Creation, Visualization and Edition of Timelines for Journalistic Use

Xavier Tannier
LIMSI, CNRS, Univ. Paris-Sud,
Université Paris-Saclay
F-91405 Orsay, FRANCE
xavier.tannier@limsi.fr

Frédéric Vernier
LIMSI, CNRS, Univ. Paris-Sud,
Université Paris-Saclay
F-91405 Orsay, FRANCE
frederic.vernier@limsi.fr

Abstract

We describe in this article a system for building and visualizing thematic timelines automatically. The input of the system is a set of keywords, together with temporal user-specified boundaries. The output is a timeline graph showing at the same time the chronology and the importance of the events concerning the query. This requires natural language processing and information retrieval techniques, allied to a very specific temporal smoothing and visualization approach. The result can be edited so that the journalist always has the final say on what is finally displayed to the reader.

1 Introduction

Timelines are a natural way to describe series of related events in a compact manner, and journalists use them a lot. However, writing and maintaining such timelines, as well as building a comprehensive visualization, requires a considerable amount of human effort.

For this reason, automatic timeline summarization (TS) has known a wide interest in the last past years. TS is generally seen as a special case of multi-document summarization. For that matter, multi-document summarization systems have been used to generate timelines, and focus on the selection of the most representative sentences in an already time-stamped corpus [Yan *et al.*, 2011; Chieu and Lee, 2004; Tran *et al.*, 2015b]. Some previous work have focused on extracting salient dates before selecting the description of events corresponding to these dates [Tran *et al.*, 2013; 2015a; Kessler *et al.*, 2012; Nguyen *et al.*, 2014].

The final output of these systems is generally made of the k top ranked events, presented in chronological order. The visualization can then be obtained with traditional librairies such as TimelineJS, SIMILE, TimeGlider, vis.js (see Figure 1). The information concerning the rank and the weight of the events is only used for selecting the top k, and is then lost.

We argue that readable timelines (or chronologies) should present first an overview with the most important events, but also let the reader discover intermediate events at will. A timeline must certainly be followed along a temporal axis, but a feedback of the importance of the events should also be displayed.

In this paper, we follow [Nguyen *et al.*, 2014] and describe a system taking a set of keywords as an input, producing an output that is not a constrained summary or list of events, but a weighted list of dates, together with a description of the event that occurred at each date. We first focus on how the articles are processed in order to rank the dates, and especially on how the events are time-stamped. We show that considering only the article publication date does lead to shifted peaks, and then to irrelevant timelines. For this reason, we use a temporal normalization of texts to adjust the peaks. Then, we choose the best article headline related to the date and topic, as an event description.

Finally, we present a visualization tool specially dedicated to this system, where all the extracted events in the considered time span can be shown, and where the importance of the events is symbolized by a time-series graph filtered through event-specific smoothing functionalities.

The system is demonstrated on French data[1].

2 System

The first step of our approach can be seen as a task of "date extraction". Our system extracts a maximum of temporal information and uses only this information to detect salient dates for the construction of event timelines. Then, textual content is used for selecting the description of each event. Finally, an original data visualization is proposed.

2.1 Event Extraction

Figure 2 shows the general architecture of the system:

① The system Heideltime [Strötgen and Gertz, 2013; Moriceau and Tannier, 2014] is used to normalize temporal expressions in the texts. Absolute (e.g. "*January 6, 2016*") and relative dates (e.g. "*on Friday*") are turned into a `YYYY-MM-DD` common format (see examples in Figure 3). This allows us to link event to specific dates, instead of relying on the document creation time.

② The corpus is indexed by the Solr search engine[2], where one document per sentence is created, considering only

[1] This work has been partly founded by a Google award for computational journalism.

