S.T at TREC 2017: Real-Time Summarization Track

Junjie Xiong   Dongdong Xiang   Qian Guo   Haiguang Chen*

Chhg@shnu.edu.cn

College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, China

Abstract— **This paper presents the participation of Shanghai Normal University to the TREC 2017 Real-Time Summarization (RTS) Track. We adopted three different composed methods by applying various factors, i.e., count, cosine and distance to measure relevance between a tweet and a given topic. By setting static relevance threshold for each run, we selected the most relevant but non-redundant tweets and then pushed them to user's phone in Scenario A. For Scenario B, we used a similar but much simpler approach. The evaluation results showed that there was still a long way to go in practice. Nonetheless, some progress has been made. We submitted three runs for both scenarios. This paper demonstrates the implementation details and official evaluation results of our system.**

I. INTRODUCTION

The identification of text relevance is an important field of study, with social media platforms such as Twitter garnering the interest of researchers in real-time text processing as well as in social sciences. Getting the potential information needs over continuous stream of texts is one of the research topic. Real-Time Summarization (RTS) is a track at the 2017 Text Retrieval Conference (TREC) that commit to advance the system of potential information demands.

The evaluation was conducted from July 29th 00:00:00 UTC to August 5th 23:59:59 UTC. During the evaluation period, all participants must monitor the Twitter Sample Stream about predefined "interest profiles" via the Twitter Streaming API. In this track, we hold that users who have a number of "interest profiles" represent the needs of prospective information. All the participating systems should automatically filter the twitter sample stream in order to help users to keep eyes on the occurrence of a specific event. For example, a woman is a breast cancer survivor and wants to keep current with the latest advancements in research for new treatments for breast cancer. Two types of disseminating updates in this year's track:

- **Scenario A: push notifications.** As long as the system identifies a relevant tweet of a topic of "interest profiles", it will be pushed in real time. At a high level, relevant tweets pushed to the user should be novel(users should not receive multiple notifications that say the same thing), and timely(after the actual event occurs, the system provides the updates as soon as possible).
- **Scenario B: email digest.** Alternatively, a user might want 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.

In this paper, we present our real-time summarization system as a participant in TREC-2017 Real-time Summarization Track. In our system, we used Tweepy to monitor the Twitter stream. The system identifies the tweet that is relevant to a topic from the predefined "interest profiles", taking novelty and timely into account at the same time. After that, we implemented relevance models with a diverse static relevance threshold(this threshold was acquired from our early experiments before the evaluation). Traditionally, a real-time system should use simple and efficient approaches that can process data in parallel. But the method is complex. We use process-intensive queries(one tweet at a time for a topic) instead of parallel processing, which helps us improve and simplify our design. Challenge due to the nature of tweet, it makes short text language processing quite different from the traditional natural language processing. The basic way to tackle such a problem is to enrich the very short text by semantically related terms. By this method, we upgraded the user's "interest profiles". Stream rate limit also needs to be taken into account in practice. After we analyzed the fields of stream and this year's rules, two-level filtering and Redis queue had been integrated into our system to relieve the rate crisis.

The rest of this paper is organized as follows. The system design is posted in Section II. Section III describes the approaches used in our system. The results of our participation are discussed in Section IV and conclusions are provided in Section V.

II. SYSTEM DESIGN

In this section, we mainly describe the architecture of our system, which is shown in figure 1. Actually, we simplified it to two parts.

A. *Offline Module*

This part contains three components: (1)external resource (2)interest profile (3) upgrading model. We utilize the external resources to obtain semantically related terms and incorporate with primary "interest profiles" to generate an upgraded "interest profiles".

