PKUICST at TREC 2017 Real-Time Summarization Track: Push Notifications and Email Digest

Jizhi Tang Chao Lv Lili Yao Dongyan Zhao*
{tangjizhi, lvchao, yaolili, zhaody}@pku.edu.cn

Institute of Computer Science and Technology
Peking University, Beijing 100871, China

Abstract

In this paper, we describe our approaches and corresponding results in the Real-Time Summarization(RTS) track at the 2017 Text Retrieval Conference(TREC). The main idea is to build a two-stage filter system for both scenario A and B. In the first stage, tweets are filtered according to its relevance score to a particular topic, while in the second stage, they are filtered according to its novelty score to previous submitted tweets. We tried several approaches to model the text similarity, such as negative KL-divergence and cosine distance, as well as blending models. Especially, in scenario A, the push notification scenario, we designed a decoupled system that can maintain high availability in order to meet the real-time requirements. The experiment results show that our methods reach good performance with respect to several metrics such as EG-p and nDCG-p.

Introduction

Microblog, such as Twitter and Weibo, has become one of the most important accesses for people to get information. However, finding out helpful information from massive microblogs by hand can be very difficult and exhausting. Building an automatic system that helps to pick out specific microblog is a good solution. The TREC 2017 Real-Time Summarization (RTS) Track aims to explore techniques that helps build such systems. There are two scenarios contained in the RTS Track:

- **Scenario A (push notification):** Content that is identified as relevant and novel by a system based on the user's interest profile should be sent to the user in a timely fashion.
- **Scenario B (email digest):** Participating systems should identify tweets and aggregate them into an email digest. The email should be periodically sent to a user. Under that circumstances, users can read a longer story about the contents.

For both scenario A and B, we perform a two-stage filter system separately.

In push notification scenario, our system contains three function modules, Filter Module, Judge Module and Submit Module, and two tables, i.e. Pre-process Table and Submit Table, that transfer data between modules. The Filter Module listens to the tweet sample stream, roughly filter out tweets that obviously irrelevant to the interest profiles, and insert the remain tweets into the Pre-process Table. The Judge Module continuously detects new tweets from the Pre-process Table, and compute the relevance score between those tweets and the interest profiles. A tuned relevance threshold α is utilized to judge whether a specific tweet and an interest profile are relevant. Then, for every relevant profile, we compute the novelty score between current tweet and previous submitted tweets. Similarly, a novelty threshold β is used to determine whether a tweet is novel. Those tweets passed the two threshold will be inserted to the Submit Table. Finally, the Submit Module submit tweets in the Submit Table to the Evaluation Broker. The independence of the three module guarantees that the system can recover quickly and safely from system crash.

In email digest scenario, we directly precess the tweets from Pre-process Table. The whole procedure is basically the same as Judge Model in scenario A. Two threshold are performed to filter out those 'relevant but novel' tweets. the only difference is that after the filtering, we sort the tweets by the relevance score for every interest profile, and select the top 100 tweets per interest profile per day.

Method

In this section, we discuss our system design thoroughly. Fig.1 shows the architecture of our system. The Pre-process Module and the Pre-process Table are actually shared by both scenario A and B, we take these together as an independent part, Preliminaries. And then Scenario A and B will be discussed separately.

Preliminaries

In this section, we mainly talk about how our system perform preliminary operation, to make the follow-up filtering easier and more precise.

Filter Module The preprocessing we adopt on interest profile and tweet stream follows (Yao et al. 2016) and (Lv, Yang, and Zhao), which is described as follows:

- **Non-English Filtering:** Tweets written in a language other than English would be judged as not relevant based on guidelines of Real-Time Summarization Track. Thus,

*Corresponding author.

