Online Temporal Summarization of News Events

Tobias Schubotz
Hasso Plattner Institute
Potsdam, Germany
Email: tobias.schubotz@student.hpi.de

Ralf Krestel
Hasso Plattner Institute
Potsdam, Germany
Email: ralf.krestel@hpi.de

***Abstract*—Nowadays, an ever increasing number of news articles is published on a daily basis. Especially after notable national and international events or disasters, news coverage rises tremendously. Temporal summarization is an approach to automatically summarize such information in a timely manner. Summaries are created incrementally with progressing time, as soon as new information is available. Given a user-defined query, we designed a temporal summarizer based on probabilistic language models and entity recognition. First, all relevant documents and sentences are extracted from a stream of news documents using BM25 scoring. Second, a general query language model is created which is used to detect typical sentences respective to the query with Kullback-Leibler divergence. Based on the retrieval result, this query model is extended over time by terms appearing frequently during the particular event. Our system is evaluated with a document corpus including test data provided by the Text Retrieval Conference (TREC).**

I. Introduction

A great amount of news articles is produced every day which makes it harder and harder for interested readers to stay up to date with all information available and to filter articles for novel and relevant content. Additionally, users do not want to read the same information over and over again. Therefore, summarization of information containing all novel and relevant event updates is crucial. Usually it is the task of newspapers to summarize all recent and relevant content. As time proceeds, often an overview or a timeline over past updates is given. The main shortcoming of this approach is that tremendous effort is required by journalists to read and consume all available and relevant news before a manual news timeline can be created. Hence, there will always be a significant time delay between occurrence of news and timeline publication, even when publishing on the Internet. The process of creating summaries with an emphasis on the temporal order of events is called *temporal summarization*.

Li et al. [1] explore the problem of generating storylines from microblog data for user-defined queries. The system they developed is only applicable to retrospective data analysis. Shou et al. [2]. propose a system that uses stream clustering techniques for continuous summarization of Twitter streams. Yan et al. [3]. propose a framework called Evolutionary Timeline Summarization (ETS). Given a user-defined query, a timeline created by the system needs to score high concerning four metrics:relevancy, coverage, coherence, and diversity. By optimizing based on these measures, ETS selects the best sentences for the summary. In contrast to our work, ETS is mainly built for offline applications, meaning that the optimization for a given time frame is dependent on the knowledge of all past and future data in advance. Tran et al. [4] built a system that exploits the headline of news articles to build timelines automatically. Event Registry is a system developed by Leban et al. [5] for the temporal summarization of news events. By using data from a newsfeed, core information such as time, location and who is involved are extracted automatically by named entity recognition. Leban et al. apply stream clustering algorithms in order to identify all articles that refer to the same event. Event Registry contains no efforts to reduce redundancy within resulting data. Beside clustering, there is no further summarization involved. That is why we consider it to be merely a system for temporal event exploration instead of real temporal summarization.

A problem with all kinds of summaries is that some information that was left out due to relevance or brevity might be relevant for some readers. Temporal summarization cannot overcome this issue. However, temporal summarization of events, as they evolve and news get published, is superior in showing what was considered important at the time of the update. Existing approaches are mainly intended to be applied after some time has passed in order to decide later on what is important instead of making a live decision. Rarely, evolutionary characteristics of news are considered. *Online temporal summarization* [6] tries to overcome these issues. It aims at providing summarization techniques that are temporally as close as possible to the occurrence time of information and articles. That means, it should be decided whether they contain relevant and new information as early as possible, preferably at the time new articles become available. Systems have to make decisions based on information available up until now.

Several related approaches have already been developed in this context [7]. Liu et al. [8] propose a relatively simple method where only the titles of news articles are compared against user-specified query terms. Zhang et al. [9] apply query expansion and use latent Dirichlet allocation to detect potential topics according to a given query.

II. An Online Temporal Summarizer

Conceptionally, three main steps are performed for temporal summarization. (1) The input query and incoming documents need to be processed. If applicable, query expansion and any further preprocessing of data takes place. (2) From all available documents, the relevant sentences have to be extracted. (3) Finally, from the set of relevant sentences, novel ones need to be chosen that conatin both relevant and novel information. Figure 1 depicts our implementation of this processing chain.

