StoryTracker: A Semantic-Oriented Tool for Automatic Tracking Events by Web Documents[*]

Welton Santos[1], Elverton Fazzion[1], Elisa Tuler[1],
Diego Dias[1], Marcelo Guimarães[2], and Leonardo Rocha[1]

[1] Universidade Federal de São João del-Rei, Brazil
{welton,elverton,etuler,diegodias,lcrocha}@ufsj.edu.br
[2] Universidade Federal de São Paulo/UNIFACCAMP, Brazil
marcelo.paiva@unifesp.br

Abstract. Media vehicles play an essential role in investigating events and keeping the public informed. Indirectly, logs of daily events made by newspapers and magazines have been built rich collections of data that can be used by lots of professionals such as economists, historians, and political scientists. However, exploring these logs with traditional search engines has become impractical for more demanding users. In this paper, we propose *StoryTracker*, a temporal exploration tool that helps users query news collections. We focus our efforts (i) to allow users to make queries by adding information from documents represented by *word embbedings* and (ii) to develop a strategy for retrieving temporal information to generate *timelines* and present them using a suitable interface for temporal exploration. We evaluated our solution using a real database of articles from a huge Brazilian newspaper and showed that our tool can trace different *timelines*, covering different subtopics of the same theme.

Keywords: Timelines · Search Engines · Word Embeddings

1 Introduction

Search engines' evolution (e.g., Google Searching, Microsoft Bing, DuckDuckGo) has increased access to information. Individuals and institutions feed these mechanisms daily, which indirectly document history through records of daily events. These tools are based on keyword queries that aim to retrieve information by associating the database's information with a few keywords. This type of query returns broad information about a topic and does not focus on details. For example, an economist who needs to find news and information about the stock market will provide some keywords (e.g., variations on the stock market price) to the search engine. The result will be the latest news about events that caused the stock prices to oscillate (e.g., the fusion between two companies) [21].

Despite being efficient in retrieving recent and more comprehensive information, traditional search engines still have limitations for users with more demanding goals looking for more specific and temporally related themes. For example,

[*] Supported by CAPES, CNPq, Finep, Fapesp and Fapemig

a historian interested in the stock's behavior market over time from a particular event (e.g., the Mariana dam collapse in Brazil in 2015 or Brazilian President Dilma Rousseff's impeachment in 2016) will require costly and not always workable work. Therefore, it will force him to perform multiple queries, sort and filter large volumes of data. Thus, from the perspective of more specific searches, performed by more demanding users such as historians, we observe that there are still two particular challenges during the search process: (i) generating good queries to retrieve relevant and more specific documents in databases and; (ii) temporally organizing large numbers of documents (e.g., news, articles) in such a way that navigation through these data is efficient [1].

The present work aims to provide a tool for more demanding users to easily search, organize, and explore documents by presenting the results through timelines. Our proposal consists of dividing a traditional query into two parts, as described below. First, we perform a traditional search, in which the user defines the keywords of the topic s/he wants to search. It uses a traditional search engine, presenting several documents related to the topic sorted according to some criterion predetermined by the search algorithm (i.e., often related to commercial issues). In the second part, which corresponds to this work's first contribution, among the various documents returned by the search machine, the user chooses the one s/he considers most relevant (reference). This reference document makes it possible to construct a timeline through the algorithm we proposed, called Story Tracker (ST). From an initial date (t_0) determined by the user, the temporal coverage for the timeline (e.g., five months) and the information related to the query and the selected document, the ST algorithm divides the temporal coverage into subspaces, e.g., weeks (e.g., t_1, t_2,...t_n) preserving the chronological order of the subspaces. For the time subspace t_0, the ST algorithm uses the representative vectors of the documents generated via *word embbedings* `doc2vec` model and identifies the documents most similar to the query and the reference document selected by the user, which will be presented for time t_1. Generally, to maintain consistency in the relationship between documents in an t_i interval with documents in the t_{i+1} interval, ST updates the reference document for each interval. This updated reference document in the t_{i+1} interval refers to the document most similar to query and the reference document in the t_i interval. This update process allows the user to force the tool to induce the retrieval of documents associated with more specific subjects addressed by the document and the query.

