University of Lugano at TREC 2015: Contextual Suggestion and Temporal Summarization Tracks

Mohammad Aliannejadi, Seyed Ali Bahrainian, Anastasia Giachanou, and Fabio Crestani

Faculty of Informatics, University of Lugano, Switzerland

{*mohammad.alian.nejadi|seyed.ali.bahreinain|anastasia.giachanou|fabio.crestani*}*@usi.ch*

ABSTRACT

This technical report presents the work of the University of Lugano at TREC 2015 Contextual Suggestion and Temporal Summarization tracks. The first track that we report on, is the Contextual Suggestion. The goal of the Contextual Suggestion track is to develop systems that could generate user-specific suggestions that a user might potentially like. Our proposed method attempts to model the users' behavior and interest using a classifier, and enrich the basic model using additional data sources. Our results illustrate that our proposed method performed very well in terms of all used evaluation metrics. The second track that we report on, is the Temporal Summarization that aims to develop systems that can detect useful, new, and timely updates about a certain event. Our proposed method selects sentences that are relevant and novel to a specific event with the aim to create a summary for this event. The results showed that the proposed method is very effective in terms of Latency Comprehensiveness (LC). However, the approach did not manage to obtain a good performance in terms of Expected Latency Gain (ELG).

Keywords: contextual suggestion, user modeling, SVM, temporal summarization, TREC, DFR

1. INTRODUCTION

This paper describes the participation of the University of Lugano (USI) at TREC 2015 Contextual Suggestion[1] [3] as well as the Temporal Summarization [2] track. This year's Contextual Suggestion track consisted of two tasks, namely, *Task 1: Live Experiment* and *Task 2: Batch Experiment*. We participated in the later task. In this task, for each user contexts and profiles are defined. Each profile consists of 60 places and the user's opinion regarding them. Additionally, We were provided with a list of 30 candidate suggestions and our task was to rank them. To address this task, we crawled all the main sources of information (i.e. Yelp, Foursquare, and TripAdvisor) to possibly build the most comprehensive dataset of attractions. We then built user model using machine learning classification. Furthermore, we enriched the basic model by combining simple yet effective additional measures with the base user model.

For this year's Temporal Summarization track, two tasks were defined: *Task 1: Filtering and Summarization* and *Task 2: Summarization Only*. We participated in the first task. For this task we were provided with high-volume streams of news articles and blog posts crawled from the Web regarding a set of specific events. The aim of the task was to process the streams in time order, filter out irrelevant content and then select the appropriate sentences that could summarize each event over time. The sentences were to be selected on the basis of being relevant, novel and important regarding a given event [2]. To address the task, we applied the Divergence From Randomness (DFR) Framework and particular InL2 model proposed by Amati and Van Rijsbergen [1] to calculate the relevance of a sentence given a topic. We also used query expansion to address the problem of word mismatch. To calculate the novelty of a candidate sentence, we leveraged the number of unique terms between the sentence and the summary already produced. We also considered different approaches to determine the number of sentences selected each time. We explored three different approaches: increasing the number of sentences as time passes, decreasing the number of sentences as time passes and selecting a stable number of sentences as time passes.

2. CONTEXTUAL SUGGESTION

In this task we modeled users with making use of Yelp data and we further enriched these models using data from other data sources. We used classifiers to model users' behavior and we enriched user models by combining other measures. Our system generally consists of four modules:

- Information gathering
- User modeling
- User model enrichment
- Suggestion ranking

Our system execution cycle starts with the data collection module. This module collects data such that data from the

[1] `https://sites.google.com/site/treccontext/`
[2] `http://www.trec-ts.org/`

most important data sources, namely, Yelp, Foursquare, and TripAdvisor are gathered. Using the gathered data, the user modeling module, creates a basic user model per user. Furthermore, the model enrichment module aims at improving the user models by fusing additional measures such as user category profile and Foursquare *taste* tag models with the basic models. Subsequently, by using the enriched models the suggestion ranking module ranks all candidate places. In the following subsections we elaborate on each component in greater detail.

