IRIT at TREC Real-Time Summarization 2018

Abdelhamid Chellal and Mohand Boughanem

{abdelhamid.chellal, mohand.boughanem}@irit.fr,
Institut de Recherche en Informatique de Toulouse (UMR 5505) University of Toulouse II,
118 route de Narbonne F-31062 Toulouse cedex 9 France

Abstract. This paper presents the participation of the IRIT laboratory (University of Toulouse) to the Real-Time Summarization track of TREC RTS 2018. This track is consisting of two scenarios (A: push notification and B: Email digest) which tackle the challenge of fulfilling the prospective and the retrospection information needs repressively. We submitted three runs for both scenarios A and B. For scenario A, we propose to use a supervised learning approach to build a binary classifier that predicts the relevance of an incoming tweet with respect to the topic of interest. The proposed approach leverages social signals as well as query dependent features to enhance the detection of relevant tweets. Additionally, we investigate the impact of the use of live relevance feedback to re-train the classier each time new relevance assessments are made available. For scenario B, the daily digest is generated by iteratively selecting the top tweets that pass the relevance filter with discarding the redundant ones.

1 Introduction

User-generated content on social media, such as Twitter, provides in many cases, the latest news before traditional media, which allows having a retrospective summary of events and being updated in a timely fashion whenever a new development occurs. However, social media, while being a valuable source of information, can be also overwhelming given the volume and the velocity of published information. To shield users from being overwhelmed by irrelevant and redundant posts, retrospective summarization and prospective notification (real-time summarization) were introduced as two complementary tasks of information seeking on document streams.

TREC Real-Time Summarization (RTS) track focus on the aforementioned types of information needs. In this track, participant systems are required to monitor the live stream provided by Twitter streaming API over a period of twelve days (from Monday July 23, 2018 00:00:00 UTC to Friday August 3, 2018 23:59:59 UTC) and to identify relevant tweets per day with respect to predefined user interest. This track is composed of two scenarios namely scenario A (push notification) and scenario B (Email digest). In the former, systems monitor the live posts stream and push relevant and novel notifications as soon as possible in order to update the user whenever a novel information occurs. The latter aims to generate a daily summary after the day ends that consists of a batch of up to 100 ranked tweets that capture ”what happened” regarding each topic

As previous edition (TREC RTS 2017 track), participant systems can fetch mobile assessor relevance judgments in real time for its pushed tweets. The availability of relevance feedback provides opportunities for techniques based on adaptive learning and relevance feedback.

To tackle scenario A, the core task is to determine whether a tweet is relevant or not. The majority of existing approaches rely on the threshold-based filter in which the decision to select or discard an incoming tweet depends on whether its relevance score falls above a predefined threshold. However, it was shown that the accuracy of the tweet filtering relies on identifying an appropriate threshold for pushing updates [1, 2]. The relevance threshold value has a serious impact on the filtering effectiveness. To overcome the issue of setting the relevance threshold value, we propose to use a supervised learning approach to build a binary classifier that predicts the relevance of an incoming tweet with respect to the topic of interest. To identify relevant tweet to push to the user, we consider the relevance filtering as binary classification problem in which the incoming tweet is classified as relevant or not relevant.

For scenario B, the daily digest is generated by selecting the top weighted tweets iteratively and with discarding those having their similarity with respect to the current summary above a certain threshold.

2 Learn to filter strategy for prospective notification

2.1 System overview

Knowing that to be effective a system needs to optimize three constraints: the relevance with respect to the topic of interest, the novelty/redundancy (avoid pushing multiple tweets that convey the same information) and the latency between the publication time and the notification time of selected tweets (provide updates as soon as the event occurs). To fulfill these requirements, our approach consists of three filters adjusted sequentially namely:

– Pre-processing and low-quality tweet filtering;
– Relevance filter based on a binary classifier that takes advantage of the ongoing user relevance feedback;
– Novelty filter.

In these filters, the decision to select/ignore the incoming tweet is made as soon as the tweet is collected. In order to reduce the latency between notification time and publication time, the tweet that passes these filters is pushed immediately without delay.