Our second contribution comprises a Web tool that uses the data retrieved by the process described above and presents them using an intuitive visual metaphor, which allows the user to explore the data in an organized way with convenience. Our tool organizes the time intervals in a sequence of sections (slides). Each section contains the documents from one interval to allow the user to explore the data after performing the query for building the timeline. The tool also provides document filters by categories (e.g., sports, market) to assist in the exploration process, which can be used in the originally retrieved data and in the timeline's construction, allowing specialized searches by categories. The tool also has a function that allows the user to define the number of documents he wants to view per interval. With this functionality, the user can generate more summarized and specific timelines, relating few documents per interval or more comprehensive timelines with a larger number of documents per interval.

We can integrate our tool with any search engine. However, to perform a controlled and consistent evaluation, we simulated a search engine using a database of the Brazilian newspaper Folha de São Paulo[3], with documents from January 2015 to September 2017, composed of 167 thousand documents of different categories. To instantiate our tool, we built a vector representation for each document in the database considering the *word embbedings* `doc2vec` model. We also built the vector representation of the user query with the *word embbedings* model. We then select the most similar documents to the user query using cosine distance between these vectorial representations. The N documents most similar to the query are presented to the user, and within the N documents returned by the simulated search machine, it must choose a reference document.

To evaluate the functioning of the proposed tool, we simulated a search for the terms "the impeachment of Dilma Rousseff and its impacts," choosing different reference documents and then detailing the timelines built from them. Our results show that by varying the reference document resulting from the original query, the proposed tool can trace different timelines, covering different sub-topics of the impeachment process (e.g., lava jato investigation, public manifestations). It would not be possible through a single query in a traditional search engine. Also, we analyzed the impact of the query and the reference document for the construction of the timelines. We observe that, as we increased the importance of the reference document to the query, we retrieved documents with topics more correlated to the reference document. This result is a strong sign that the reference document works well as an information source. To the best of our knowledge, we have found no work in the literature that aims to perform more specific and temporally related topic searches. Moreover, our proposal is generic enough that any current search engine can apply and use it.

2 Related works

2.1 Query Expansion

Query expansion is a strategy widely known in the literature and employed in search engines to optimize search results. Query expansion can change the user's original query by recalibrating the weight of terms or adding new terms. One of the main reasons for using query expansion is that users rarely construct good queries that are short on based on ambiguous words. Within the existing lines for query expansion, many works focus on identifying new terms from the analysis of an initial document ranking [4, 18, 7]. Basically, after an initial query of documents, the search engine returns a ranking of K documents, and the query expansion algorithm will use these documents to perform query enrichment. This approach is known as feature distribution of top-ranked documents (FD-TRD)

[3] `https://www.kaggle.com/marlesson/news-of-the-site-folhauol`

and assumes that the K documents in the initial ranking are good sources of correlated query information because of their proximity to the words.

We can classify FD-TRD application into two fronts, Relevance Feedback (RF) and Pseudo-Relevance Feedback (PRF) [5]. The difference between the two strands comprises the user interaction with the initial document ranking, returned from the original query. PRF-based techniques assume that the initial document ranking is consistent with the query and use it as a source of information for its expansion. RF techniques rely on user interaction with the ranking documents and use this interaction to learn about the user's interests [18]. The system can learn about the user's interests in two ways, implicit or explicit. In the implicit form, the system analyzes a user's interaction with documents (e.g., the opened ones). In the explicit form, the user voluntarily shows the system which documents are relevant to him, and therefore should be used as a source of extra information. PRF-based techniques are popular because of the convenience of not depending on the user to collect information. However, that is hardly available, being more susceptible to querying drift, adding irrelevant information to queries, which is not suitable for real-time systems.

Since this work aims at users interested in conducting further research, and the user's explicitness about which documents are important is available, our work closely resembles works that apply PR to query expansion. Among the works that present approaches with PR, new proposals explore the potential of *word embbedings* models. [19] presents a KNN-based approach, which computes the distance between words using representation vectors based on the Word2Vec model [16, 15]. In [24], the authors present that approaches with *word embbedings* trained on local data show better results than global approaches. Compared to other literature approaches, wembb-based techniques are equal or superior to traditional techniques (e.g., RM3) using RF [10]. Unlike all the work presented above, our approach is the first to add to the original user query the vector representations of documents.