[2] http://lucene.apache.org/solr/

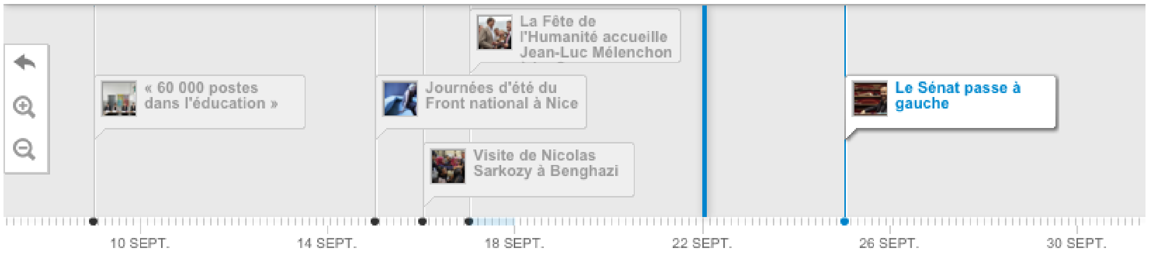

La Fête de l'Humanité accueille Jean-Luc Mélenchon
« 60 000 postes dans l'éducation »
Journées d'été du Front national à Nice
Visite de Nicolas Sarkozy à Benghazi
Le Sénat passe à gauche
10 SEPT.
14 SEPT.
18 SEPT.
22 SEPT.
26 SEPT.
30 SEPT.

Figure 1: An existing, manually produced timeline on the French presidential race (TimelineJS).

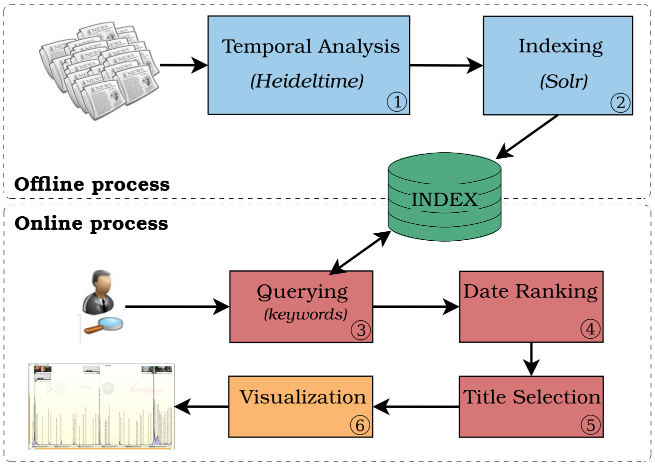


Figure 2: System overview.

sentences containing a normalized date. Each sentence is indexed together with the dates and the title of its article, using stemming.

③ At query time, documents are retrieved from the index, without any number limit.

④ Dates are extracted from the documents and weighted according to the number of occurrences of the date in the retrieved documents. Thus we obtain a plot where each peak corresponds to an "important" date. This is why considering dates inside the text instead of the document creation time is important: using document creation time gives us a measure of the mediatic response to events, making the peaks match with the days after the events. Retiming the events w.r.t. the dates specified in the text allows to reposition the peaks in front of the actual date of the events (see an illustration at Figure 4).

⑤ We then need to associate a textual description to each event. This is done by collecting the more important words for each date with a classical tf.idf:

$$tf.idf(w,d) = tf(w,d) \log \frac{N}{df(w)}$$

where $tf(w,d)$ is the frequency of word w in all sentences containing the date d, $df(w)$ is the frequency of word w in the entire corpus, and N is the total number of documents in the corpus. For each date d, the 20 words having the highest weight are used to query the Solr index again and to select the top article published at this date d. The description of the day event is then the headline and a picture (if any) of this article.

⑥ Visualization is described at next Section.

*At least 129 people died after a series of violent incidents around Paris, France, on **Friday 13 November 2015**.*

*The attacks in Paris on the night of **Friday 13 November** left 130 people dead and hundreds wounded.*