2.1.1 Pre-processing and low-quality tweet filtering

The pre-processing step consists of stop-words removal, stemming and tokenize the tweet using Twokenize tool [1]. Then, we apply a simple quality and topical filters to discard potential trash and irrelevant tweets and yields to boost the efficiency of our approach to handle the velocity of the tweet stream. The quality filter excludes any tweet that does meet at least one of the following rules:

– **Non-English Filtering:** We rely on Twitter's language detector to discard the non-English tweets. In addition, tweets that contain more than 35% of non-English characters are also filtered out.
– **BadWrods Filtering:** Tweets including swear or bad words are filtered out since we assume that it would be inappropriate to push notification containing such kind of vocabulary.
– **Retweet de-duplication:** Since the practice of "retweeting" the same content is very common on Twitter, we de-duplicated tweets using retweet mechanism and tweet identifier (tweet id). If the incoming tweet is a re-tweet of an already seen tweet in the stream then this new tweet is eliminated.
– **Trash filtering:** Any tweet that meets one of the following conditions is considered as trash and hence it is filtered out:
  • It contains less than five unique tokens;
  • It includes more than one URL ;
  • It mentions more than two usernames ;
  • It contains more than three hashtags ;

The topical filter step is a word overlap filter that drops all incoming tweets that do not contain a predefined number of query words. The incoming tweet T is considered as a candidate tweet if its number of overlapping words with the query title is higher than the minimum of either a predefined constant (K) or the number of words in the query title $min(K, |Q^t|)$. In our runs, the value of (K) was set to 2.

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

2.1.2 Relevance filtering

The main task in the prospective notification is the identification of relevant posts, in a timely fashion, in the social media stream. To shield users from unwanted notifications, systems attempt to find a trade-off between pushing too many or too few tweets. To achieve this purpose, it is common to rely on a threshold-based filter. In the case of a high threshold value, a system may miss pushing interesting content to the user. Conversely, in the case of a low threshold value, a system may overwhelm the user with irrelevant tweets. However, it is difficult to properly and effectively set the relevance threshold value [1, 2].

To overcome the issue of relevance threshold setting, we propose a learn to filter approach based on machine learning. Instead to rely on a threshold-based filter, we use a binary classifier that predicts the relevance of an incoming tweet with respect to the topic of interest. The proposed approach leverages social signals as well as query dependent features to enhance the detection of relevant tweets. We propose a set of social and other non-content features suitable for real-time tweet filtering. Furthermore, we explore and evaluate an adaptive learning strategy in which the live user relevance feedback is used to update periodically the relevance classifier. This allows investigating the gain that can be achieved by taking advantage of an ongoing relevance feedback which is generated by users as tweets are pushed.

We use TREC real-time tweet filtering and summarization dataset [3](TREC RTF 2015) to train the binary classifier as follows: we extract for each topic tweets from the judgment pool of TREC 2015 RTF dataset. We obtain 94,068 tweets among them 8,164 tweets were labeled as relevant. We notice that the classes of these sets are unbalanced. To get a balanced training dataset, we filter out all tweets that do not contain at least two query's words. Thus, we obtain a training dataset that contains 6,663 tweets in which the distribution of relevant and irrelevant tweet is 50.18% and 49.81% respectively.

In the context of real-time tweet filtering, we are limited to use features that are already available in the meta-data of a tweet. This allows us to predict the relevance of the incoming tweet as soon as it is published. Hence, we are not able to use Twitter's REST APIs to collect further features such as the profiles of followers or to crawl external URL webpage text. This is due to restrictions imposes by Twitter (e.g., limiting the number of calls to a time slot) on using REST APIs, which makes these features inapplicable in real-time filtering scenario. Among the set of available features extracted from the tweet's meta-data, we used feature selection algorithms to determine the best relevance-dependent signals that can be effectively used in the tweet filtering task. We use an information gain algorithm implemented in Weka tool [4], which allows us to identify 22 features categorized into three classes: query dependent, tweet specific and user account features. The query dependent features measure the relevance with respect to the query and social signals features (tweet specific and user account features) are query independent.