2.1 Information Gathering

Considering that we participated in the batch experiments, we were provided with users' history and the attractions to be ranked. Thus, the number of links presenting attractions that we had to process was reduced to virtually 9000. Since, the task of suggesting places per each given user was limited to 30 attractions, it was crucial not to miss any information relevant to the target attractions. Additionally, although almost half of the URLs was from known sources such as Yelp and Foursquare, another half of the URLs we were provided, was from less known websites (e.g. the places' official web pages). Consequently, effort was made to find the corresponding profiles of these places on Yelp, Foursquare, and TripAdvisor. To collect data we performed the following steps:

1. We discard the attractions that users assigned them a score of '−1' or '2'. This is due to the fact that these places either weren't assigned any ratings or their rating was neutral, thus insignificant.
2. We detect and discard broken links.
3. We download the links from the known sources, namely, Yelp, Foursquare, and TripAdvisor.
4. For each attraction on each of the above-mentioned sources we find the corresponding profiles on the other two sources. (e.g. for a given Yelp profile, we find its corresponding profiles on Foursquare and TripAdvisor).
5. For the other attractions with unknown links, we download the web pages, analyze their contents to find their corresponding profiles on the three above-mentioned websites.

Following the above steps, we crawled homogeneous data for virtually all given attractions from the three websites. The final layout of our dataset is as follows:

- Yelp
  - Name
  - Yelp URL
  - Overall rating
  - Categories
  - Subcategories
  - Reviews
    - Rating
    - Comment
    - Date
    - ...
  - ...
- Foursquare
  - ...
  - Tips
  - Visits
  - Visitors
  - ...
- TripAdvisor
  - ...
  - Dining options
  - Rating summary
  - Attraction ranking
  - ...

2.2 User Modeling

We model each user by training a classifier using example suggestions. Our intuition is that a user's opinion regarding an attraction could be learned based on the opinions of other users who gave the same rating as the target user to the same attraction. To train a classifier per user we extract negative and positive samples as explained in the following:

- **Positive samples:** We elicit the positive reviews of positive example suggestions.
- **Negative samples:** Likewise, we elicit the negative reviews of negative example suggestions.

We define positive example suggestions as the attractions which a user rated 3 or 4. Positive reviews are the reviews whose corresponding ratings are 3 or 4. Analogously, negative example suggestions and reviews are defined, taking ratings of 0 and 1 as negative.

We use Support Vector Machine (SVM) and Naive Bayes (NB) as the classifiers of our choice. We train these classifiers using TF-IDF measures as feature vectors.

2.3 User Model Enrichment

This module takes as input the created user model and further enriches them using the following algorithm:

1. For each user and for each place we create an index of all the categories (e.g. Italian Restaurant), based on Yelp data, he/she has referred to as positive or negative.
2. Then we compute the normalized count per category and add it to positive or negative user models.

3. By having all normalized counts per category for both the positive and negative models for all places, we compute two negative and positive vectors which represent the user's interests in various categories.

4. Finally, for a given place in which we would like to predict the level of user's interest, we sum up all the normalized scores for all the existing categories for the place (UI). The result would be a score between '−1' and '+1'

We follow the above procedure to compute a similar interest score based on the *taste* tags regarding each place on Foursquare (UF). Likewise, a third score is computed based on the categories of TripAdvisor (UT).

Foursquare *taste* tags are special terms extracted from users' tips[3] and are very informative. For example, 'Central Park' in 'New York' is described by these *taste* terms: picnics, biking, trails, park, scenic views, etc. These terms are very informative and often express characteristics of an attraction. Therefore, they can be considered as categories.

2.4 Suggestion Ranking

We estimate the similarity between each user and a candidate suggestion using the equation below:

$$\begin{aligned}\text{Similarity}(u,p) = \omega_1 UC(u,p) + \omega_2 UI(u,p) + \\ \omega_3 UF(u,p) + \omega_4 UT(u,p)\end{aligned} \quad (1)$$

Where $\omega_{1...4}$ are the weights assigned to these scores, u is a given user, p is a given attraction, and UC is the confidence score of the classifier which models a user. To find the optimum setting for the weights associated with each score, we conducted a 5-fold cross validation, for which the best setting is:

$\omega_1 = 1$	$\omega_2 = 1$	$\omega_3 = 0.3$	$\omega_4 = 0.3$

We rank the candidate suggestions according to the similarity measure computed by this module. The higher the similarity score, the higher the rank would be.

2.5 Experimental Results

By applying our method to our gathered dataset, we submitted two runs: '*11*' and '*22*'. The two runs differ from one another merely in that, run '*11*' utilizes an SVM classifier while '*22*' is based on a NB classifier. The best sets of configuration parameters for the classifiers and equation (1) were obtained based on a 5-fold cross validation.