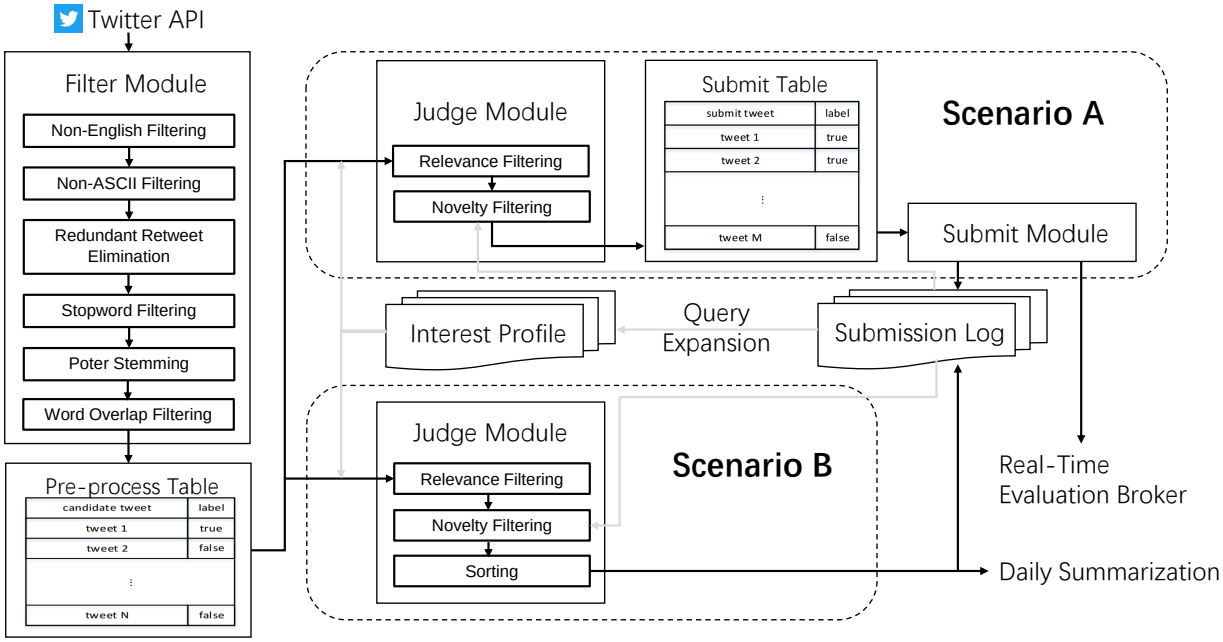

Figure 1: The System Architecture.

we use the twitter's language detector to abandon the non-English tweets.

- **Non-ASCII Words:** Removing all NON-ASCII characters from the tweets will also helps remove non-English tweets.
- **Redundant Retweet Elimination:** All additional commentary in the tweets containing "RT @" will be ignored. As the guideline mentioned, all retweets should be normalized to the underlying tweets.
- **Porter Stemming and Stopword Filtering:** We remove all stopwords and stem the tweet text using the Natural Language Toolkit.
- **Word Overlap Filtering:** We filter out tweets that have no overlap with all interest profile in word-level, because our methods don't consider semantic information. This can accelerate the he speed of identifying possible relevant tweets for each profile.

Statistics Information In the language model, if any word in the query is not in the document, the relevant score between them will equal to zero, which is unreasonable. Smooth techniques could solve this problem by merging global word probability distribution with current document model. In our proposed approach, we obtain the global word probability distribution by computing word count information of tweet stream during the 10 days before the evaluation period.

Scenario A

In this section, we will introduce our method for Scenario A.

- **Judge Module** This module keeps 'listening' to the Pre-process Table. Every time a new inserted tweet is detected (actually we collect a bunch of tweets in every time interval T1 in practice), that tweet is sent to the next stage for filtering. In the first stage of filtering, for every interest profile, we compute a relevance score using a text similarity function f. If that relevance score is bigger than the relevance threshold α, this tweet-profile pair can go to the next stage. In the second stage of filtering, for every selected tweet-profile pair, The same as previous stage, we compute the similarities using f between the tweet and each of the pushed tweets of this profile, we select the biggest one as the novelty score and compared to the novelty threshold β. If that novelty score is smaller than β, we insert this tweet to the Submit Table, and remove it from the Pre-process Table.
- **Submit Module** Like the Judge Module, this module keeps 'listening' to the Submit Table. Every time a new inserted tweet is detected (actually we collect a bunch of tweets in every time interval T2 in practice), that tweet is sent to the Evaluation Broker. If the response tell us the submission is accepted successfully, we remove that tweet from the Submit Table and write it into the Submission Log. Otherwise that tweet will stay in the Submit

Table until a successful submission.