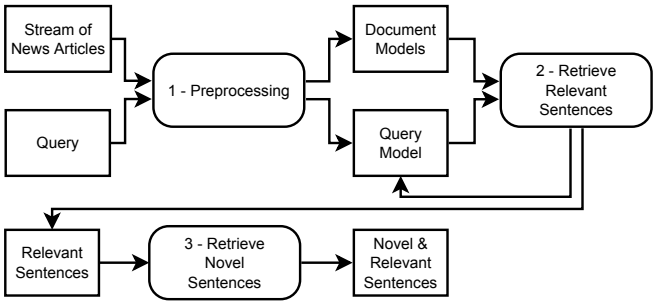

Fig. 1. Overview of our processing steps for temporal summarization.

Inputs to the system are a stream of news articles and a query (together with a broad category such as *"accident"* or *"earthquake"*) defined by a user. The preprocessed articles are converted to document bag-of-words models. Moreover, a query model is created from either the raw query or the result of a preprocessing step (e.g. query expansion). Both models are used to retrieve relevant sentences from the stream of articles. The directed edge back to the query model indicates, that our system uses information about relevant sentences in order to *refine the query model incrementally*. Finally, based on the set of relevant sentences, novel sentences are detected. We describe the steps in more detail in the following using the user-supplied query: *"buenos aires train crash" (accident)* as our running example.

III. Relevant Update Retrieval

The goal of the following is to retrieve relevant documents and sentences. We decided to combine document and sentence level retrieval and developed a system that first filters on document-level, before typical sentences are retrieved in a second filtering step. For both steps, we regard queries, documents, and sentences as single bag-of-words models.

A. Detecting Relevant Documents

We use BM25 to score each incoming document with respect to the given query terms. For the IDF-component of BM25 we rely on a background document corpus from the past. This background corpus contains only documents published before the current event start date and allows to distinguish general from specific keywords. For our example query we want to retrieve relevant documents about the train crash in Buenos Aires. We will most likely also include documents about other events in Buenos Aires or train crashes that occurred somewhere else.

B. Detecting Typical Sentences

After identifying relevant documents for the user-supplied query in the stream of news articles, we are now faced with the challenge of extracting relevant sentences. For query Q1 there could be documents about other events in Buenos Aires that scored high because the city's name occurred in them several times. We need to filter out such sentences. Therefore, we aim at detecting the underlying general topic of the query in order to extract only appropriate sentences that represent good candidates for novel event updates. Our main idea for this step is to create a language model representing the topic of the query. To automatically build such a language model we make use of the category information of each query. The category for *"buenos aires train crash"* is *"accident"*. We can combine each individual query term and the category in order to determine the topic of the event by checking how many article titles in Wikipedia matched for each combination *(query term, query category)*. For our example "train accident" has the most matches and we can use this combination to build a topical language model using Wikipedia. Just the category information *"accident"* would have been too broad and would not have given us train accident specific Wikipedia pages containing terms such as *"platform"*, *"locomotive"*, or *"conductor"*. Instead, we retrieve the top 100 articles from Wikipedia scored by BM25 using the topical query "train accident". We aggregate all terms of these 100 articles to get a query language model LM_Q. Finally, we compute Kullback-Leibler divergence (KLD) between this language model LM_Q and each sentence language model LM_S of all sentences S found in relevant documents.

C. Refining the Query Language Model

Up to this point, the language model LM_Q used to retrieve typical sentences by KLD-score is determined once at the query start time and does not change over time. This yields two shortcomings that could impair retrieval performance. First, if the original LM_Q constitutes not a good fit for the original query, our approach is not able to detect this and adjust accordingly. Besides that, as time progresses, knowledge about retrieved sentences is accumulated, but never utilized to refine LM_Q. Therefore, we store frequencies of all terms in the set of relevant sentences R in order to create a language model LM_R for all sentences judged relevant. LM_R is used to update LM_Q in periodic time intervals. In our experiments, an update interval of one hour showed the best retrieval performance. At the end of every hour, we extract the 50 most frequent terms from LM_R and extend the original LM_Q.