Query dependent features: To capture the relevance of a tweet's content, we used six query dependent features that measure the relevance of the given tweet text with respect to a topic. These features are as follows:

- $|(Q^t \cup Q^d) \cap Hashtag|$: The number of words overlaps between the query terms and hashtags in the tweet. The rationale behind this feature is that the presence of a query term as a hashtag is a valuable signal of relevance since hashtags are used to draw attention and to label the content of a given tweet;
- The cosine similarity between the query title and the tweet's text vectors using a word embedding model (word2vec [5]). The vectors of the title of the query and the tweet's text are obtained by summing up all vectors of their words. This feature can be considered as a semantic-based relevance score which aims to leverage the probable semantic relationship between terms of the query and the tweet by taking advantage of a word embedding model;
- $|(Q^t \cap T|)$ and $|(Q^d \cap T|)$: The number of words that overlap between the text of the tweet and the query's title $|Q^t|$ and the query's description $|Q^d|$ respectively;
- $RSV(T, Q^t)$ and $RSV(T, Q^d)$: The relevance score of the incoming tweet with respect to the title Q^t and the description Q^d respectively.

The relevance scores $RSV(T,Q^t)$ and $RSV(T,Q^d)$ are evaluated using an adaptation of Extended Boolean Model proposed by [6], in which the word embedding is used to estimate the weight of query terms in order to cope with the shortness of tweets and word mismatch issues.

Tweet specific features: These features describe elements that are mentioned in a tweet text and the nature of the tweet itself which can be a retweet of another tweet or a reply to an old tweet of another user. We leverage seven (7) tweet-specific features that are defined as follows: We exploit five tweet-specific features: (1) URL& Hashtag: Whether the tweet contains a URL or a hashtag; (2) Retweet count per day: The ratio of times this tweet has been retweeted and its age (in days); (3) the number of words that a tweet text contains; (4) the hour at which the tweet was published; (5) whether an entity (PERSON, ORGANIZATION, LOCATION) is mentioned in the tweet. For this, we use Stanford Named Entity Recognizer [2].

User account features: We argue that the importance of tweet content is related to the authority of the user who posts the tweet. The authority of a user can be captured through social features that are available in the meta-data of tweets. Notice here that in the case that the given tweet is a retweet, we consider the features of the user that published the original tweet, and not the one who retweeted it. These features are time-sensitive, the importance of a signal depends on the account age. An old account may have much more followers than a recent one. Therefore, in user account features, we implicitly consider the age of the account (in days) at the time of the tweet publication. The user account features are as follows:

- Follower: Number of followers of the author of the tweet;
- Friend: Number of followees of the author of the tweet;
- Verified: Whether the user account is verified;
- Tweet/day: Ratio of the number of posted tweets and the age of the account;
- List/day: Ratio of the number of lists a user appears in and the age of the account;
- Fol/day: Ratio of the number of user followers and the age of the account;
- Fr/day: Ratio of user friends and the age of the account;
- Fol/Fr: Ratio of the numbers of followers and followees of the user;
- $(List + Fol/Fr)/day$: A combination of Fol/Fr and the number of lists the user appears in.

2.1.3 Novelty filtering

The novelty detection is based upon an adaptation of word overlap similarity in which the incoming tweet is compared to all tweets previously pushed to the user. To do so, the tweets of the summary are aggregated into a summary set of terms SW and the incoming tweet is compared to this summary set. The novelty score of the incoming tweet $T = \{t_1, ..., t_n\}$ is computed using word overlap as follows:

$$NS(T,SW) = 1 - \frac{|SW \cap T|}{|T|} \tag{1}$$

With :

$$SW = \bigcup_{j=1}^{M} \{t_1^j, t_2^j, ..., t_n^j\} \tag{2}$$

Where M is the number of tweets already selected in the summary and t_i^j is the i-th term in tweet j in the summary.

In our runs, the incoming tweet is pushed only if its novelty score is greater than a predefined threshold. We set the novelty threshold value to 0.5 for all topics and overall the evaluation period based on pilot experiments carried out on TREC RTS 2017 dataset.

[2] http://nlp.stanford.edu/software/CRF-NER.shtml

2.1.4 Adaptive learning strategy

To further improve the effectiveness of the binary classifier, we use relevance information feedback to update the binary classifier. The system takes advantage of relevance feedback to re-train the classifier. To do so, the classifier is initialized with a training dataset and it is retrained periodically each times new relevance judgments have been made available. The system fetches the relevance judgment of users periodically (every 10 minutes: rate fixed by track organizer) and uses it to label the new instances that correspond to the features of the pushed tweets. These new labeled instances are added to the current training instances set and the model is retrained. We set this strategy in order to fit a real-world scenario in which the user may choose to judge the pushed tweet immediately or later (if it arrives at an inopportune time) or may choose to not do it.

