IRIT at TREC Real-Time Summarization 2017

Abdelhamid Chellal and Mohand Boughanem

{ abdelhamid.chellal, mohand.boughanem}@irit.fr,
Université de Toulouse UPS-IRIT,
118 route de Narbonne F- 31062 Toulouse cedex 9

Abstract. This paper presents the participation of the IRIT laboratory (University of Toulouse) to the Real-Time Summarization track of TREC RTS 2017. This track aims at exploring prospective information needs over document streams containing novel and evolving information and it consists of two scenarios (A: push notification and B: Email digest). In this year the live mobile assessment was made available in real-time which provides the opportunity to propose an approach based on adaptive learning that leverages relevance feedback. We submitted two runs for scenarios A and three runs for scenarios B. In both scenarios, the identification of relevant tweets is based on a binary classifier that predicts the relevance of an incoming tweet with respect to an interest profile. We examine the impact of the use of the live relevance feedback to re-train the classier each time new relevance assessments are made available. For scenario B, the summary generation is modeled as an optimization problem using Integer Linear Programming.

1 Introduction

Social media streams are valuable sources of information for a user who wishes to receive timely notification to keep up-to-date on the topic of interest. To fulfill this information need, notifications should be relevant (on topic), timely (provide updates as soon as the event occurs), and novel (avoid pushing multiple tweets that convey the same information). TREC Real-Time Summarization (RTS) track focus of these 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 eight days and to identify relevant tweets per day with respect to predefined user interest. Tweets identified as relevant to the users interest profile are pushed in two different scenarios:

1. Scenario *A*, called "*Push notifications*": In this scenario, tweet that is identified as relevant and novel is pushed in real-time to TREC RTS evaluation broker. Participating system are allowed to push up to 10 notifications per day per interest profile and pushed tweets are routed immediately to the mobile phones of assessors for relevance judgment.
2. Scenario *B*, called "*Periodic email digest*": This scenario is more like a top-100 retrieval task based on a one-day tweet collection. It consists of identifying a batch of up to 100 ranked tweets per day and per topic to be pushed as an email notification to the user after the day ends.

Comparing TREC RTS 2017 track with previous editions TREC RTF 2015 [1] and TREC RTS 2016 [2], the major change consists in the deployment this year a mechanism whereby 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.

For scenario A, we investigate a use of a supervised learning based approach aiming at identifying relevant tweets in a real-time *fashion*. Our approach consists of three filters that are adjusted sequentially namely quality and topicality, relevance and novelty filters. In our participation, we focus on the relevance filter which we consider as a binary classification task. Hence, we propose to build a binary classifier that determines whether the incoming tweet is relevant or not with respect to the topic of interest. We train the binary classifier using the labeled tweets of the TREC RTF 2015 track data-set.

Our classifier considers social signals as well as query dependent features to filter tweet stream. In order to explore the effect of the use of the live relevance feedback, we submitted two different runs. The main difference between our runs is that in the first is an adaptive learning strategy in which live relevance feedback is used to update periodically the classifier whereas in the second run the classifier is not re-trained.

For scenario B, we investigate the use of Integer Linear Programming (ILP) to generate a daily summary and we compare this approach to the traditional method that consists of selecting top weighted tweets iteratively and discarding those having their similarity with respect to the current summary above a certain threshold.

2 Adaptive learning strategy for push notification

2.1 System overview

In prospective information need, notifications should be relevant (on topic), timely (provide updates as soon as the event occurs), and novel (avoid pushing multiple tweets that convey the same information). The aforementioned requirements are fulfilled by our approach as follows: (i) To reduce the latency between notification time and publication time, the decision of selecting/ignoring an incoming tweet is taken immediately in real-time as soon as tweet occurs. (ii) To enhance the relevance and novelty of pushed tweets, the proposed method relies on three consecutive filters 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.

2.1.1 Pre-processing and low-quality tweet filtering

The pre-processing step first filters out any non-English tweet using language attribute available in the tweet meta-data. After stop-words and URL removal and stemming, the text of the incoming tweet is tokenized with considering hash-tags (without # prefix). Finally, we apply a quality filter that discards any tweets containing less than five tokens or more than one URL or three hashtags. To boost filtering speed, we apply a boolean relevance filter that eliminates any tweet that does not match at least 3 terms of the title and the description of the interest profile. For the matching against tweet text, the title and the description of the interest profile were tokenized and stop-words were removed.