In this year's task, we were given 211 profile/context pairs. For each pair a user's history was 60 places and the number of candidate suggestions to be ranked were 30. How interesting a website is, has a scale from '0' up to '4', with '0' being the least interesting, '2' neutral, and '4' the most interesting. Track organizers evaluated all submitted runs using two evaluation metrics, namely, P@5 (precision at 5) and MRR (mean reciprocal rank). In this evaluation, a suggestion is relevant if it is rated 3 or 4 by user.

[3]Tips on Foursquare are short reviews written by users.

Table 1: Overall Average Performances

Runs	P@5	MRR
11	**0.5858**	**0.7404**
22	0.5450	0.6991
TREC Median	0.5090	0.6716

Table 1 demonstrates the overall average performances of our runs. It could be seen that both of our runs outperform the median performance of all submitted runs by other contestants, which confirms the effectiveness of the user modeling and enrichment method we proposed in this work. However, our run '11' performs better than the other, which proves that in this task, an SVM classifier is a more powerful tool for building the basic user model.

Furthermore, the optimal parameter set, $\omega_{1...4}$ indicates that in our system the basic user model (UC) and category model (UI) are more important, since their corresponding weights are higher.

3. TEMPORAL SUMMARIZATION

In recent years, temporal summarizing received a lot of attention in the research community. To address this problem, a method should address three sub-problems: (a) *relevance* of sentences by selecting those that are the most relevant given an event, (b) *novelty* by selecting the sentences that contain novel content and (b) *importance* by selecting the sentences that a person would put into a summary. In terms of relevance towards a given event, any IR relevance model can be used. Previous studies have applied different techniques including BM25 and vector space model [8]. To calculate the novelty of a sentence, researchers usually measure the text similarity between the candidate sentence and the summary already produced and remove those that are too similar [4]. Calculating importance is more challenging than calculating relevance and novelty. In terms of importance, Xu et al. [7] proposed a language model from named entities to measure the sentence's topical salience.

Another important challenge in the temporal summarization is the number of sentences to select from each time interval. Most of the approaches in the literature, select a stable number of sentences as time passes. Instead, McCreadie et al. [5] considered to adjust the number of the selected sentences as time passes. This idea is based to the fact that as time passes, the need for new sentences is not the same. The results showed that this approach performs well.

3.1 Methodology

When a user submits a query to a temporal summarization system, it aims to identify the most relevant, novel and important sentences. In this section, we present our methodology.

3.1.1 Relevance

In order to rank the sentences by relevance, we applied the Divergence From Randomness (DFR) Framework proposed by Amati and Van Rijsbergen [1]. In particular, we applied InL2 model since this model is effective for tasks that require early precision. DFR models are based on the idea that the

term-weight is inversely related to the probability of term-frequency within the document d. In our case, we consider each sentence s as a document. Therefore:

$$weight(t|s) =\propto -logProb_{InL2}(t \in s|Collection) \quad (2)$$

Based on the equation 2 we can then calculate the relevance of the whole sentence s as follows:

$$Relevance(s) = \sum_{t \in s} weight(t|s) \quad (3)$$

When a term that is rare in the collection appears a lot in a sentence, then this term has high probability to be informative for the topic discussed in the sentence. In other words, if a term frequency is high, then the risk of not being informative is low. In this case, equation 2 gives a high value but a minimal risk provides a small information gain. Equation 2 is smoothed to consider the portion of weight which represents the information gained with the term:

$$gain(t|s) = P_{risk} * (-logProb_{InL2}(t \in s|Collection)) \quad (4)$$

In the equation 4 $Prob_{InL2}$ represents the model of randomness and is calculated as:

$$-logProb_{InL2}(t \in s|Collection) = -log_2 \frac{N+1}{n_t + 0.5}$$

where n_t is the term frequency of the term t in the collection and N is the number of documents in the collection.

To compute the information gain with a term within a sentence, we use the Laplace model:

$$P_{risk} = \frac{1}{tf + 1}$$

where tf is the term frequency of the term t in the sentence s.