2.2 Temporal Information Retrieval

Organizing and retrieving information in databases over time is a challenge addressed in several works [6, 2, 17, 13, 3, 8, 21, 14]. In [3], the authors conduct a systematic review of time-focused information retrieval work and present the main issues inherent in existing challenges in distinct lines (e.g., temporal clustering, timelines, and user interfaces). In [8], the authors review temporal information retrieval techniques and classify the existing challenges into temporal indexing and query processing, temporal query analysis, and time-aware ranking. In literature, we noticed efforts aimed at two research fronts: **(i)** to develop models for using time as a variable to enhance classification, clustering, and information retrieval techniques; and **(ii)** to create suitable visual metaphors for users to easily explore the data returned by information retrieval models.

Following the first front, works in the literature introduce the time parameter into various text processing tasks. In [2], the authors develop an add-on for reranking search results from web content search engines (e.g., Google Searching).

The authors process the data returned by the search engines and use the creation and update dates of the documents with temporal data present in the page's content to generate timelines in the users' browsers. Focusing on clustering and summarizing social network data, in [13], the authors propose a technique to extract and generate timelines from Twitter users. They develop a non-parametric topic modeling algorithm able to efficiently identifying tweets related to important events in users' lives (e.g., getting a university place) amid tweets with low importance (e.g., comment about a movie). With the model, the authors can trace ordinary users' timelines and more influential users (e.g., celebrities). In [6], the authors present a model for detecting patterns of behavior, called episodes, by analyzing social network user profiles. The authors propose a Bayesian model to generate summarized timelines with important episodes.

Some works direct their efforts toward presenting information to users. [14] presents the Time Explorer tool that explores graphical resources to generate intuitive timelines for users, aiming to generate timelines. The tool allows users to relate information extracted from the content of documents and make predictions about possible future events involving entities (e.g., celebrities, companies). More similar to our work, we found [21], which presents a topic coverage maximization tool focusing on generating timelines for historians. The authors argue that the way a historian seeks information differs from ordinary users. This type of user gets broad and general views of an event and not one-off information.

The works presented in this section focus on building timelines and exploratory research. Differently, we propose a complementary tool to traditional search engines. Our proposal is the only one focused on searching for more specific and temporally related topics with RF query expansion.

2.3 Document Representation

Many works largely use document vector representation in word processing because of its importance in document classification and clustering. Among the state-of-the-art works in the vector representation of documents, *word embbedings* models stand out. Traditional models based on Bag-Of-Words (BOW) have two inherent weaknesses in their representations: **(i)** loss of semantic information in the representations; and **(ii)** the vectors generated to represent documents have large dimensions. Usually, the amount of dimensions is equal to the size of the vocabulary of the document set.

Facing these problems, *word embbedings* models represent words in reduced vector spaces, preserving the semantics of words and documents. Among the existing works in *word embbedings*, [15, 16, 11] stand out. [15] proposes `word2vec`, a widely used model. The work proposes to use words in a multi-dimensional vector space so that a vector represents each word (usually 50 to 2000 thousand dimensions) in which semantically close words have similar vectors. Following the line of word representation, [11] extends word representation to sentence and document representation with the widely used `doc2vec` model.

Regarding document classification, [20] compares several representation methods for binary document classification based on data taken from clinical records to identify patients who are users of alternative medicines. In [12], the authors compare the doc2vec model with traditional representation models and with the word2vec model. They investigate the performance of the representation techniques for the task of classifying product categories from product descriptions. They show that traditional models perform similarly to word2vec, being better than doc2vec and suggesting that tasks, where semantic context preservation is not required *word embbeginds* algorithms do not promote improvement. In the paper [22], the authors use the doc2vec model and the HDBSCAN algorithm to identify duplicate use case tests and reduce software testing costs. Focusing on topic modeling, [9] uses vector representations with *words embbedings* to train classifiers and suggest to researchers topics for paper submission. Closer to our work, in [23] they train a doc2vec model with a Twitter database and propose a new technique to measure document distance in information retrieval tasks. Thus, although we have not found works that use *word embbedings* models to improve query results in search machines and organize them temporally, the excellent results in other areas motivated our proposal, detailed in the next section.