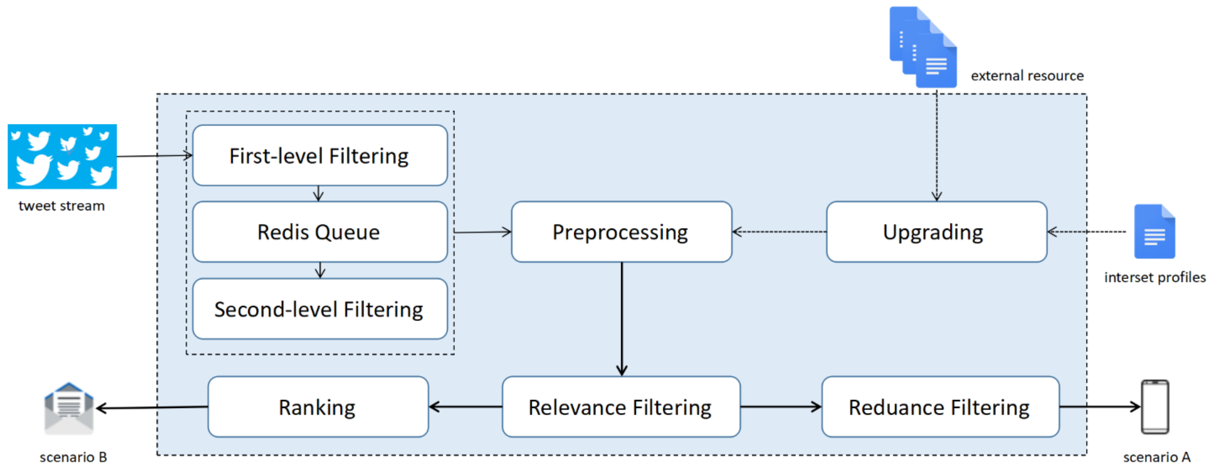


Fig. 1: System Framework

B. *Online Module*

This part contains six components: (1)tweet stream (2)two-level filter (3)Redis queue (4)relevance filter (5)redundancy filter (6)ranking. We monitored and processed the tweet stream continuously, then used two-level filter model to obtain more potential relevant tweets. Considering that the stream has rate limitation, we employed Redis queue. As soon as the relevance filter finds a tweet relevant, it will immediately push the tweet to redundancy filter model and ranking model to make decisions.

III. APPROACH

Query Expansion

The length of tweets is strictly enforced, so they bring less information directly. Traditionally, query expansion is the most frequently used to advance retrieval hit ratio. In our approaches, we only feed "title" to make an expansion and then get three types of updates during this period. What constitutes an interest profile can be seen from Table I.

TABLE I: Interest profile

Interest profile			
topid	title	description	narrative
RTS48	Panera Bread	What are people ordering at Panera Bread?	The user wishes to track what people order at Panera Bread.

After making the expansion above we removed stopword by using Spacy and calculated the TF-IDF score based on "The Signal Media One-Million News Articles Dataset"[1] for every term, then top25, top20 and top10 terms were returned respectively. The expansion method are as follows.

- Bing News Search API: The first technique applies Bing New API and TF-IDF to get back the top50 related sentences(also called snippet) and top25 terms successively. We then take it as the query expansion document set $bing_t$.
- Twitter Search API: This technique is similar to the previous. The difference is that we use Twitter Search API and TF-IDF to obtain top20 terms as query expansion document set $twitter_t$.
- Interest Profiles: We simply mix "title","narrative" and "description" up together and take it as query expansion document set $interest_t$.

After that we got three types of expansion document sets for every topic, namely, ($bing_s$,$bing_w$,$bing_t$), ($twitter_s$,$twitter_w$,$twitter_t$) and ($interest_s$,$interest_w$,$interest_t$). Subscript s denotes sentence, w denotes word after removing stopword and t denotes top score word after doing TF-IDF.

Two-level data filter

1) First-level filter: As a rule, non-English tweet is marked as irrelevant. So we discarded the non-English tweet according to the Twitter's language detection field 'lang'. If $\alpha/\beta < 0.8$, we also discard it. Here α denotes the length of ASCII characters, β denotes the length of this tweet. As soon as the retweet's identification is fetched, it will immediately replace it by original tweet. Then we discard the tweet if the character length is less than 30. A tweet that survives from the first-level filter will be passed to Redis queue.
2) Second-level filter: At this stage, we fetched data from Redis queue. In consideration of relevancy and efficiency, we applied keyword filtering to filter out the obviously irrelevant tweet and pornographic tweet. We chose keywords from each topic of interest profiles based on dropped stopword's document sets and pornographic list made manually. If a tweet doesn't match any one of these, then it would be discarded.

Redis FIFO Queue

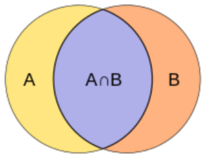
As we all know Redis is an in-memory database open-source software project implementing a networked, in-memory key-value store with optional durability, "FIFO" is an acronym which most commonly stands for "first in, first out". We implemented a Redis with FIFO queue to cache stream data.

Data Processing

Once we got a tweet that survives from two-level filtering, we immediately applied regular expressions to search for and replace the insignificant characters or url only keeping English words.

