which apply document-length-favor. The strategy seems too aggressive, but it should be interesting to discuss its usage in other collections.

Pseudo Feedback and external source query expansion help us dig out some latent information from original query. For example, both "UDMicroComb1" and "UDMicroComb2" show a good improvement (0.31, 0.26vs0.14) for query MB002 "2022 FIFA soccer", because they dig out the important latent information "Qatar". However the query expansion of both methods are not always safe, they may hurt the performance for some queries. We would do more research to improve their stability.