3 Story Tracker

In this section, we explain *Story Tracker* (ST), our proposal for performing query result refinements in search engines that aims to make the search easier, organize and explore documents by presenting the results using *timelines*. Figure 1 shows an overview of the tool, and we describe each of the steps next.

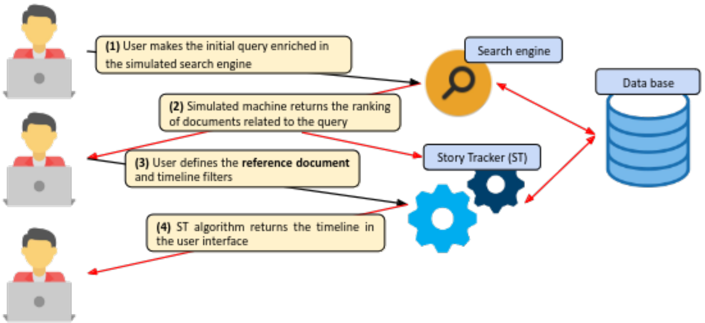

(1) User makes the initial query enriched in the simulated search engine
Search engine
Data base
(2) Simulated machine returns the ranking of documents related to the query
Story Tracker (ST)
(3) User defines the reference document and timeline filters
(4) ST algorithm returns the timeline in the user interface

Fig. 1. Overview of user interaction in the process of *timelines* generation

3.1 Enriching the Query

The ST's first component uses query expansion to build *timelines* about specific subjects with the documents' information. The technique developed here comprises a variation of the Relevance Feedback technique by user interaction. Our goal is to allow the user to create *timelines* based on specific subjects from simple *queries*, exploiting information from few documents selected by the user. To do so, we show to the user a *ranking* of documents related to a *query* before building the *timeline* from the results of a search. Among the documents in the *ranking*, the user chooses a document that is closer to his interest. This document will be used as a source of information to retrieve documents next to the *query* in *timelines* creation (Section 3.2). We can see the process of choosing the reference document in Figure 1 in steps one and two.

Given a search engine operating on a document base D relative to all indexed documents and a user-created *query* q with the keywords on the topic of interest, we construct a tuple (q, C_q, p_q, m_q), where C_q, p_q, m_q are also user-defined. C_q is a set of categories used for filtering where only documents ($d_i \in D$) with categories in C_q are retrieved ($Cat(d_i) \in C_q$). p_q consists of a base date for document exploration by our solution. Finally, m_q is a time period in months that is used as a search radius for base date p_q. From p_q and m_q, we define the interval $T_q = [(p_q - m_q), (p_q + m_q)[$. For example, for $p_q = 2016/04/01$ and $m_q = 2$, documents in the period $T_q = 2016/02/01$ to 2016/06/01 will be analyzed. Thus, the scope of documents to be analyzed by the ST is given by $E_q \in D$ (Eq. 1).

$$E_q = \{d_i \in D | d_i \in T_q \wedge Cat(d_i) \in C_q\} \tag{1}$$

In this way, the new *query* E_q is submitted to a search engine, which will return a set of documents D_q sorted by some criteria pre-defined by the search algorithm (i.e., often related to commercial issues). Then, the user chooses a reference document in D_q to be used as an information source in the *timeline* building process described in Section 3.2. For each of the D_q documents, the ST will create a vector representation using the *word embbedings* `doc2vec` model. Similarly, the linear combination of all the vectors of each word of query q will transform the query q into a vector representation. The similarity calculation of these vector representations of documents and *query* will create the *timeline*.

