BJUT at TREC 2017: Real-Time Summarization Track

Qingwei Meng[1], **Kai Wang**[1], **Zhen Yang**[1]✉
1. College of Computer Science, Faculty of Information, Beijing University of Technology, China
✉yangzhen@bjut.edu.cn

Abstract

This paper describes our effort for the TREC Real-Time Summarization task in 2017, which is pushing notifications to the users mobile phone (Task A) and submitting periodic email digest according to tweets posted on the previous day (Task B). In essence, both of the tasks are about a process of extracting the relevant data from tweets stream with respect to the users' interest profiles. For each task we submitted three runs, in this paper, we presented the system framework and experimental results briefly.

Introduction

We are in an era of information, that is filled with varieties of data every day. As one of the most popular social network platforms, Twitter has a total of more than 300,000,000 users, millions of tweets are posted on the web every day. In this case, its of great importance to help people find useful and non-redundant information the user wanted.

The TREC 2017 Real-Time Summarization (RTS) Track aims to explore techniques and systems that automatically monitor streams of social media posts such as Twitter to keep users up to date on topics of interest. We might think of these topics as "interest profiles", specifying the user's prospective information needs. There are two tasks about the Real-Time Summarization (RTS) Track this year:

- **Track Scenario A**: Push notifications. As soon as the system identifies a relevant post, it is immediately sent to the user's mobile device via a push notification. At a high level, push notifications should be relevant (on topic), novel (users should not be pushed multiple notifications that say the same thing), and timely (provide updates as soon after the actual event occurrence as possible).
- **Track Scenario B**: Email digests. Alternatively, a user might wish to receive a daily email digest that summarizes "what happened" that day with respect to the interest profiles. At a high level, these results should be relevant and novel; timeliness is not particularly important, provided that the tweets were all posted on the previous day.

Since the core tasks in A and B are the same, the similar approaches are used in task A and B. The paper is mainly organized as follows: in Section 2, we present our approach for Tweets message recommendation based on users interest profiles. In Section 3, we report our experimental results and in Section 4, the concluding remarks are given.

Real-Time Summarization Framework Design

To address the two tasks mentioned above, we construct a system to catch the relevant tweets. Figure 1 shows our system framework. It consists mostly of five parts: Query Expansion, Tweet Preprocess, Relevance Verification, Redundancy Detection, Ranking and Submission.

Query Expansion

For this year, 188 interest profiles are given in the same form of "topid" "title" "description" "narrative". The "title" contains a short description of the information need, similar to what users would type into a search engine. The "description" and "narrative" are sentence- and paragraph-long elaborations of the information need, respectively.

In order to find the synonyms and related keywords from the interest profiles to measure the relevance better, we do the query expansion. For every topic, we feed the topic title into the search engine (e.g. Bing News Search, Wikipedia) and get top 100 search results snippets as the expanded items.

Tweet Preprocess

During the evaluation period (from July 29, 2017 00:00:00 UTC to August 5, 2017 23:59:59 UTC, 8 days in total), we listened to the tweet stream using the Twitter streaming API. We can receive a log file every minute, and in each log file it contains about 1000 tweets, that is to say more than 1,440,000 tweets every day.

In consideration of efficiency, it is necessary to preprocess a vast amount of tweets to filter out most of the trash or irrelevant tweets and transform each tweet into a standard and clean format.

First we discard date, time, tweetid, @, RT and URL to get pure tweet contents. Then we identify and filter out the crash tweets. If a tweet meets one or more these conditions below, we regard it as trash tweet and filter out it.

- All characters are capital (typical trash tweet)
- All characters are single letters instead of words (e.g. a b c d e f g)

