University of Waterloo at TREC 2015 Microblog Track

Luchen Tan Adam Roegiest Charles L.A. Clarke
School of Computer Science
University of Waterloo, Canada
{luchen.tan, aroegies, claclark}@uwaterloo.ca

ABSTRACT

Given a topic with title, narrative and description, we start by building a language model for the topic. The top 1000 tweets were retrieved from Twitter commercial search engine by applying the title of the topic as a query. We exploit pseudo relevance feedback technologies to estimate probability distributions of each term in the topic, then comparing these probabilities with a background distribution model. We select the highest different terms as our expanded query terms. We then generate a vector for each topic, the features of the vector are non-stop word title terms, selected narrative terms and query expansion terms. Different weights are assigned to the different types of terms. Since we are allowed to deliver at most 10 tweets every day, and the latency time can not exceed 100 minutes, we solve the tweet notification scenario as a multiple-choice secretary problem. Two different solutions were tested.

1. INTRODUCTION

TREC 2015 Microblog Track included a completely different task from previous years, which was the real-time filtering task. The goal of this task was to monitor the real Twitter streaming data and determine whether or not to push each tweet to a user. Each user's interest profile was given in the format of a traditional TREC topic; below is an example:

```
<top>
<num> Number: MB10001
<title> crossword puzzle tournaments
<desc> Description:
Return announcements of and commentary
regarding crossword puzzle tournaments.
<narr> Narrative:
The user likes to do crossword puzzles
and intends to participate in upcoming
crossword puzzle tournaments. She wants
to see any Tweets that relate to a
tournament: Tweets that announce a
tournament or give logistical information;
Tweets about a tournament from its
participants including Tweets that express
anticipation of the tournament or
travelling to/from the tournament;
Tweets that comment on the quality
of a tournament; etc.
</top>
```

Two task models were provided:

- **Scenario A**: Push notifications on a mobile phone. Participating systems should identify interesting tweets based on the user's interest profile(topic), and determine whether or not to push a notification to the user's mobile phone. Such notification is expected to be triggered within 100 minutes after the tweet is created. A maximum of 10 tweets is allowed to be pushed by a system to a single user per day.
- **Scenario B**: Periodic email digest. Participating Systems should identify tweets based on the user's interest profile, and aggregate them into an email. The email should be periodically sent (i.e. every day) to the user.

Scenario A is a real-time filtering task, but it does not require an on-line decision. It means that participating systems do not need to decide whether or not push notification for a tweet before seeing the subsequent tweets. A 100-minute latency time is permitted. Thus, in addition to the normal retrieval processes, the choice of pushing strategy is also significant. However, Scenario B is more like a retrieval task based on a one-day tweet collection.

In this report, we describe our algorithm, decision and strategies of solving these two tasks.

2. PRE-PROCESSING TWEETS

Tweets are relative short and informally written. To better understand tweets, a tokenizer that was designed for English Twitter text was used. We applied a simple method to remove non-English tweets and partially kept retweets. No stemming or stop word removal was utilized on tweets, but tweet quality was considered.

2.1 Language detection and Retweets

Tweets written in a language other than English would be judged as not relevant based on guidelines of Microblog Track. To obtain only English tweets, we read the language tag for each tweet feed by the streaming API, and only kept

the English ones. In addition, we remove all NON-ASCII characters from the tweets, which also helps remove non-English tweets. The treatment of retweets is different from previous years. The retweets returned will be replaced by the original tweets that were retweeted. In this case, all additional commentary in the tweets will be ignored, and created time will be considered as the original tweets' time. Thus, there will still be high risk to return a retweet given the original tweets are posted older than the retweets. In our submitted runs, we ignore all retweets in Scenario A and keep them in Scenario B.

2.2 Tokenizer

Our tokenizer was based on Twokenize, which is a tokenizer designed specific for English tweet text. It is part of CMU's Tweet NLP project[1]. For the hashtags, we kept the base term without the '#' symbol as a token, as well as the hashtags themselves. Since tweets are short, the tokens were treated as token set, which means each token was only counted once in each tweet.

2.3 Tweet Quality

With a tweet quality filter, we can filter out low quality tweets and some of junk tweets. Although tweets are at most 140 characters, short tweets are hard to extract topics from. We set an arbitrary threshold to detect high quality tweets, such that, any tweet that has fewer than 5 tokens, or tweets with more than three hashtags are treated as low quality tweets. These low quality tweets will be ignored.

3. USER PROFILES

User profiles are made up of three fields: titles, descriptions and narratives. Titles contain several key words or key phrases; descriptions are one-sentence statements of the users' information needs; narratives are paragraph-length descriptions of the tweets that the users want to receive. To implement the filtering tasks, we built a term vector for each user profile, and assigned different weights to different types of terms.