3.2 Timeline Construction

The focus of the *Story Tracker* algorithm is to explore the similarity between documents belonging to the result of a specific query published in closer periods (e.g., published in the same week). This proximity is due mainly to the reactive nature of publication vehicles such as newspapers and magazines, which, to keep their audience updated, constantly publish stories about sub-events of a broader event. As sub-events are the causes of new sub-events, the relationship between them tends to be stronger, as does the similarity between the content of the stories that record them. Based on this assumption, the core of the algorithm proposed here consists of correlating documents temporally. Given an initial query and a reference document, documents published in neighboring time intervals can be retrieved and become sources of information to retrieve other documents. The application of the ST algorithm on the database is related to Steps 3 and 4 of the example in Figure 1.

As input, the ST algorithm receives a tuple of parameters (q, d_r, D_q, T_q, s). d_r comprises the reference document chosen by the user, s refers to the granularity in days between the intervals that compose the *timeline* and q, D_q and

T_q are defined as before. From these parameters, the ST algorithm divides the construction of the *timeline* into two parts: past and future. Starting from the publication date of the reference document d_r ($Pub(d_r)$), the algorithm creates two sets of intervals (time subspaces) T_{pas} and T_{fut} referring to the past and future of $Pub(d_r)$, respectively. These interval sets comprise dividing the period T_q into T intervals of size and s in days. To generate the *timeline*, the expansion process described below applies to the interval sets T_{pas} and T_{fut} similarly. After an expansion of each interval set, the past and future outputs are concatenated and presented to the user as a *timeline*.

Expansion

The expansion process is presented in Algorithm 1. For each time interval $t \in T$, the ST algorithm creates subsets of documents S_t (lines 5-8) with the documents $d_t \in D_q$, whose publication date is within the t interval. For each document, its importance (semantic proximity) regarding the *query* and the reference document d_r (line 7) is calculated by the function $util(q, d_r, d_i)$ (Eq. 2). This function balances the importance between *query* q and reference document d_r, relative to any given document d_t through the parameter α, which varies in the range [0,1]. Higher α, the higher the importance of *query* and vice versa.

Algorithm 1: Story Tracker (ST) Expansion

```
1  Input: $(q, d_r, D_q, T_q, s)$
2  Output: Set S of documents for all intervals t
3  $T \leftarrow \frac{T_q}{s}$;
4  foreach interval $t \in T$ do
5      $S_t \leftarrow \{\}$;
6      foreach document $d_t$ where $((d_t \in D_q) \& (Pub(d_t) \in t))$ do
7          $d_t.util \leftarrow util(q, d_r, d_t)$;
8          $S_t \leftarrow S_t \cup d_t$
9      $sort\_util(S_t, descending)$;
10     $d_r \leftarrow MAX(S_t)$;
11     $S \leftarrow S \cup S_t$;
12 return $S$
```

We control the amount of noise added in a reference document (d_r) using the α parameter since it can address several correlated subjects. Thus, using all the document information, we may correlate more documents than expected by the user, configuring the effect of *query drifting*, present in *query* expansion systems, where irrelevant information is added to the initial *query*. This way, with higher values of α, the user can reduce the impact of possible noisy information and leave the results closer to the original *query*.

$$util(q, d_r, d_i) = \alpha.cos(q, d_t) + (1 - \alpha).cos(d_r, d_t) \quad (2)$$

Once the set S_t is built, we sorted it in descending order according to the value of the *util* function calculated for each of its documents (line 10). This set is concatenated to S, containing all the documents D_q sorted chronologically and by their relevance to the created timeline.

Updating the reference document

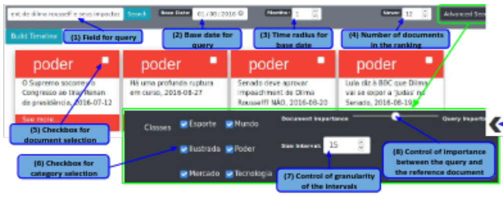
ST relies on a reference document to add information to the user's *query* and assist in retrieving subsets of documents that make up the *timeline*. Based on the assumption that documents in one interval tend to be more similar to documents in their neighboring intervals, ST updates the reference document for each subset of documents in each interval to search for more similar documents. Starting from the time interval t_0, ST retrieves the documents most closely related to *query* q and the reference document choose by the user, named d_{r0}. After constructing the subset of documents from the t_0 range, ST advances to the t_1 range. In constructing the subset of documents in this interval, the reference document d_{r0} is replaced by another document named d_{r1} (line 10), by means of the MAX function, which returns the most relevant document (*util* function) within the time t_0. We repeat this process for all remaining intervals so that the reference document used in the interval t_n comes from the interval t_{n-1}.