Redundancy Filter and Push Strategy

For Scenario A, we found that most of the redundant tweets are not rephrased but simply copies of the original tweet. This means the efficiency and accuracy can be improved with adopted normal similarity algorithm. Hence we used the Jaccard index to measure the similarity between finite sample sets. Venn diagram is shown on the right. Our similarity algorithm is defined as the size of the intersection divided by the size of the union of the two tweets. $Similarity(A, B) = intersection(A, B)/union(A, B)$. This method is convenient for real-time application because it requires less computation overhead. As for Scenario B, we used the same redundancy measurement with Scenario A.



In our system, we only pushed the highly-relevant tweets to Scenario A, and for Scenario B we pushed both relevant and highly-relevant tweets. The main difference between the relevant and highly-relevant is the size of static relevance threshold.

Ranking Strategy

Relevant score and redundant score add together to give a ranking strategy in our system. Every tweet has two scores according to system scheme. The first score comes from relevance filter and shifts over topics. The second score comes from function $redundancy(t_1, t_2) = (1 - r_1)r_2$, where r_1 is $Similarity(t_1, t_2)$, r_2 is a base which determined by early experiments. After translating the two scores into the same range before calculation, we add them up and get back the ranking score.

Run 1: *Counting Features*

This Run implemented a simple but efficient strategy to estimate the relevance between an interest profile and a tweet. Before proceeding to describe this run, we first introduce quite a few notations. We would like to use tuple (q_i,t_i) to denote the ***i***-th tweet and expansion terms, where q_i is the query,t_i is the expansion terms. We use function ngram(s,n) to extract string/sentence's n-ngram(splitted by space), where $n \in 1, 2, 3$ if not specified. For example,

$$ngram(i\ love\ you, 2) = [i\ love, love\ you]$$

- $q_u b$:The count of $ngram(query, 1) \cap ngram(bing_w, 1)$.
- $q_b t$:The count of $ngram(query, 1) \cap ngram(twitter_w, 1)$.
- $q_t i$:The count of $ngram(query, 1) \cap ngram(interest_w, 1)$.
- $q_u b$:The count of $ngram(query, 2) \cap ngram(bing_w, 2)$.
- $q_b t$:The count of $ngram(query, 2) \cap ngram(twitter_w, 2)$.
- $q_t i$:The count of $ngram(query, 2) \cap ngram(interest_w, 2)$.
- $q_u b$:The count of $ngram(query, 3) \cap ngram(bing_w, 3)$.
- $q_b t$:The count of $ngram(query, 3) \cap ngram(twitter_w, 3)$.
- $q_t i$:The count of $ngram(query, 3) \cap ngram(interest_w, 3)$.
- q_u :The number of $ngram(query, 1)$.
  So,the relevance score for a specify topic is

$$\begin{aligned} RS =&(q_u b + q_u t + q_u i + q_b b + q_b t \\ &+ q_b i + q_t b + q_t t + q_t i + q_u)/10 \end{aligned} \tag{1}$$

The threshold we used in this run is more than **3.6**.

Run 2: *Similarity Based Features*

We extracted various TF-IDF features and the corresponding dimensionality reduction version via SVD. Then we computed the basic cosine similarity.

- **TF-IDF Feature**
  1) Common Vocabulary. Note that to ensure the TF-IDF feature vectors of $\{bing_t, twitter_t, interest_t\}$ are projected into the same vector space, we first concatenated $\{bing_t, twitter_t, interest_t\}$, and then fit a TF-IDF transformer to obtain the common vocabulary. We then use this common vocabulary to generate TF-IDF features for $\{bing_t, twitter_t, interest_t\}$,respectively.
  2) Individual Vocabulary. In this way, we fit TF-IDF transformer for $\{bing_t, twitter_t, interest_t\}$, separately, with individual vocabulary.
- **SVD Reduced Features**
  1) Common SVD. We first concatenated the TF-IDF vectors of $\{bing_t, twitter_t, interest_t\}$ (using common vocabulary), and fit a SVD transformer to obtain the common vocabulary. We then use this common vocabulary to generate SVD features for $\{bing_t, twitter_t, interest_t\}$, respectively.
  2) Individual SVD. We fit a SVD transformer for $\{bing_t, twitter_t, interest_t\}$, separately.
- **Cooccurrence Features**
  In this way, we extracted cooccurrence terms between query and $\{bing_t, twitter_t, interest_t\}$, then count number.
  1) query terms and $bing_t$.
  2) query terms and $twitter_t$
  3) query terms and $interest_t$

After that, we use basic cosine to compute relevance.

$$RS = \cos(\theta) = \frac{\alpha \cdot \beta}{\|\alpha\| \cdot \|\beta\|} = \frac{\sum_{i=1}^{n} \alpha_i \times \beta_i}{\sqrt{\sum_{i=1}^{n} (\alpha_i)^2} \times \sqrt{\sum_{i=1}^{n} (\beta_i)^2}} \tag{2}$$

Then we added all of them, and took the average. In our system, we set the threshold more than **0.52**.