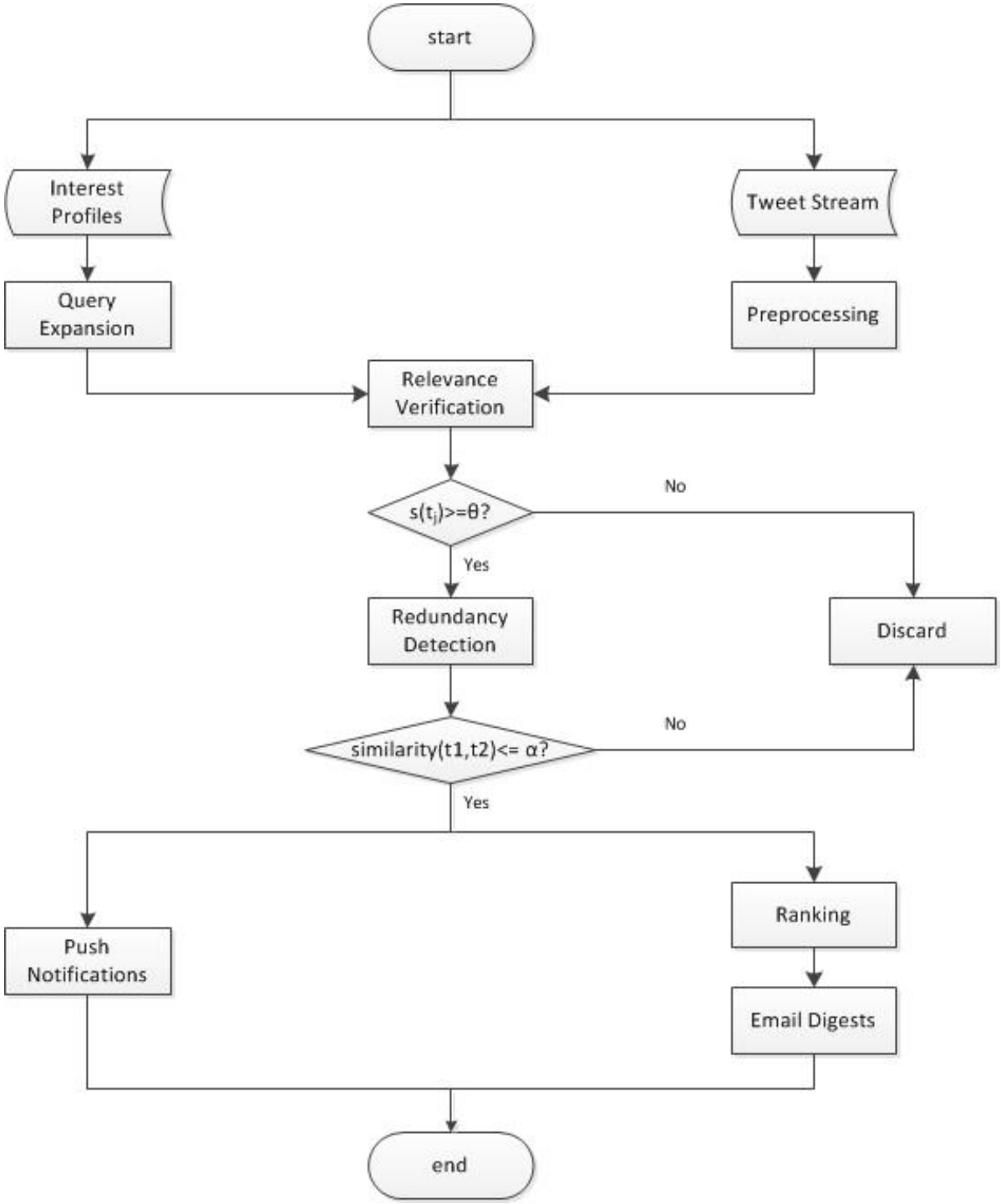

Figure 1: System Framework.

- The length of text is less than 20 (too few words to provide enough information)
- The length of unique words is no more than 3 (e.g. I love you I love you I love you I love you I love you I love you I love you)

Relevance Verification

According to the guideline 2017, we submitted 3 runs finally. The most difference among the 3 runs is about the part of Relevance Verification and Redundancy Detection. For run1, we mainly use MATLAB and its toolbox - TMG (Zeimpekis and Gallopoulos 2007) to classify the prepro-

cessed tweets into relevant topic. We use the Indexing Module in TMG toolbox to process A (the interest profiles together with the search results from query expansion) and B (the preprocessed tweets) to get the term document matrices (tdms) C (C_1, C_2, C_3, ...) and D respectively. By matrix multiplication (C_iD), we can get a score $s(t_j)$ with respect to 188 different topics. We set threshold θ= 0.24 based on a previous work, if the calculated score $s(t_j) \leq \theta$,we regard it as a irrelevant tweet and discard it. According to The principle of Naive Bayesian classification, a tweet is classified into the topic in which it has the best score.

For run2 and run3 (Wang and Yang 2016), we train a classifier based on labelled Microblog Track (2015) and Real-Time Summarization Track (2016) past data. Label 1 presents relevant tweet and label 0 presents irrelevant tweet.

Redundancy Detection

As we all know, users can retweet any tweets they interested in, or because of the occasionality, it is very likely to produce some similar or the same tweets. To avoid the simple copy and high similarity between the candidated tweets, it is necessary to take some measures to do Redundancy Detection.