3.3 Graphical Interface

According to Figure 1, the *timeline* construction uses two separated steps: *query* enrichment and *timeline* construction. The enrichment of the *query* comprises the user's first contact with the tool, interacting with the interface presented in Figure 2. In this step, the user enriches the *query* by configuring the base date and the coverage of the *timeline* in Box 2 and Box 3, respectively. Figure 2 shows an example of the initial *ranking* with the cards containing the document title and a checkbox to mark the document as the reference document (Box 5). In Box 4, the user can define the number of documents in the initial *ranking*. In addition to the parameters present in the initial interface, the user can enrich their *query* with advanced parameters, found in the "Advanced Search" button, as highlighted in green in Figure 2. The set of categories for document filters (Box 6), the granularity of time intervals (Box 7), and the balance between the importance of the *query* and the reference document (Box 8) are configured in this advanced tab.

After selecting a reference document and considering all the documents returned by the query, ST builds the timeline and returns it to the user. The interface depicts in Figure 3. On this screen, the documents most associated with the *query* and the reference document in the interval are presented to the user. The title and date of the reference document are highlighted in blue in Box 1 and Box 2. The buttons used by the user to navigate through the *timeline* intervals are highlighted in Box 3 and Box 4. The interface offers another way to move through the *timeline* via the bottom bar (Box 6). Each link in the bar contains the title of the interval's reference document and a redirect function.[4]

[4] The code is available at `https://github.com/warSantos/StoryTracker.git`


Search
Base Date
01/30/2016
1
12
Advanced Search
Build Timeline
(1) Field for query
(2) Base date for query
(3) Time radius for base date
(4) Number of documents in the ranking
poder
poder
poder
poder
See more...
(5) Checkbox for document selection
(6) Checkbox for category selection
Classes
☑ Esporte
☑ Mundo
☑ Ilustrada
☑ Poder
☑ Mercado
☑ Tecnologia
Document Importance
Query Importance
Size Interval
15
(7) Control of granularity of the intervals
(8) Control of importance between the query and the reference document

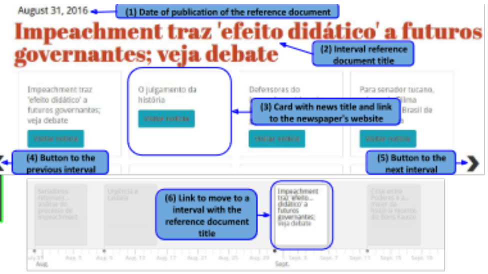
O Supremo socorreu o Congresso ao tirar Renan da presidência, 2016-07-12

Há uma profunda ruptura em curso, 2016-08-27

Senado deve aprovar impeachment de Dilma Rousseff NÃO, 2016-08-20

Lula diz à BBC que Dilma vai se expor a 'Judas' no Senado, 2016-08-19

Fig. 2. Screen for building the initial document *ranking*


August 31, 2016
(1) Date of publication of the reference document
(2) Interval reference document title
(3) Card with news title and link to the newspaper's website
(4) Button to the previous interval
(5) Button to the next interval
(6) Link to move to a interval with the reference document title
Visitar notícia
Visitar notícia
Visitar notícia
Visitar notícia
Aug.
Aug. 5
Aug. 9
Aug. 13
Aug. 17
Aug. 21
Aug. 25
Aug. 29
Sept.
Sept. 3
Sept. 7
Sept. 11
Sept. 15
Sept. 19

Impeachment traz 'efeito didático' a futuros governantes; veja debate

Impeachment traz 'efeito didático' a futuros governantes; veja debate

O julgamento da história

Defensoras do

Para senador tucano,

Fig. 3. Screen with documents from a *timeline* interval

4 Experimentation Evaluation

4.1 Experimental Setup

Models of *word embbedings* are famous for capturing semantic information between words and documents [16, 15, 11]. We explored the capacity of the `doc2vec` [5] model to find documents that are strongly correlated with each other. To train our models, we used the Folha de São Paulo Newspaper document base [6], which contains over 167,000 documents, from January 2015 to September 2017. Each document has the story title, textual content, publication date, category, and link to the page. We extracted the textual content of the document and removed the stopwords from each document. From this extraction, we trained a `doc2vec` representation model using the Paragraph-Vector Distributed Memory (PV-DM) algorithm, considering the default values for the algorithm parameters (100 dimensions, ten epochs, ...).