Before using the equation 4, the sentence length dl is normalised to a standard length sl. The term frequencies are calculated with respect to the standard sentence length, as:

$$tf_n = tf * log(1 + c * \frac{sl}{dl})$$

3.1.2 Query Expansion

One frequent problem that hurts the performance of a retrieval system when returning relevant documents given a query is word mismatch. This problem occurs because users do not use the same words to form the query as those used in relevant documents. To address this problem, we used query expansion through which the query is expanded with new terms and this possibly helps to retrieve more relevant documents. For our experiments, we expanded the query using the 5 most appeared words in the summary having been already produced.

3.1.3 Novelty

Selecting the most relevant sentences is not enough for a temporal summarization system. In addition to the relevance, the system should also select the most novel sentences. In order to measure the novelty of a sentence, we computed the similarities between a candidate sentence and the summary that was already produced (until the specific timestamp). We applied a simple method that was based on the number of unique terms of the candidate sentence when compared to the generated summary. To present the method more formally, we introduce some notation. Let S_i represent the summary produced until the timestamp t_i and s_{i+1} a candidate sentence with a timestamp t_{i+1}.

$$Novelty(s) = Novelty(s_{i+1}) = \frac{N_T(S_i, s_{i+1})}{|s_{i+1}|}$$

where $N_T(S_i, s_{i+1})$ is the number of unique terms in the sentence s_{i+1} when it is compared with the summary S_i already generated until the timestamp t_i and $|s_{i+1}|$ is the length of the sentence s_{i+1}.

3.1.4 Combine Relevance and Novelty

As a last step, we need to combine the relevance and the novelty of the sentences to produce the final ranking. We factorize relevance and novelty as follows:

$$Score(d) = (\lambda) * Relevance(s) + (1 - \lambda) * Novelty(s)$$

where $Relevance(s)$ denotes the relevance of the sentence, $Novelty(s_{i+1})$ denotes the amount of novelty in the sentence s_{i+1} compared to the summary S_i and $\lambda \in [0, 1]$.

Because of the lack of training data, it is not possible to find the best parameter for λ. Therefore, λ is set to 0.5 so relevance and novelty contribute equally to the final ranking.

3.2 Experiments

In this section, we present more details for our experiments.

3.2.1 Dataset

This year's temporal summarization track was based on the TREC KBA 2014 Stream Corpus. The corpus contains a set of timestamped documents crawled from a variety of news and social media sources. The documents in the corpus span from October 2011 until April 2013. Each document in the collection comprises of a set of sentences. Each sentence has a unique identifier. Due to the size of the KBA 2014 corpus, two smaller datasets were released: the TREC-TS-2015F and the TREC-TS-2015F-RelOnly. The TREC-TS-2015F is a pre-filtered dataset and contains for each event the top documents from a high precision retrieval process. This dataset is high-volume and contains a lot of irrelevant documents. The TREC-TS-2015F-RelOnly collection contains manually selected documents for each event and therefore does not need any filtering to remove the irrelevant content.

For our experiments we used the TREC-TS-2015F dataset and participated in the *filtering and summarization* task.

This year's task included 21 events, each of which represented by a set of features including title, description, start

time, end time, query and type. In this year's task, the type of the events could be any of bombing, accident, earthquake, protest, storm, conflict.

3.2.2 Preprocessing

The TREC-TS-2015F corpus is about 38GB and needs preprocessing. The overall process is as follows:

- Decrypt the file: After having downloaded the files, we need to decrypt them. To decrypt the files, we use the authorized key provided by the organizers. This step converts the GPG file format to SC.
- Parsing: The SC files are then converted to TXT files. We use the streamcorpus toolbox[4] to parse the files.
- Index: The last step is to build the index. We index the collection with the Terrier IR system[5]. Our preprocessing also involves stop-word removal and stemming using the Porter stemmer [6]. We consider each sentence as a document and create separate indexes for each timestamp.

3.2.3 Submitted Runs

We submitted seven runs for the *filtering and summarization* task. For all the runs, we used the same approach to calculate relevance and novelty in order to generate the final ranking. For the relevance, we have used DFR_{InL2}. The novelty score is based on the number of novel terms that each sentence has compared to the terms of the summary that was already produced until the specific timestamp. The runs differ in terms of how the number of sentences was decided at each timestamp and the way that the first sentence of the summary was selected. After having selected the first sentence of the summary, we use query expansion for the rest of the blocks. For the query expansion, we use the top 5 most frequent terms of the summary already produced. Here we present details of our pooled runs:

- InL2DecrQE1ID1: For this run the number of sentences selected each hour decreases as time passes. The first sentence of a summary should contain all the query terms.
- InL2DecrQE2ID2: For this run the number of sentences selected each hour decreases as time passes. The first sentence of a summary may contain any of the query terms.
- InL2StabQE2ID3: For this run the number of sentences selected each hour is the same for every block and depends on the number of time blocks per event. The first sentence of a summary may contain any of the query terms.
- InL2IncrQE2ID4: For this run the number of sentences selected each hour increases as time passes. The first sentence of a summary may contain any of the query terms.