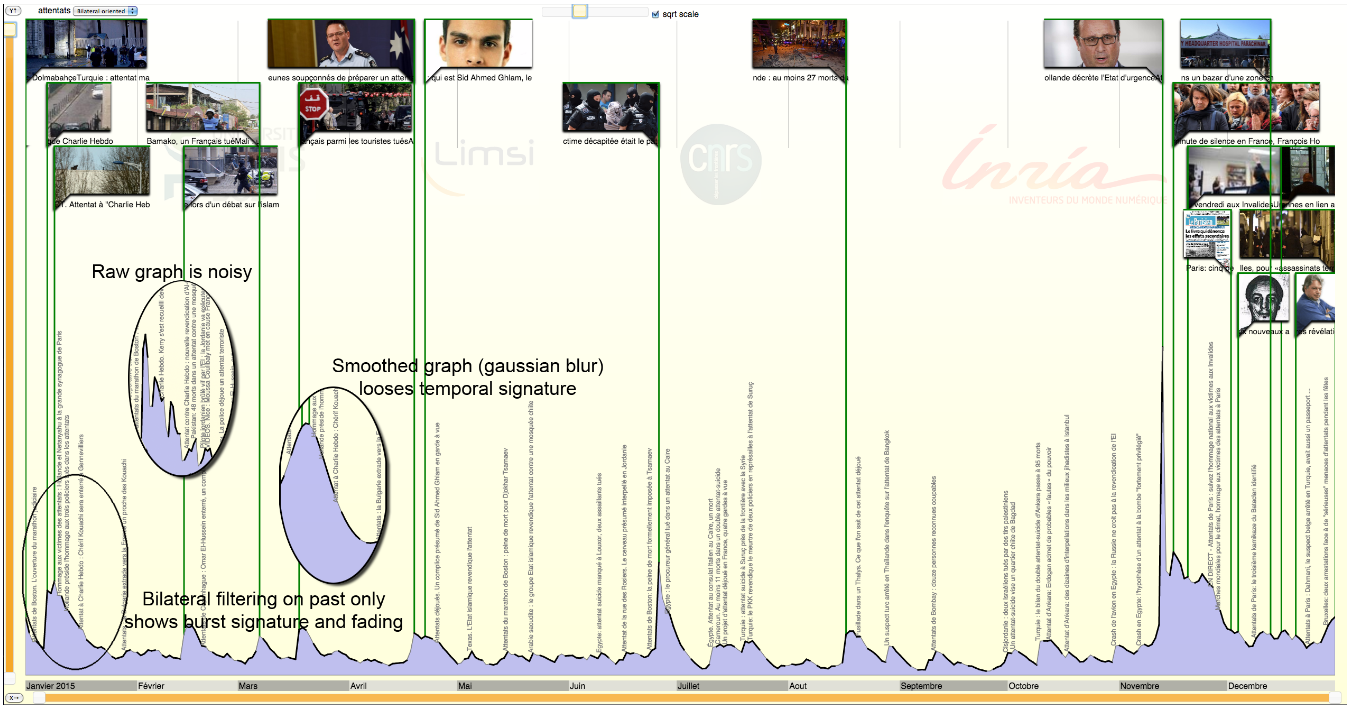
*At least 128 people were killed in shootings and explosions in Paris **late Friday***

*Attacks such as the one in Paris **three days ago** cannot obliterate our desire to understand*

Figure 3: Examples of sentences refering to November 13 events with absolute or relative dates. The normalized is "2015-11-13" for all the sentences.

2.2 Visualization Tool

Figure 5 shows an example of graph produced by the system. On top, the most relevant events are presented (headline and picture from the selected article for the specific day). At bottom, the graph is displayed along the same temporal axis. The graph represents the weights of each day as calculated at step ④. However, like all measures representing a human activity, these weights lead to a very noisy graph. We provide then a smoothing function to make the graph more readable to the user. A traditional Gaussian blur can be added and controled to obtain a smoother curve, but it also shifts the maxima to the right. Therefore, it would lose the temporal signature of the burst and decoy model (Descending Triangle Reversal [Hochheiser and Shneiderman, 2001]). In consequence, we added another smoothing functionality based on Bilateral Filtering [Paris and Durand, 2008] to preserve the discontinuity at event burst. However it does not match our burst and decoy model since days just before a burst are raised by the following days if decoy happens into the kernel size. We refined then the Bilateral Filtering by accounting only past events in the smoothing function. This function is then:



Raw graph is noisy

Smoothed graph (gaussian blur) looses temporal signature

Bilateral filtering on past only shows burst signature and fading

Figure 5: Visualization for the query “attentats” (“*attacks*”) in 2015.