IV. Novel Update Retrieval

After retrieving relevant sentences the system now needs to filter out the novel ones. There are two basic strategies: We can assume all sentences to be novel and only discard the ones that are duplicates, i.e. that are too similar to an earlier sentence. Or we can assume no sentence is novel by default and only explicitly select the ones that contain novel information. We decided for the latter approach since it allows us to model new information more directly. To ensure we only get significant and novel information we require new information to be mentioned by more than just one source before we accept it as reliable (significant), novel information.

A. Novel Updates by Number Occurrence

We assume that new updates often contain numbers. Numbers in articles could represent important facts like the number of casualties, maximum sustained winds of hurricanes, or the magnitude of earthquakes. As time progresses, these numbers get updated. We introduce a threshold t_{num} that a particular number needs to occur before it is considered a significant novel number. Numbers usually do not stand alone. Instead, the context is important in order to determine whether the

number represents injuries, deaths, speeds, etc. We refer to these words before and after a number as *context words*. Instead of just looking at plain numbers, we think that this context can lead to more valuable updates. For example, in reports on train crashes, the speed of the train is very important. Consequently, all numbers in front of "mph" or "kph" (miles or kilometers per hour) are particularly important. Given a query, we already extracted the top 100 matching Wikipedia articles. After processing these articles with a POS tagger, we extract all words that occur before and after numbers and which are not numbers by themselves. The result is a set of number context words. Vice versa, a number that occurs before or after such a word is called *context number*. We only use the top 50 context words that occurred most frequently. For query Q1, the top words are "injuries", "deaths", and "passengers" For a relevant sentence we can now extract all context numbers. We consider the number to be significant enough if it occurred at least $t_{context\_num}$ times. At the point in time where we decide that a number is significant and novel, it occurred at least t_{num} or $t_{context\_num}$ times before, respectively.

B. Novel Updates by Term Frequency

Besides numbers also other terms can indicate novel information. Therefore we incrementally build a language model LM_R, as mentioned before, based on the sentences judged to be relevant and novel so far. We monitor the inverse document frequencies (IDFs) of all terms and compare them with the respective IDFs in the background corpus. In order to be regarded as significant and novel, a term has to fulfill the following three criteria: (1) The IDF of the term during the time frame of the temporal summary so far (IDF_{TS}) needs to be less then its IDF within the background corpus (IDF_B): $IDF_{TS} \leq p \cdot IDF_B$, with $p \in [0, 1]$ (we used $p = 0.25$). (2) The term has to occur at least t_{term} times during the period of the temporal summary so far. (3) The term has not been selected to be significant and novel before.

C. Completing the Temporal Summary

Finally, relevant sentences that satisfy at least one of the criteria described in the past two sections are selected as novel update: Such sentences need to contain either a novel and significant number, a context number, or a novel, relevant term. By introducing the thresholds t_{num}, $t_{context\_num}$, and t_{term} we are also introducing latency into the temporal summary. That means, when a relevant update is retrieved there is a time gap before our system decides if the update contains novel information. Since we deal with global events where lots of news sources are available (and lots of noise as well) this strategy is a good trade-off between latency and irrelevant or false updates.

V. Evaluation

The TREC temporal summarization track[1] offers guidelines, test data and metrics to evaluate online temporal summarization systems. By using this framework we can also compare our performance not just with a baseline but with other participants of the track.