[4]http://streamcorpus.org/

[5]Available at: `http://terrier.org/`

3.2.4 Evaluation Metrics

To evaluate the effectiveness of the systems, track organizers define two metrics: the Expected Latency Gain (ELG) and the Latency Comprehensiveness (LC) which are similar to the traditional IR metrics Precision and Recall respectively. ELG measures the sum of latency-discounted relevance of he nuggets for which that update is the earliest issued. LC measures the nugget recall over all updates issued, where the score for a nugget is not considered if it is reported late. Systems are ranked based on the harmonic mean between ELG and LC, denoted as H.

3.2.5 Results

Table 2 reports the performance of our pooled runs for the *Filtering and Summarization* task in terms of ELG, LC and H for the task filtering and summarization.

Table 2: Results of our runs

	ELG	LC	H
InL2DecrQE1ID1	0.0185	0.3203	0.0184
InL2DecrQE2ID2	0.0089	0.3284	0.0172
InL2StabQE2ID3	0.0082	0.3225	0.0159
InL2IncrQE2ID4	0.0053	0.1848	0.0101
ALL submitted runs	0.0483	0.2381	0.0612

The results show that all our submitted runs submitted very well in terms of LC. That means that we manage to detect a good number of sentences that should be included in the summary. However, we do not perform well in terms of ELG. This suggests that our produced summaries include a lot sentences that should not be part of the summary. We believe that one reason for that is that we did not consider importance at all. Trying better techniques to filter out the irrelevant documents and measuring the importance of the sentences are two directions we want to explore in the future.

4. CONCLUSIONS

In this technical report, we presented the methodology followed for our participation in the 2015 Contextual Suggestion and Temporal Summarization tracks. In the Contextual Suggestion track, we showed that our method in recommending places to users based on their profiles is very effective. The approach we followed for the Temporal Summarization track performed very well in respect to latency comprehensiveness. However, it did not perform well in terms of the expected latency gain. The results suggest that we should modify our approach in a way that the expected latency gain can be improved. One possible way is to also consider the importance of the sentences.

Acknowledgement

This research was partially funded by the RelMobIR and OpiTrack projects of the Swiss National Science Foundation.

5. REFERENCES

[1] G. Amati and C. J. Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389, 2002.

[2] J. Aslam, F. Diaz, M. Ekstrand-Abueg, R. McCreadie, V. Pavlu, and T. Sakai. TREC 2014 Temporal Summarization Track Overview. In *Proceedings of the 23rd Text REtrieval Conference (TREC 2014)*, 2014.

[3] A. Dean-Hall, C. L. Clarke, J. Kamps, P. Thomas, and E. Voorhees. Overview of the trec 2014 contextual suggestion track. Technical report, DTIC Document, 2014.

[4] R. McCreadie, R. Deveaud, M.-D. Albakour, S. Mackie, N. Limsopatham, C. Macdonald, I. Ounis, T. Thonet, and B. T. Dincer. University of Glasgow at TREC 2014: Experiments with Terrier in Contextual Suggestion, Temporal Summarisation and Web Tracks. In *Proceedings of the 23rd Text REtrieval Conference (TREC 2014)*, 2014.

[5] R. McCreadie, C. Macdonald, and I. Ounis. Incremental update summarization: Adaptive sentence selection based on prevalence and novelty. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 301–310, New York, NY, USA, 2014. ACM.

[6] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[7] T. Xu, P. McNamee, and D. W. Oard. HLTCOE at TREC 2013: Temporal Summarization. In *Proceedings of the 22nd Text REtrieval Conference (TREC 2013)*, 2013.

[8] Y. Zhao, F. Yao, H. Sun, and Z. Yang. TBJUT at TREC 2014 Temporal Summarization Track. In *Proceedings of the 23rd Text REtrieval Conference (TREC 2014)*, 2014.