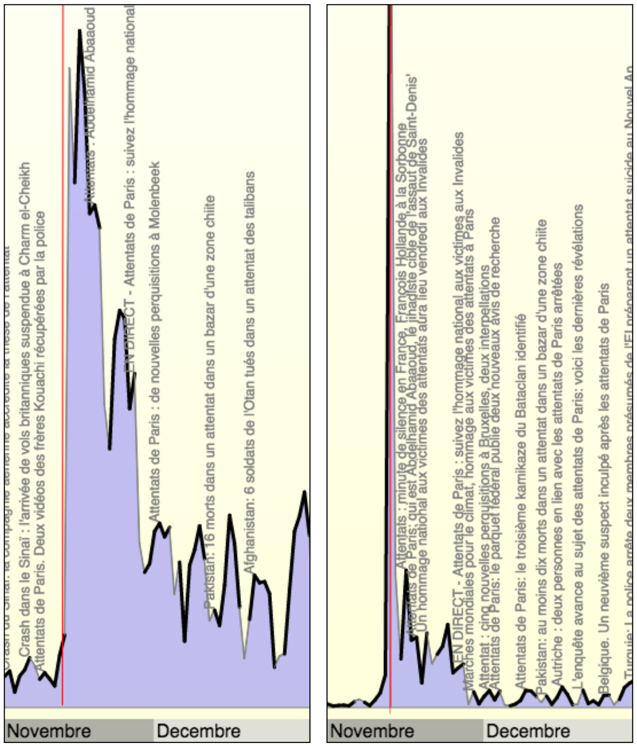


Figure 4: Raw graphs on query “attacks” in November and Decembre 2015, with time weights given by the document creation time (left) or by the temporal normalization (right). November 13 is represented by the red vertical line.

$$w'(d) = \frac{\sum_{i=-2\rho}^{0} w(d+i) \times e^{\frac{-i^2}{\rho^2}} \times e^{-\frac{(w(d)-w(d+i))^2}{\sigma^2}}}{\sum_{i=-2\rho}^{0} e^{\frac{-i^2}{\rho^2}} \times e^{-\frac{(w(d)-w(d+i))^2}{\sigma^2}}}$$

where $w(d)$ is the initial weight as described in previous section, d is the considered day, ρ an integer parameter and σ a real parameter. ρ and σ represent the extension of the neighborhood (ρ^2 is the variance of the Gaussian function) and can be modified by the user through sliders, for more or less smoothing. Default values are $\rho = 2$ and $\sigma = 0.1$. It produces the nicely readable graph of Figure 5 instead of the ones circled in the same Figure.

Even if a smoothing is necessary, we still aim at obtaining strong and sharp peaks when important events occur. Instead of using a pure burst model as in Kleinberg [2002] or Zhu and Shasha [2003], which have already been applied to media content [Xie *et al.*, 2013; Takahashi *et al.*, 2012], we prefer using our refined Bilateral Filtering with a decreasing threshold detection. We use the double gaussian Kernels of the Bilateral Filtering as the aggregate function F of the Shasha Model [Zhu and Shasha, 2003]. The article at highest burst is selected then removed from the time serie. This process is repeated until the timeline is filled with the targeted number of selected events.

Selected articles are displayed with both an image and the title of the article underneath. The system crops the image to a flag shape to highlight the temporal nature of events. The flag shape is attached to the graph with a line from the triangle. When two events happen very close to each other the flags can float on different sides of their pole (first event is displayed on the left to avoid overlapping). When more than two events occur very close we chose to display them arbitrarily on different tracks. As we process selected events from highest to lowest ranked, most important events are displayed on the top track and least important ones appear underneath. The smaller, not selected peaks display vertically the titles of the selected articles for these days.

The users of the system (i.e. journalists) can zoom on both axes by using the range sliders at bottom and at left of the combined graph. A temporal legend always display time ticks at bottom of the graph. The users can interactively rearrange events since the layout mechanism can fail to optimize screen real estate. Users can flip the flags on both side, change track (up or down) of an event and switch between two sizes (big or small). Furthermore, users can downgrade an event (and make it a smaller peak) or upgrade a smaller peak by double clicking on an vertical title above the graph. This makes the result fully editable, so that the journalist has the final say on what is displayed.

3 Discussion

3.1 Limits

The first important limit of the system is that its event granularity is fixed. This leads to two main issues: 1/ The tool is not able to detect more than one event per day. 2/ A macro-event that would last more than one day (e.g. a conference) could not be extracted nor vizualized.

Workarounds have been considered [Nguyen *et al.*, 2014] but tend to reduce the overall accuracy of the system. Our further work will focus on this issue.