[1] http://www.trec-ts.org/

A. TREC Corpus, Test Data, and Metrics

We used the dataset from the 2013 TREC temporal summarization track. The corpus contains documents of various sources between October 2011 and April 2013. Since the entire corpus contains over one billion documents, we filtered out parts of it. First, we only included time frames that were relevant to the test queries. Afterwards, documents created in any language other than English were discarded. This resulted in a corpus containing approximately 85 million documents. Document sources occurring the most were weblogs, news and social media. There were two main challenges we faced when processing this corpus: boilerplate code removal and sentence splitting were performed poorly by TREC. But we did not correct these mistakes to be able to compare our results with the participating systems. Alongside the corpus, TREC defined a set of ten queries that are used for evaluation of temporal summaries. Each query contains query terms, a time frame of ten days, and a category. On average, there are approximately 8 million documents per query time frame corresponding to 581 documents per minute. Human assessors created a silver standard by matching outputs of participants' systems to a list of predefined information pieces (nuggets). Based on this matching, performance of a system is assessed and compared by metrics proposed by TREC[2]. We report results on *verbosity normalized latency-discounted expected gain* $\boldsymbol{E_V[G_L]}$ and *latency comprehensiveness* $\boldsymbol{C_L}$. The first one can be seen as a measure for precision whereas the latter one describes rather the recall of a system. Temporal summarizers are usually optimized for either expected gain or comprehensiveness. Optimizing for both is rather challenging. This makes it hard to compare systems with opposing optimization goals. Therefore, we adapt the $\boldsymbol{F_1}$ *score* traditionally used in information retrieval to evaluate trade-offs between precision and recall, and apply it to expected gain and comprehensiveness: $F_1 = 2 \cdot \frac{E_V[G_L] \cdot C_L}{E_V[G_L] + C_L}$

B. Experimental Setup

We will perform three evaluation runs. The name in boldface represents the run identifier used in the following sections. The middle part indicates the approach used for finding relevant sentences, the third part indicates the approach used to filter novel information.

HPI-BL-BL: This run employs baseline approaches for retrieving relevant as well as novel updates. We use a boolean AND search to find documents that contain all query terms and then mark the sentences in these documents relevant that conatain at least one query term. A sentence was then considered novel if it contains a novel number or a novel entity mention.
HPI-LMe-BL: This run uses the described methods including query model refinement for finding relevant sentences and applies a baseline novel update retrieval approach based on novel numbers and entities.
HPI-LMe-TempSum: Finally, this run implements our entire approach for online temporal summarization. After retrieving relevant sentences by using BM25 and KL scoring, novel updates are retrieved that contain novel and significant numbers, context numbers, or terms.

[2] http://www.trec-ts.org/metrics-10242013.pdf

C. Results & Discussion

Table I depicts latency-discounted expected gain and comprehensiveness scores for the two best TREC 2013 runs, the average for all TREC 2013 runs, as well as our 3 runs. Furthermore, F_1 scores as well as the average number of sentence updates per query each system has outputed. Boldface values represent the best result per column. The TREC results were calculated based on a set of matches created by TREC assessors. We extended this original set of matches in order to be able to calculate all metrics for our runs as well. All values in Table I were determined by using the official evaluation script provided by TREC. The absolute number of updates per query is important with respect to the assumed scenario: a journalist who uses a temporal summary for research purposes is probably willing to read through more sentence updates than a casual reader.

TABLE I. RESULTS OF OUR RUNS (PREFIXED WITH *HPI*) IN COMPARISON TO THE RESULTS OF BEST TREC 2013 SYSTEMS AND TREC 2013 AVERAGE RESULTS.

Run ID	F_1	$E_V[G_L]$	C_L	#Updates per Query
ICTNET-run2	0.133	0.127	0.251	55
PRIS-cl5	0.119	**0.136**	0.126	22
AVG TREC 2013	0.089	0.058	0.288	146
HPI-LMe-TempSum	**0.162**	0.129	**0.545**	280
HPI-BL-BL	0.102	0.075	0.344	2,428
HPI-LMe-BL	0.096	0.068	0.287	171

As shown, our proposed temporal summarization system provides well above average performance considering expected gain and comprehensiveness. However, when inspecting the results there are still opportunities for improvement of precision and recall. From the retrieved updates just over 10% contain relevant and novel information. Furthermore, the system was not able to detect at least half of all existing nuggets. We identified mainly two reasons for that: our baseline as well as our actual approach extracted previously unseen numbers in order to detect novel content. While the baseline run output contained a lot of numbers that were not relevant, most numbers found by our actual approach proved to be indeed relevant and novel. However, only some of them were represented in the set of nuggets. For instance, when a fictitious numbers of casualties rose from 50 to 100 according to nuggets, our system also found intermediate number in between. Of course, this is not true for all our updates, but the precision would have been better with more fine-grained nuggets. While this is an explanation for imperfect precision, the reason for imperfect recall is different. The quality of the nuggets varies accross queries. Good, short nuggets are, e.g. , "hundreds injured" or "dozens killed". In contrast to that, there are nuggets such as "tropical storm (TS) watch: Jamaica, Bahamas, the Florida Keys" which do not carry concise information. These issues are part of the reasons why the recall for all approaches never gets close to 100%.