2.1.2 Relevance filtering

The relevance filter is the main component in real-time tweet filtering system. 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. We use a supervised learning approach to build a predictive binary classifier that produces tweet filtering predictions. The binary classifier is built using a Random Forest algorithm [3] that is trained on a TREC 2015 RTF dataset 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 utilize features that are already available in the meta-data of a tweet, and we are not able to use Twitter REST APIs to collect further features such as the profiles of followers. From the set of available features, we used feature selection algorithms to determine the best relevance-dependent signals that can be effectively exploited in a tweet filtering task. We use an information gain algorithm implemented in Weka tool [4], which allows us to identify 21 features categorized into three classes: query dependent, tweet specific and user account features.

Query dependent features: In our approach, we assume that the user information need follows the standard TREC topic format which includes a title (Q^t) of the information need and a description (Q^d) that indicates what is and what is not relevant. We used six query dependent features: (1) the number of words overlap between the query's title $|Q^t|$ and hashtags in the tweet, (2,3) the number of word overlaps between the text of the tweet and the query's title $|Q^t|$, and the query's description $|Q^d|$ respectively, (4) the cosine similarity between the query title and the tweet's text vectors using word2vec [5] word embedding. The vectors of the title of the query and the tweet's text are obtained by summing up all vectors of their words. (5,6) the relevance score of the incoming tweet with respect to the title Q^t and the description of the query Q^d. These scores 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. In this adaptation denoted by Word Similarity Extended Boolean Model (WSEBM), the query title Q^t represents "**AND**ed terms", and Q^d represents "**OR**ed terms". The relevance of the tweet T to "AND query" Q^t and "OR query" Q^d are estimated respectively as follows [6]:

$$RSV(T, Q^t_{and}) = 1 - \sqrt{\frac{\sum_{q^t_i \in Q^t} (1 - W_T(q^t_i))^2}{|Q^t|}} \tag{1}$$

$$RSV(T, Q^d_{or}) = \sqrt{\frac{\sum_{q^d_i \in Q^d} (W_T(q^d_i))^2}{|Q^d|}} \tag{2}$$

where $W_T(q)$ is the weight of the query term q in the tweet T, which is evaluated as follows:

$$W_T(q) = \max_{t_i \in T}[w2vsim(t_i, q)] \tag{3}$$

where $w2vsim(t_i, q)$ is the cosine similarity between vectors of tweet words t_i and query words q.
Tweet specific features: 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 [1].

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. 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 user;
- $(List + Fol/Fr)/day$: A combination of Fol/Fr and the number of lists the user appears in.

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

2.1.3 Novelty filtering

For novelty filtering, a word overlap pairwise similarity is used to detect redundancy of the incoming tweet with respect to tweets previously pushed for a given topic over all days. The incoming tweet is pushed if it novelty score is greater than a predefined threshold. We set the novelty threshold to 0.7 for all topics and over the evaluation period.

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 fetches the relevance judgment of users every 10 minutes (rate fixed by track organizer) and used it to label the new instances that correspond to the features of the pushed tweets. The classifier is re-trained periodically once new relevance assessments are made available. 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 two runs based upon a binary classifier (IRIT-Run2 and IRIT-Run3) in which we use the same novelty threshold and the minimum word overlap between interest profile and tweet. The main difference between these two runs is that in IRIT-Run2 the binary classifier is re-training using live assessment whereas in IRIT-Run3 the classifier is not updated.

2.3 Results for scenario A

The evaluation framework of scenario A (Push notification) follows the interleaved approach proposed by [7] in which runs are evaluated in two different ways namely live mobile user assessment and post hoc batch evaluation. In the former, tweets submitted by participating systems were immediately routed to the mobile phone of an assessor with the corresponding interest profile. 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. As the assessor judges tweets, the results are relayed back to the evaluation broker and recorded. The post hoc batch evaluation relies on two stages: relevance assessment and semantic clustering. Tweets returned by participating systems were judged for relevance by NIST assessors via pooling based on both scenarios (A and B) after the end evaluation period. Tweets were assessed as not-relevant, relevant, or highly relevant. Then relevant tweets that convey similar information are clustered into the same cluster. Metrics for live mobile user assessment and post hoc batch evaluation are detailed in track guidelines[2] and in [7].

2.3.1 Results in terms of mobile live assessment

	Rel	Red	Not_Rel	S_U	L_U	S_P	L_P	unjudged	$\|R\|$	Mean_latency	Median_latency
IRIT-Run2	404	75	551	-222	-72	0.3922	0.465	48	607	115.1	32.0
IRIT-Run3	554	113	812	-371	-145	0.3745	0.4509	62	875	131.2	29.0
Median	–	–	–	-805	-456	0.3403	0.4174	–	–	–	–

Table 1. Performances of our submitted runs for scenario A evaluated by the mobile assessors.