The definition of an event "importance" is also open to question. In this paper we considered to the importance depends only on repetition. Other factors have been studied and applied with learning-to-rank approaches [Kessler *et al.*, 2012], and should be integrated into this system.

Finally, the process requires a large number of search engine queries, which makes it time-consuming. A first query returns a potentially high number of documents; then, one query per day in the time span is run to select the best article. Even if they can easily be parallelized, all these queries make the entire process quite heavy[3].

3.2 Adaptation to Other Languages

This study has been achieved on a French dataset. Only two steps are language-dependent and need little adaption to another language:

- Tokenization and stemming (widely available in many languages)
- Temporal normalization. Heideltime is available in 13 languages at the time of writing, and other tools may be existing for other languages.

The next step that is now being conducted consists in an evaluation with our journalist partners, and considers both the accuracy of the timeline and the ergonomics.

[3] Within a single thread on a simple server, about one minute for a one-year query on a popular subject as "attacks".

References

[Chieu and Lee, 2004] Hai Leong Chieu and Yoong Keok Lee. Query based event extraction along a timeline. In *Proceedings of the 27th ACM SIGIR conference*, 2004.

[Hochheiser and Shneiderman, 2001] H. Hochheiser and B. Shneiderman. Visual Specification of Queries for Finding Patterns in Time-Series Data. Technical Report CS-TR-4326, University of Maryland, 2001.

[Kessler *et al.*, 2012] Rémy Kessler, Xavier Tannier, Caroline Hagège, Véronique Moriceau, and André Bittar. Finding Salient Dates for Building Thematic Timelines. In *Proceedings of the 50th Annual Meeting of the ACL*, 2012.

[Kleinberg, 2002] J. Kleinberg. Bursty and Hierarchical Structure in Streams. In *Proceedings of the 8th ACM SIGKDD Conference*, 2002.

[Moriceau and Tannier, 2014] Véronique Moriceau and Xavier Tannier. French Resources for Extraction and Normalization of Temporal Expressions with HeidelTime. In *Proceedings of the 9th LREC Conference*, 2014.

[Nguyen *et al.*, 2014] Kiem-Hieu Nguyen, Xavier Tannier, and Véronique Moriceau. Ranking Multidocument Event Descriptions for Building Thematic Timelines. In *Proceedings of the 30th Coling Conference*, 2014.

[Paris and Durand, 2008] Kornprobst P. Tumblin J. Paris, S. and F. Durand. A Gentle Introduction to Bilateral Filtering and its Applications. In *Proceedings of the 42nd International SIGGRAPH Conference*, 2008.

[Strötgen and Gertz, 2013] Jannik Strötgen and Michael Gertz. Multilingual and Cross-domain Temporal Tagging. *Language Resources and Evaluation*, 47(2), 2013.

[Takahashi *et al.*, 2012] Y. Takahashi, T. Utsuro, M. Yoshioka, N. Kando, T. Fukuhara, H. Nakagawa, and Kiyota Y. Applying a Burst Model to Detect Bursty Topics in a Topic Model. In *Proceedings of JapTAL*, 2012.

[Tran *et al.*, 2013] G. Tran, M. Alrifai, and D. Q. Nguyen. Predicting Relevant News Events for Timeline Summaries. In *Proceedings of WWW Conference*, 2013.

[Tran *et al.*, 2015a] G. Tran, E. Herder, and K. Markert. Joint Graphical Models for Date Selection in Timeline Summarization. In *Proceedings of the 53rd ACL*, 2015.

[Tran *et al.*, 2015b] Giang Tran, Mohammad Alrifai, and Eelco Herder. Timeline Summarization from Relevant Headlines. In *Proceedings of the 37th ECIR*, 2015.

[Xie *et al.*, 2013] F. Xie, W.and Zhu, J. Jiang, and Lim E.P.and Wang K. TopicSketch: Real-time Bursty Topic Detection from Twitter. In *Proceedings of IEEE 13th ICDM*, 2013.

[Yan *et al.*, 2011] Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. Timeline Generation through Evolutionary Trans-Temporal Summarization. In *Proceedings of the 2011 EMNLP*, 2011.

[Zhu and Shasha, 2003] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *Proceedings of the Ninth ACM SIGKDD* , 2003.