2.2 Runs

We submitted three runs based upon a binary classifier (IRIT-Run1, IRIT-Run2 and IRIT-Run3) in which the same novelty threshold value and a minimum word overlap between the interest profile and the tweet text were used. The main difference between these runs is the classification algorithm and whether the classifier is adaptive (the live assessment feedback were used to re-train the classifier) or not. Table 1 describes the configuration of our three runs.

Run	Classifier	Adaptive
IRIT-Run1	XGboost[7]	Yes
IRIT-Run2	XGboost[7]	No
IRIT-Run3	Random Forest [8]	Yes

Table 1. Configuration of our different runs.

2.3 Results for scenario A in terms of mobile live assessment

In scenario A, tweets submitted by participating systems were immediately routed to the mobile phone of an assessor with the corresponding interest profile. Judgments happened online as systems submitted tweets. The assessor may choose to judge the tweet immediately or later (if it arrives at an inopportune time) or may choose to not do it. Note that in this evaluation, a tweet might be judged by more than one mobile assessor. In this year, among 298 topics, assessors judged 135 topics.

Table 2 reports the performance of our runs based upon learning to filter approach in terms of the number of tweets that were judged relevant (Rel), redundant (Red), and not relevant (Not_Rel); the number of unjudged posts; the total length of each run ($|R|$), the coverage of judged tweets the mean latency and median latency. The four last columns report the strict and lenient precision (P_S, P_L) and the strict and lenient utility (U_S, U_L).

	Rel	Red	Not_Rel	unjudged	$\|R\|$	coverage	Mean_latency	Median_latency	S_P	L_P	S_U	L_U
IRIT-Run3	3115	84	2385	62	1836	0.966	328.8	34.0	**0.5578**	**0.5729**	**646**	**814**
IRIT-Run1	3226	93	3112	90	2182	0.959	316.6	33.0	0.5016	0.5161	21	207
IRIT-Run2	3507	71	4337	61	2579	0.976	270.0	32.0	0.4431	0.4521	-901	-759
Baseline	–	–	–	–	–	–	–	0.4791	0.4829	-27.5	-12	

Table 2. Performances of our submitted runs for scenario A evaluated by the mobile assessors.

From Table 2, first, we observe that the performance of runs IRIT-Run1 and IRIT-Run3 based on adaptive learning strategy outperform the baseline overall evaluation metric. The best performances

	nDCGp@10	**nDCG1@10**
IRIT-RunB3	**0.8449**	0.7411
IRIT-RunB2	0.8294	0.6613
IRIT-RunB1	0.8287	0.6614
Baseline	0.8287	0.7531

Table 3. Performances of our submitted runs for scenario B.

were achieved by run IRIT-Run3 in which the binary classifier is based on Random Forest algorithm. The comparison between the performance of IRIT-Run1 that use the ongoing relevance feedback to retrain the binary classifier and the performance of IRIT-Run2 which is not adaptive reveals that the adaptive learning strategy outperforms the passive learning strategy overall metrics. These results show that taking advantage of the ongoing relevance feedback allows improving the ability to identify relevant tweets.

Interestingly, we note that the best performing run is the one that pushed the smallest numbers of tweets. The number of tweets pushed by run IRIT-Run3 is 40.46% less than the number of tweets pushed by run-IRIT2. This result reveals that the adaptive learning strategy based on Random Forest algorithm managed to achieve a good balance between pushing too many or too few tweets. This result may suggest that the learn to filter model that takes advantage of an ongoing assessment feedback is able to detect silent days and to keep silence in such days.

For the timeliness, we notice that the mean latency of our approach is less than 1 minutes (32-34 seconds). This result can partially be explained by the fact that the decision to select/ignore an incoming tweet is made in real-time as soon as a tweet is available. However, notice that latency is computed with respect to the first tweet in each cluster, and thus a system may have a high latency even if it submits a tweet immediately after it is identified. This allows to concludes that our approaches trade off summary quality with latency and hence produces good quality output at the cost of low latency.