[2] http://trecrts.github.io/TREC2016-RTS-guidelines.html

In the mobile live assessment, assessors judged 188 topics, and some tweets were judged by multiple assessors. As a result, the number of relevant tweets, the number of redundant tweets, and the number of non-relevant tweets may sum up to a value that is greater than the total number of pushed tweets.

Table 3 reports the performance of our two runs based upon learning to filter approach in terms of the number of posts 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 strict and lenient precision (P_S, P_L) and the strict and lenient utility (U_S, U_L). The two last columns report the mean latency and median latency. Form Table 3, first we observe that the performance of our two runs in terms of utility and precision metrics are above the median scores of all participant systems. Second, the comparison of the performance of the two runs (IRIT-Run2 and IRIT-Run3) reveals that the use of relevance assessment feedback improves the performance overall metrics. However, we notice that the number of tweets pushed by our approaches is relatively low. The run IRIT-Run2 that use the live assessment feedback to re-train the classifier returned 607 tweets among them 48 tweets were un-judged which means that the classifier were re-trained by adding only 559 new instances to the initial training data-set. We think that the low number of pushed tweets is due to the fact that the novelty threshold (set to 0.7) and the tweet-query word overlap threshold (set to 3) were a little bit too tight.

2.3.2 Results in terms of post hoc batch evaluation

	EGp	EG-1	nCGp	nCG-1	GMP.33	GMP.5	GMP.66	mean latency	median latency	total_length
IRIT-Run2	0.2212	0.2041	0.1996	0.1825	-0.0942	-0.0557	-0.0195	96894.1	14768.0	201
IRIT-Run3	0.2194	0.1895	0.2015	0.1716	-0.1853	-0.1221	-0.0626	98864.9	15623.0	320
Median	0.2194	0.1951	0.2095	0.1826	–	–	–	–	–	–

Table 2. Performances of our submitted runs for scenario A evaluated by using post hoc batch evaluation.

In the post hoc batch evaluation, NIST assessors judged 96 topics. Table 4 reports the performance of our two runs in terms of EG-p, EG-1, nCG-p, nCG-1 metrics as well as the median scores of the aforementioned metrics as reference. We also report the performance of our runs in terms of GMP metrics (with alpha = .33, .50, and .66) and the latency (mean and median) of tweets that contributed to system gain.

We note that the performance of our best run (IRIT-Run2) in terms of expected gain metrics (EGp and EG-1) overpasses the median scores overall participant systems whereas performances in terms of normalized cumulative gain metrics (nCGp and nCG-1) are slightly below the median. These results reveal that our approaches have a good precision but low recall for 96 topics. These results can be explained by the low number of tweets pushed by our approaches due to the tweet-query word overlap and novelty thresholds.

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 we generate 3 different runs as follows:

1. The first run (IRIT-RunB1) takes as input tweets that passe the relevance filter based on adaptive learning strategy in which the relevance feedback of mobile assessor is used to retrain the binary classifier periodically. After filtering stage, tweets are clustered according to the content similarity. Then form a set of candidates tweets, a subset of tweets is selected so as to maximize their

overall relevance to the query subject to constraints related to, summary length, temporal diversity, coverage, and redundancy. In order to handle this selection, we formulate the tweet summary generation as integer linear problem (ILP) [8] in which unknowns variables are binaries and both the objective function (to be maximized) and constraints are linear in a set of integer variables. Constraints ensure that at most one post per cluster from the two categories of clusters (topical and temporal) is selected with respect to the defined summary length limit. To solve the problem, we use the GNU Linear Programming kit footnotehttps://www.gnu.org/software/glpk/, which is a free optimization package.
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 the same set of candidate tweet as the first run. In this run, candidate tweets are ordered by the relevance score computed according to the equation 4 and the top ten tweets are iteratively selected with the exclusion of those having word overlap above a static redundancy threshold set to 0.7.

3.1 Tweets clustering

The tweet clustering is based on a pairwise similarity comparison between an incoming tweet and centroids of existing clusters. For an incoming tweet T the key problem is to decide whether to absorb it into an existing cluster or to upgrade it as a new cluster. We first find the cluster whose centroid is the nearest to T. The decision of whether T is added to the closest cluster is made if the similarity score is greater than a predefined threshold $\gamma = 0.7$; otherwise, T is upgraded to a new cluster with T as the centroid. Each time an incoming tweet is added to an existing cluster its centroid is updated. We choose as new centroid the tweet that has the highest value of the sum of similarity scores with all other tweets in the cluster. The tweet-tweet similarity is estimated using word overlap.