Similarity Algorithm The key components of the Judge Module is the similarity function f. We use f to compute the similarity of (tweet, profile) pairs and (tweet, tweet) pairs. Note that we only use the 'title' of each interest profile for computing similarities.

We utilize two different methods to model similarity.

- **negative KL-divergence language model** One of the most powerful approach is language model. Each tweet D and each interest profile Q can be regard as a probability distribution. We use notation $\widehat{\theta}_D$ and $\widehat{\theta}_Q$ to represent the language model respectively. The negative KL-divergence between $\widehat{\theta}_Q$ and $\widehat{\theta}_D$ with the help of collection language model $\widehat{\theta}_C$ can be calculated as below:

$$\begin{aligned} DIR(Q,D,C) = &\sum_{w \in Q} P(w|\widehat{\theta}_Q) \cdot \\ &log\left((1-\lambda) * P(w|\widehat{\theta}_D) + \lambda * P(w|\widehat{\theta}_C)\right), \\ with\ \lambda = &\frac{\mu}{|D|+\mu} \end{aligned} \tag{1}$$

- **cosine distance** Another method is using cosine distance model the similarity directly. We build the IDF(Inverse Document Frequency) vector for each tweet and interest profile. The IDF of each word is computed by the tweets collected before evaluation period, as we have talked in the Preliminaries section. The formula is shown below:

$$COS(Q,D) = \frac{\vec{Q} \cdot \vec{D}}{|\vec{Q}||\vec{D}|} \tag{2}$$

Query Expansion As microblog retrieval suffers severely from the vocabulary mismatch problem (i.e. term overlap between query and tweet is relatively small). Query Expansion (Zhai and Lafferty 2001) can play an important role in this situation. Moreover, more terms in the query makes retrieval more precise in general.

We use Pseudo Relevance Feedback technique to expand the interest profile with IDF-cosine method in PKUICSTRunA3 and PKUICSTRunB3. At the end of each day, we collect the submitted tweets of each interest profile, compute the word count, and sort by the word count. For each interest profile, we add the top K words that are not original in the profile to the profile. We set the expanded words a weight $\gamma(\gamma < 1)$ while the original words with a default weight 1.

Parameter Selection The parameters shown in Table.1 are tuned via grid search on TREC 2016 dataset.

Scenario B

From Figure.1, we can see our approach of Scenario B is not much different from that of Scenario A. We remove the Submit Module and Submit Table, and add a sorting process at the end of Judge Module. Besides, we try to utilize model blending in this scenario. Model blending has been proved useful in many situation. We simply use the formula below to blend our two similarity models discussed before.

$$\begin{aligned} BL(Q,D,C) = &\delta \cdot DIR(Q,D,C) + \\ &(1-\delta) \cdot COS(Q,D), 0 < \delta < 1 \end{aligned} \tag{3}$$

Parameter Selection Like Scenario A, these parameters shown in Table.2 are tuned via grid search on TREC 2016 dataset.

Table 1: Parameters of the Push Notifications Scenario.

Run ID	f	α	β	Query Expansion
PKUICSTRunA1	DIR $\mu = 50$	0.7	0.5	-
PKUICSTRunA2	COS	0.8	0.85	-
PKUICSTRunA3	COS	0.8	0.85	$K = 2$ $\gamma = 0.2$

Table 2: Parameters of the Push Notifications Scenario.

Run ID	f	α	β	Query Expansion
PKUICSTRunB1	DIR $\mu = 50$	0.72	0.72	-
PKUICSTRunB2	BL $\mu = 50$ $\delta = 0.5$	0.78	0.78	-
PKUICSTRunB3	BL $\mu = 50$ $\delta = 0.5$	0.78	0.78	$K = 2$ $\gamma = 0.2$

Experiment

The evaluation of TREC 2017 Real-time Summarization track takes place from July 25, 2017 UTC to August 3, 2017 UTC.