3.1 Feature Vector

Titles were tokenized by space and punctuation. The stop words were removed from these tokens. Additionally, for noun tokens, derived both singular and plural forms of these noun tokens. We processed the title of each user profile, and added processed tokens to the feature vectors of this user profile. To extract important words from descriptions and narratives. we applied a pointwise K-L divergence method [2, 3], which was also applied later in the process to generate expansion terms. We took all the 225 user profiles as a background model. For each user profile, description and narrative sentences were combined together as a foreground model of the profile. To discover the most significant tokens in each user profile, we calculated pointwise K-L divergence and ranked the scores for each token in the profile:

$$p_t \log(p_t/q_t), \tag{1}$$

where p_t is the relative frequency of term t in the profile foreground model and q_t is the relative frequency of term t in the overall background model. We took the top-10 terms from this ranking to form a set important narrative and description terms. In the runs that used narratives and descriptions, these terms were added to the feature vectors.

[1]http://www.ark.cs.cmu.edu/TweetNLP

3.2 Pseudo-Relevance Feedback

We utilized the same idea of pointwise K-L divergence to generate expansion terms from pseudo-relevance feedback. We took a large collection of historical tweets as our background model. The collection was collected through the Twitter Streaming API from November 2013 to March 2015. We restricted the collection to English-language tweets on the basis of the language field associated with each tweet. There are in total approximately 291 million tweets in the collection. To build a foreground model for each user profile, we took the title terms as query, and searched the query in Twitter search engine. The top 1000 tweets were crawled on the search result pages as our foreground model. In the top retrieved tweets, URLs in each tweet were replaced by their web page titles, if the <title> tag existed in the HTML source of the web pages. To keep the foreground models up-to-date, we re-built the foreground models every day during the evaluation period. A ranking of terms can be generated by the scores for each term in each foreground model. We took the top-10 normal terms and top-5 hashtags from the ranking and added them to the feature vector of each profile.

4. RELEVANCE SCORING

A simple vector space model was applied to calculate relevance score between each incoming tweet and each user profile. We set arbitrary weights to different types of feature terms based on our previous experience. There were three types of feature terms: title, narrative+description, expansion. We denote types of feature terms by $i = \{t, n, e\}$, where t stands for title, n stands for narrative+description and e stands for expansion.

$$rel = \frac{\sum_{i=\{t,n,e\}} w_i * N_i}{\frac{N_t}{|T|}} \tag{2}$$

Here, w_i denotes the weight for type i and N_i stands for the number of times type i feature terms appeared in the tweet. In the denominator, we normalized the score by ratio of title terms appeared in the tweet, where $|T|$ denotes the total number of title terms of the user profile. We assumed that the more title terms appeared in the tweet, the more relevant the tweet would be. A naive and arbitrary method of deduplicating tweets was applied, which simply the unigram token overlap between candidate and previously pushed tweets. Two tweets with over sixty percent of overlapping unigrams would be counted as similar tweets.

5. PUSH NOTIFICATION STRATEGIES

The secretary problem is a famous optimal stopping problem. The basic form is described as hiring the best secretary out of n rankable applicants for a position. The applicants are interviewed one by one and the employee has to make immediate decision after each interview. An applicant cannot be recalled once rejected. A variation problem of multiple-choice secretary problem [1, 4] is that up to k secretaries are allowed to be hired during the interview period. A decision need not be made immediately after each interview. But it's better to be as fast as possible. We determined

Strategy	RunID	ELG	nCG
Title only+Dynamic Emission	UWaterlooATDK	0.3150 (0.2366 - 0.3933)	0.2679 (0.1864 - 0.3494)
Title only+Fix time	UWaterlooATEK	0.2654† (0.1892 - 0.3415)	0.2365† (0.1576 - 0.3154)
Title+Narrative+Description+Fix time	UWaterlooATNDEK	0.2470† (0.1796 - 0.3144)	0.2170† (0.1474 - 0.2865)

Table 1: Results for Scenario A (push notification) for submitted runs with 95% confidence intervals. † denotes $p < 0.01$ in a paired t-test with run UWaterlooATDK.

Strategy	RunID	nDCG
Title only	UWaterlooBT	0.2200 (0.1684 - 0.2716)
Title+Narrative+Description	UWaterlooBTND	0.2196 (0.1682 - 0.2710)

Table 2: Results for Scenarion B (email digest) for submitted runs with 95% Confidence Intervals.

the Scenario A is a instance of the multiple-choice secretary problem. Two different strategies were tested by our system.