3.2 Summary generation

From M candidate tweets (those that pass the relevance filter) we select N tweets which receive the highest relevance score with respect to the interest profile subject to series of constraints related to coverage, redundancy, and length limit (maximum number of tweets allowed in the summary set to 10). Assume that there are clustered in A clusters (denoted C_j) among them there are s clusters that contain at least two tweets. In the same way, The tweet summarization problem can be formulated as the following ILP problem:

We include an indicator variable X_i which is set to 1 when tweet T_i is added to the summary and 0 otherwise. The goal of the ILP is to set these indicators variables to maximize the payoff subject to the set of constraints that guarantee the validity of the solution. Notice here that the first constraint states that the indicator variables are binary.

$$\forall i \in [1, M], X_i \in \{0, 1\}$$

3.2.1 Objective function

Top-ranked tweets are the most relevant tweets corresponding to the related aspects which we want to include in the final summary. Thus, the goal is to maximize the global relevance score of selected tweets that improve the overall coverage, temporal diversity and relevance of the final summary. The objective function is defined as follows:

$$\max(\textstyle\sum_{i=1}^{M} X_i \times RSV(T_i, Q))$$

The final relevance $(RSV(T_i, Q))$ score of the tweet T_i with respect to the query is given by combining linearly the relevance score of tweet T regarding the query title (equation 1) and the query description (equation 2) as follows:

$$RSV(T, Q) = RSV(T, Q^t_{and}) + RSV(T, Q^d_{or}) \tag{4}$$

	nDCG@10-p	nDCG@10-1
IRIT-RunB1	0.2130	0.1962
IRIT-RunB2	0.2142	0.1833
IRIT-RunB3	0.2117	0.1961
Median	0.2194	0.1865

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

3.2.2 Coverage and redundancy constraints

These constraints fulfill both redundancy and coverage requirements. In order to avoid redundancy, we just choose at most one tweet from each topical cluster. Indeed, the limitation of the number of tweets from each cluster guarantees that a maximum of sup-topics (aspects) will be presented in the summary such that the summary can cover most information of the whole tweet set. These constraints are formulated as follows:

$$\forall C_j \in \{C_1,\ldots,C_s\} \sum_{i;T_i \in C_j} X_i \leqslant 1$$

3.2.3 Length Constraints

We add this constraint to ensure that the length of the final summary is limited to the minimum of either a predefined constant N (i.e. the maximum length) or $M-1$ where M is the number of candidates tweets.

$$\sum_{i=1}^{M} X_i \leqslant min(N,M-1)$$

3.3 Results for scenario B

Table 5 reports our results for Scenario B, which aggregated the user's interest in a daily summary with a maximum of 10 tweets. We note that performances of our runs in terms of nDCG@10p are below the median. The difference between performances of runs based on ILP compared to run (IRIT-RunB3) that is based on the section of the top-10 tweets is not significant. These results can be explained by the low number of candidate tweets that passe the relevance filtering stage. In fact, if the number of candidate tweets (M) is less than the maximum length of the summary N (set to 10 in our experiments), the ILP component acts almost like top-K ranking methods since it selects all candidate tweets with discarding the redundant ones.

4 Conclusion and future work

We presented in this paper three different approaches for real-time summerization of tweet stream. The proposed approached compute either a relevance score or filtering score which allow to determine if a new tweet should be pushed. For all these approaches, we underline that further experiments are needed, more particularly in the parameter tuning steps. However, we believe that results are quite promising and could give interesting insights in the future in terms of real-time tweet filtering, which are important components in the information access within data-streams.

References

1. 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.

2. Jimmy Lin, Adam Roegiest, Luchen Tan, Richard McCreadie, Ellen Voorhees, and Fernando Diaz. Overview of the trec 2016 realtime summarization. In *Text REtrieval Conference, TREC, Gaithersburg, USA, November 15-18*, 2016.
3. Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
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. Adam Roegiest, Luchen Tan, and Jimmy Lin. Online in-situ interleaved evaluation of real-time push notification systems. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 415–424, 2017.
8. Juraj Hromkovic and Waldyr M. Oliva. *Algorithmics for Hard Problems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2002.