Run 3*: *Distance Features

To be brief, the core task is to determine whether a real-time tweet is relevant to a topic or not. So we believe this means that they are relevant if their distance is short else not. So we use two distance metrics.

Dice distance

$$Dice\_Dis(A, B) = \frac{2|A \cap B|}{|A| + |B|} \tag{3}$$

Jaccard distance

$$Jaccard\_Dis(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4}$$

Here A and B denote two sets or string in our system. For each distance metric, two types of feature are computed. we declare that the function $D_d(a, b)$ and $D_j(a, b)$ compute the distances between a and b.

- **Set based distance feature**
  1)$D_d(ngram(query, n), ngram(bing_t, n))$
  2)$D_j(ngram(query, n), ngram(bing_t, n))$
  3)$D_d(ngram(query, n), ngram(twitter_t, n))$
  4)$D_j(ngram(query, n), ngram(twitter_t, n))$
  5)$D_d(ngram(query, n), ngram(interest_t, n))$
  6)$D_j(ngram(query, n), ngram(interest_t, n))$
  For example, the set 1 is **"hpv vaccine, vaccine effect, effect concern, concern vaccinate"**, set 2 is **"hpv vaccine, vaccine concern, concern user, user effect"**. So the D_d=$\frac{2}{8} = 0.25$, D_j=$\frac{1}{7} \approx 0.14$
- **String based distance feature**
  There is a little difference with above, the minimal calculation unit for set-based feature is list, but for string-based distance feature is string.

After that we added all of them and took the average. In our system, we set the threshold more than 0.1.

IV. RESULT

We submitted three runs both for Scenario A and Scenario B in this year. For Scenario A, TABLE II and TABLE III show the performance of our system. For Scenario B, TABLE IV reports the performance of our system.

TABLE II: Median scores of mobile assessment for Scenario A

P(strict)	P(lenient)	U(strict)	U(lenient)
0.3403	0.4174	-805	-456

TABLE III: Median scores of NIST assessment for Scenario A

EG-P	EG-1	nCG-P	nCG-1
0.2194	0.1951	0.2095	0.1826

TABLE IV: Median scores of NIST assessment for Scenario B

nDCG@10-p	nDCG@10-1
0.2194	0.1865

V. CONCLUSION AND FUTURE WORK

This paper demonstrates the implementation details and official evaluation results of our system. The evaluation results showed that there was still a long way to go in practice. Nonetheless, some progress has been made. Many further experiments are needed in the future, and the works emphasis should be on how to improve the accuracy and efficiency.

REFERENCES

[1] Suwaileh R, Hasanain M, Elsayed T. Light-weight, Conservative, yet Effective: Scalable Real-time Tweet Summarization[C]//TREC. 2016.
[2] Lee K, Qadir A, Datla V V, et al. Assorted Textual Features and Dynamic Push Strategies for Real-time Tweet Notification[C]//TREC. 2016.
[3] Moulahi B, Jabeur L B, Chellal A, et al. IRIT at TREC Real Time Summarization 2016[J].
[4] Li S, Hao Z, Han Z, et al. HLJIT at TREC 2016: The Approaches Based on Document Language Model for Real-Time Summarization Track[C]//TREC. 2016.
[5] Li H T D L W. PolyU at TREC 2016 Real-Time Summarization[J].
[6] Bei C, Hu P. CCNU at TREC 2016 Real-Time Summarization Track[C]//TREC. 2016.
[7] Wang K, Yang Z. BJUT at TREC 2016: Real-Time Summarization Track[C]//TREC. 2016.
[8] Tao K, Abel F, Hauff C, et al. What makes a tweet relevant for a topic?[C]// MSM. 2012: 49-56.
[9] Lin J, Roegiest A, Tan L, et al. Overview of the TREC 2016 real-time summarization track[C]//Proceedings of the 25th Text REtrieval Conference, TREC. 2016, 16.