We see two scenarios our system could support. First, our temporal summarizer could act as a system that integrates other sources for professional readers like journalists. The main objective for them is the completeness of information. We think that our system is able to support this scenario, since our summaries sometimes contain even more detailed information than the set of nuggets. The second scenario corresponds to a casual reader who explores past events with temporal summaries in order to understand the evolution of the event. We think that our system supports this scenario better than the first one, since completeness is not the main objective for such users. It is more important that the summary is brief enough. This is satisfied by our system with an average of 280 updates during a time period of ten days.

VI. CONCLUSIONS

In this paper, we developed a system for online temporal summarization of news articles. Temporal summarization is typically performed in two steps: At first, relevant updates are extracted from a stream of documents, before the novel ones are emitted as novel event updates. Given a user-defined event query, we determined which documents are important by means of BM25 scoring. In order to extract relevant sentences, we automatically created a language model respective to the query based on a background corpus of Wikipedia articles. This model represents the underlying topic of the query. By employing a scoring based on KL divergence we are able to retrieve a set of relevant sentences. The query language model is extended regularly by the most frequent terms from this set.

We observed that novel updates often contain numbers that are updated regularly. Depending on the type of an event, numbers could be for instance casualties or wind speeds. Hence, we filtered for the occurrence of numbers in query-specific contexts. We also introduced the notion of significancy, which means that new information needs to be backed by several updates in order to be emitted as novel update. Furthermore, another incremental language model based on upcoming relevant sentences has been created. By using this language model and a background corpus, novel terms were detected that usually do not occur that often. Consequently, they are important concerning the query and used as indicators to detect novel updates.

REFERENCES

[1] C. Lin, C. Lin, J. Li, D. Wang, Y. Chen, and T. Li, "Generating Event Storylines from Microblogs," in *Proceedings of CIKM*. ACM, 2012, pp. 175–184.

[2] L. Shou, Z. Wang, K. Chen, and G. Chen, "Sumblr: Continuous Summarization of Evolving Tweet Streams," in *Proceedings of SIGIR*. ACM, 2013, pp. 533–542.

[3] R. Yan, X. Wan, J. Otterbacher, L. Kong, X. Li, and Y. Zhang, "Evolutionary Timeline Summarization: a Balanced Optimization Framework via Iterative Substitution," in *Proceedings of SIGIR*. ACM, 2011, pp. 745–754.

[4] G. B. Tran, M. Alrifai, and E. Herder, "Timeline summarization from relevant headlines," in *ECIR*. Springer, 2015, pp. 245–256.

[5] G. Leban, B. Fortuna, J. Brank, and M. Grobelnik, "Event Registry — Learning About World Events From News," in *Proceedings of WWW*, 2014, pp. 107–110.

[6] Q. Guo, F. Diaz, and E. Yom-Tov, "Updating Users About Time Critical Events," in *Proceedings of ECIR*. Springer, 2013, pp. 483–494.

[7] J. Aslam, F. Diaz, M. Ekstrand-Abueg, V. Pavlu, and T. Sakai, "TREC 2013 Temporal Summarization," in *Proceedings of TREC*. NIST, 2013.

[8] Q. Liu, Y. Liu, and D. Wu, "ICTNET at Temporal Summarization Track TREC 2013," in *Proceedings of TREC*. NIST, 2013.

[9] C. Zhang, W. Xu, F. Meng, H. Li, T. Wu, and L. Xu, "The Information Extraction systems of PRIS at Temporal Summarization Track," in *Proceedings of TREC*. NIST, 2013.