Scenario A

For Scenario A, there are two main assessment approaches.

One is Mobile Assessment. Mobile assessors receive pushed tweets immediately, and judge whether that tweet is relevant, redundant or non-relevant. The strict precision is the proportion of relevant tweets pushed by a run, and the lenient precision is the proportion of relevant and redundant tweets pushed by a run. This time, assessors judged 188 topics(with uneven effort), and the results of Mobile Assessment are shown in Table.3.

Another assessment approach is NIST Assessment. This assessment has multiple metrics, including Expected Gain (EG) and Normalized Cumulative Gain (nCG). There two variants of EG, EG-1 and EG-p. In EG-1 metrics, on a silent day, the system receives a score of one (i.e., perfect score) if it does not push any tweets. While in EG-p, on a silent day, the score is one minus the fraction of the ten-tweet daily quota that is used. Similarly, we can define nCG-1 and nCG-p. The results of NIST Assessment are shown in Table.4

We can observe that different approaches of modelling similarity have different preferences. In PKUICSTRunA2 and PKUICSTRunA3, we select IDF-cosine algorithm and a relative high relevance threshold. That makes PKUICSTRunA2 and PKUICSTRunA3 get higher score in precision. But higher threshold also means the system tends not to push the tweets, witch makes these systems perform bad in some macro-averaged metrics (because it's common for these systems to push nothing all day), like EG and nCG. Additionally, from the comparison of PKUICSTRunA2 and PKUICSTRunA3, we learn that a good query expansion method really helps a lot, and pseudo relevance feedback technique can be a choice.

Table 3: Scenario A Mobile Assessment.

Run ID	strict precision	lenient precision
PKUICSTRunA1	0.3108	0.3642
PKUICSTRunA2	0.3673	0.4174
PKUICSTRunA3	**0.3863**	**0.4340**

Table 4: Scenario A NIST Assessment.

Run ID	EGp	EG1	nCGp	nCG1
PKUICSTRunA1	**0.2869**	**0.2588**	**0.2864**	**0.2583**
PKUICSTRunA2	0.1959	0.1866	0.1866	0.1774
PKUICSTRunA3	0.1997	0.1892	0.1908	0.1804

Scenario B

Table.5 reports our results for the email digest scenario. The main evaluation metric is nDCG-p and nDCG-1.

As we keep our parameter setting in Senario B similar to that in Senario A, the performance of our system in Scenario B have the same trend with that in Scenario A. It shows that our two-stage filtering approach also perform well in a non-real-time task.

Table 5: Scenario B NIST Assessment.

Run ID	nDCGp	nDCG1
PKUICSTRunB1	**0.3483**	**0.3003**
PKUICSTRunB2	0.1968	0.1809
PKUICSTRunB3	0.2306	0.2024

Conclusion

In this paper, we present our systems for TREC 2017 Real-Time Summarization Track. We design a two-stage filtering system for both Scenario A and B, and utilize two kinds of text similarity model, language model with negative KL-divergence and IDF with cosine distance. Pseudo relevance feedback technique is tried to do query expansion. Particularly, a decoupled real-time system are designed for Scenario A. Experimental results show our effectiveness and efficiency of our system in both tasks.

Acknowledgments

The work reported in this paper is supported by the National Natural Science Foundation of China Grant 61370116.

References

Lv, F. F. Y. F. C.; Yang, L. Y. J.; and Zhao, D. Pkuicst at trec 2015 microblog track: Query-biased adaptive filtering in real-time microblog stream.

Yao, L.; Lv, C.; Fan, F.; Yang, J.; and Zhao, D. 2016. Pkuicst at trec 2016 real-time summarization track: Push notifications and email digest. In *TREC*.

Zhai, C., and Lafferty, J. D. 2001. Model-based feedback in the language modeling approach to information retrieval. In *CIKM*, 403–410.