3 Runs for Scenario B

To generate runs for scenario B, the tweets stream is filtered using the low-quality tweet and relevance filters with ignoring the limit of ten tweets per day. At the end of each day, we obtained a set of candidates tweets for each topic. From this set of tweets, the top ten tweets are iteratively selected with the exclusion of those having word overlap above a static redundancy threshold set to 0.6. Candidate tweets are ordered according to their the relevance score computed using an adaptation of Extended Boolean Model proposed by [6].

We submit three runs which were generated as follows:

1. The first run (IRIT-RunB1) takes as input tweets that passe the relevance filter based on adaptive learning strategy used in run IRIT-Run1 of scenario A. Notice here that the binary classifier used in IRIT-Run1 is based on XGboost [7] algorithm and the relevance feedback of mobile assessor was used to retrain the binary classifier periodically.
2. The second run (IRIT-RunB2) is almost the same as the first run except that we use as input a set of tweets that were filtered without updating the binary relevance;
3. The third run (IRIT-RunB3) takes as input tweets that passe the relevance filter based on adaptive learning strategy used in run IRIT-Run3 in which the binary classifier is based on Random Forest [8].

3.1 Results for scenario B

Table 3 reports our results for Scenario B, in terms of in terms of two variants of nDCG metric namely nDCGp and nDCG1. We recall here that the difference between these variants lies in the way in which systems are penalized for pushing tweets on a silent day. On such day, nDCG1 variant is binary whereas nDCGp is based on a linear penalty. In the nDCG1 metric, a system receives a perfect score (1) if it does not push any tweet on a silent day, or zero otherwise, whereas in the nDCGp metric the penalty is gradually increased from 0 to 1 according to the number of pushed tweets.

First, we observe that the best performance is achieved by run IRIT-RunB3 that takes as input tweets filtered by the adaptive binary classifier based on Random Forest. This result confirms those obtained in scenario A.

As shown in Table 3, our best performing run (IRIT-RunB3) overpass the baseline for nDCGp with an improvement of 1.95% but it failed to beat the baseline in terms of nDCGp metrics. This result underlines the fact that our run pushed more tweets in silent days than the baseline.

4 Conclusion and future work

We presented in this paper an adaptive learning to filter strategy to tackle the challenge of for real-time tweet summerization. In the proposed approach, we consider the tweet filtering task as a binary classification problem. We use supervised learning approach to build a binary classifier that predicts the relevance of an incoming tweet with respect to the topic of interest. The proposed method allows countering the threshold setting issue and taking advantage of an ongoing assessment feedback. The binary classifier leverages social signals as well as query dependent features to enhance the detection of relevant tweets. For this purpose, we suggest a set of social and other non-content features suitable for real-time tweet filtering.

We believe that results are quite promising and could give interesting insights in the future regarding the challenge of real-time tweet filtering and summarization, which are important components in the information access within data-streams. The learning based filter achieves a good performance overall metrics with low cost of latency. Results also revealed that more improvements are achieved by taking advantage of ongoing relevance feedback.

References

1. Charles L. A. Clarke Jimmy Lin Luchen Tan, Adam Roegiest. Simple dynamic emission strategies for microblog filtering. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, 2016.
2. Abdelhamid Chellal, Mohand Boughanem, and Bernard Dousset. Multi-criterion real time tweet summarization based upon adaptive threshold. In *2016 IEEE/WIC/ACM International Conferences on Web Intelligence (WI16, Omaha, Nebraska USA, October 13-16, 2016*, pages 264–271, 2016.
3. Jimmy Lin, Miles Efron, Yulu Wang, Garrick Sherman, Richard McCreadie, and Tetsuya Sakai. Overview of the trec 2015 microblog track. In *Text REtrieval Conference, TREC, Gaithersburg, USA, November 17-20*, 2015.
4. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.
5. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
6. Abdelhamid Chellal, Mohand Boughanem, and Bernard Dousset. Word similarity based model for tweet stream prospective notification. In *Advances in Information Retrieval - 39th European Conference on IR Research, ECIR 2017, Aberdeen, UK, April 8-13, 2017, Proceedings*, pages 655–661, 2017.
7. Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.
8. Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.