5.1 Strategy #1: Fix Time Window

Under this algorithm, we returned tweets periodically. A threshold based on historical results was set for each user profile, which was the score of the top 50^{th} ranked tweet returned in the previous day. Every day during the evalusation period, we updated the threshold. We selected the highest score in tweet that was also higher than the threshold in every k minutes, where k was smaller than 100 minutes. Pushed tweets not be redundant were those that were not previously returned nor were they redundant. If there was not any tweet returned during a k minute window, the slot would be carried to the next k minute window. Whenever we had returned 10 tweets of the profile one day, we would stop returning for that day.

5.2 Strategy #2: Dynamic Emission

Our second algorithm used two thresholds for each user profile: k_0, and k_1, where $k_0 \leqslant k_1$. k_0 was the lower bound of relevant scores for the profile. Here we made several assumptions: any tweet that had score higher than k_1 was relevant to the user's information need; tweets that were scored lower than k_0 was not relevant; tweets with scores between k_0 and k_1 were considered potentially relevant. The value of k_0 and k_1 were based on the scores of returned tweet in the previous day, and would be updated every night. A dynamic k minute window was used for this algorithm. Algorithm 1 shows the strategy we applied.

6. RESULTS

For Scenario A, Table 1 reports the performance of our three submitted runs with 95% confidence intervals and the results of conducting pairwise t-tests (where $p < 0.01$) between all runs[2]. As it turns out, UWaterlooATEK significantly outperforms both other runs, indicating that the dynamic emission strategy was successful in identifying relevant tweets. The results for Scenario A also indicates that using all parts of the information profile may not be of benefit and their inclusion may in fact negatively impact performance. Although, this impact was not significant between UWaterlooATEK and UWaterlooATNDEK.

Table 2 reports our results for Scenario B, which simulated an email digest of interesting tweets. The results for Scenario B are not all that surprising and are inline with what we observed for Scenario A. That is, inclusing of the narrative and description fields did not aid performance. Furthermore, our approach for Scenario B of returning tweets for the sake of returning tweets was likely less than ideal and we should have likely adopted a more conservative strategy as we did in Scenario A.

[2]No multiple hypothesis correction was performed.

7. CONCLUSIONS

In this work we presented a method for doing dynamic tweet emission in a real-time scenario based upon infomration profiles. We have observed that making use of all pieces of an information profile may not be of much benefit over just using the title alone. In addition, using this dynamic tweet emission algorithm was significantly better than either other simple fixed emission solution. In spite of its relatively simple nature, the good performance of the dynamic emission strategy bodes well for future work that explores the parameter space better than we were able to do for TREC this year.

8. REFERENCES

[1] R. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 630–631. Society for Industrial and Applied Mathematics, 2005.

[2] L. Tan and C. L. Clarke. Succinct queries for linking and tracking news in social media. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1883–1886. ACM, 2014.

[3] T. Tomokiyo and M. Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pages 33–40. Association for Computational Linguistics, 2003.

[4] X. Zhao and K. Tajima. Online retweet recommendation with item count limits. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 1, pages 282–289. IEEE, 2014.

```
Data: Streaming of tweets
Result: Relevant tweets to the user profile
begin
    max_waiting ⟵ k mins
    cur_time ⟵ now
    last_time ⟵ cur_time − max_waiting
    time_out ⟵ 0
    k_0 ⟵ score of top10 tweet yesterday
    k_1 ⟵ score of top5 tweet yesterday
    candidate ⟵ highest score tweet between cur_time and last_time with score ⩾ k_0
    while True do
        curr ⟵ highest score tweet between now and last_time with score ⩾ k_0
        curr_time ⟵ time of curr
        /* If a tweet has score greater than k_1, report it immediately, and restart the waiting
           window.                                                                              */
        if curr > k_1 then
            Report curr
            last_time ⟵ curr_time
            time_out ⟵ 0
        end
        /* If a tweet is better than the candidate tweet, replace the candidate with the
           current tweet, and restart the waiting window.                                       */
        else if curr ⩾ candidate then
            candidate ⟵ curr
            last_time ⟵ curr_time
            time_out ⟵ 0
        end
        /* If a tweet is worse than the candidate tweet.                                        */
        else
            /* Have not reached maximum waiting time, keep waiting                              */
            if time_out < max_waiting then
                time_out ⟵ now − curr_time
            end
            /* Report the candidate, which is the best one during the waiting window.  Then
               restart the waiting window.                                                      */
            else
                Report candidate
                last_time ⟵ now
                time_out ⟵ 0
            end
        end
    end
end
```

Algorithm 1: Dynamic Tweet Emission Algorithm