For run1, our method assumes that the similarity between two tweets is merely determined by occurrences of their common vocabulary (Tan, Luo, and Li 2016). The similarity for redundancy detection is defined in Equation 1. In this equation, t_1 and t_2 are the words in tweet $t1$ and $t2$ respectively, the numerator of the fraction represents the sum of word length of the intersection of two tweets, and the denominator of the fraction represents the sum of word length of the union of two tweets. We set threshold α= 0.6, if the calculated similarity between two candidated tweets $t1$ and $t2$ similarity($t1$,$t2$)$\geq \alpha$, we regard it as a redundancy tweet and discard it.

$$similarity(t_1, t_2) = \frac{union\_score(t\_1, t\_2)}{intersection\_score(t\_1, t\_2)} \quad (1)$$

Ranking and Submission

According to the official guideline 2017, each system is allowed to push at most ten tweets per interest profile per day for task A. This per-day tweet delivery limit is to model user fatigue in mobile push notification. To address the problem, we design a counter in our system, for each topic if the number of the posted tweets reaches the limit, stop pushing notifications to the RTS evaluation broker. There is also a limit of 100 tweets per interest profile per day for task B. So we collect all the candidate tweets in a day, calculate their scores and classify them into the most likely topic, then arrange the scores of the same topic in a descending order and submit the results via Batch Upload to NIST.

Experiment Results

In this section, we introduce the evaluation methods (Lin et al. 2016) and our results.

Evaluation Measures

For each tweet, the user makes one of three judgments: relevant, if the tweet contains relevant and novel information; redundant, if the tweet contains relevant information, but is substantively similar to another tweet that the assessor had already seen; not relevant, if the tweet does not contain relevant information. From these counts, we computed $strict$ precision, defined as:

$$\frac{relevant}{relevant + redundant + not\ relevant} \quad (2)$$

as well $lenient$ precision, defined as:

$$\frac{relevant + redundant}{relevant + redundant + not\ relevant} \quad (3)$$

Expected Gain (EG) for an interest profile on a particular day is defined as follows:

$$EG = \frac{1}{N} \sum G(t) \quad (4)$$

where N is the number of tweets returned and $G(t)$ is the gain of each tweet: Not relevant tweets receive a gain of 0; Relevant tweets receive a gain of 0.5; Highly-relevant tweets receive a gain of 1.0.

Normalized Cumulative Gain (nCG) for an interest profile on a particular day is defined as follows:

$$nCG = \frac{1}{Z} \sum G(t) \quad (5)$$

where Z is the maximum possible gain (given the ten tweet per day limit). The gain of each individual tweet is computed as above.

Gain Minus Pain (GMP) is defined as follows:

$$GMP = \alpha \cdot \sum G - (1 - \alpha) \cdot P \quad (6)$$

The G (gain) is computed in the same manner as above. Pain P is the number of non-relevant tweets that the system pushed, and controls the balance between the two. Scenario B runs were evaluated in terms of $nDCG$ as follows: for each interest profile, the list of tweets returned per day is treated as a ranked list and from this $nDCG@10$ is computed. Note that in this scenario, the evaluation metric does include gain discounting because the email digests can be interpreted as ranked lists of tweets. Gain is computed in the same way as in scenario A with respect to the semantic clusters. Systems only receive credit for the first relevant tweet they report from a cluster.

Results Analysis

We got two results from the organizer lately. Results of task A and B are shown in Table 1 and Table 2. Columns represent 3 runs and the median scores, and rows represent some related evaluation measures mentioned in the previous chapter.

Results show that there is a great gap between our system performances and the median scores, especially the $EG1$ and $nCG1$ of BL1. Fortunately the latency of BL1 is satisfactory relatively.

Table 1: Task A Performances.

runtag	EGp	EG1	nCGp	nCG1
BJUT-BL1-04	0.1692	0.0774	0.1711	0.0793
BJUT-BL2-05	0.1837	0.1625	0.1809	0.1598
BJUT-BL3-03	0.1602	0.1225	0.1636	0.1258
median scores	0.2194	0.1951	0.2095	0.1826

Table 2: Task B Performances.

runtag	nDCGp	nDCG1
bjut_tmg	0.1796	0.1456
bjutgs	0.0746	0.0746
bjutg	0.1169	0.1169
median scores	0.2194	0.1865

Conclusion

In this paper, we present the implementation details of our runs for Real-Time Summarization Track. Results show that our method is effective to some degree, however there is still a great gap between our system performances and the median scores. In the future work, we will concentrate on how to improve the accuracy of the system.

Acknowledgments

This research was supported by the Data Mining & Security Lab and professor Yang from Beijing University of Technology.

References

Lin, J.; Roegiest, A.; Tan, L.; McCreadie, R.; Voorhees, E.; and Diaz, F. 2016. Overview of the trec 2016 real-time summarization track. In *Proceedings of the 25th Text REtrieval Conference, TREC*, volume 16.

Tan, H.; Luo, D.; and Li, W. 2016. Polyu at trec 2016 real-time summarization. In *TREC*.

Wang, K., and Yang, Z. 2016. Bjut at trec 2016: Real-time summarization track. In *TREC*.

Zeimpekis, D., and Gallopoulos, E. 2007. Text to matrix generator users guide. *Department of Computer Engineering and Informatics, University of Patras, Greece*.