As described in the Section 3.1 and Section 3.2, our tool supports filtering documents by category. However, starting from the database without preprocessing, we found 43 different categories among the documents, with large overlapping topics and unbalance in the documents' volume among the categories. For this reason, we chose six main categories (sports, illustrated, market, world, power, and technology) before using a search engine. We redistributed the documents from the other categories into these six categories using a logistic regression model [7]. This process reduces the redundancy of categories and balances the volume of documents among them. For model training, we used the native documents from the main classes with the vector productions generated by the Paragraph-Vector Distributed Bag-of-Words (PV-DBOW) algorithm (i.e., using the algorithm's default parameters).

To evaluate our entire strategy in a more consistent and controlled way, we simulate a search engine over the database described in Section 4.1. Our search

[5] `https://radimrehurek.com/gensim/models/doc2vec.html`

[6] https://www.kaggle.com/marlesson/news-of-the-site-folhauol

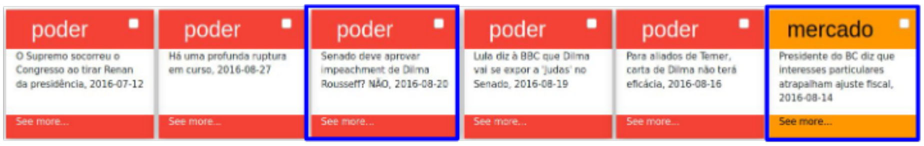
[7] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

engine operates on a subset of documents from the database defined by the scope of an enriched user *query* (Section 3.1). It presents to the user the *ranking* of the K documents closest to the user *query*. To estimate the similarity between documents and the *query* q, we first create the vector representation v_q of q as a linear combination of the vectors of the words of q. The vector of each word is extracted from the PV-DM model. Words not existing in the model are ignored in the calculation. Given the vector v_q, we measure the cosine similarity of v_q with all documents in the scope of the user-defined *query* and return the set of K documents with the highest similarity score to the *query*.

4.2 Results

In this section, we discuss the impact of the proposed tool on *timelines* construction. We analyzed our method's ability to build different *timelines* correlating documents from a single input *query*, varying the reference document and the α threshold. For constructing *timelines*, we used as a case study the impeachment process of Brazil's former president, Dilma Rousseff. We chose this topic because the impeachment process lasted a long time, generating large volumes of news.

In our search engine input, we used the *query "Dilma Rousseff's impeachment and its impacts"*, with a base date 2016/08/01 (one month before the impeachment date), time radius of one month (total of two months, one in the future and one in the past) and the granularity of the intervals of 15 days. As a set of categories, we considered *power, market* and *world*. On these parameters, the search engine returned the *ranking* of documents contained in Figure 4. The results are presented in portuguese as the original documents of our dataset, .



poder

O Supremo socorreu o Congresso ao tirar Renan da presidência, 2016-07-12

See more...

poder

Há uma profunda ruptura em curso, 2016-08-27

See more...

poder

Senado deve aprovar impeachment de Dilma Rousseff? NÃO, 2016-08-20

See more...

poder

Lula diz à BBC que Dilma vai se expor a 'Judas' no Senado, 2016-08-19

See more...

poder

Para aliados de Temer, carta de Dilma não terá eficácia, 2016-08-16

See more...

mercado

Presidente do BC diz que interesses particulares atrapalham ajuste fiscal, 2016-08-14

See more...

Fig. 4. *Ranking* generated by the search engine from the query "the impeachment of dilma rousseff and its impacts". In this Figure are the six documents most correlated to the user's query. The documents used as reference documents in the analyses in this section are highlighted in blue

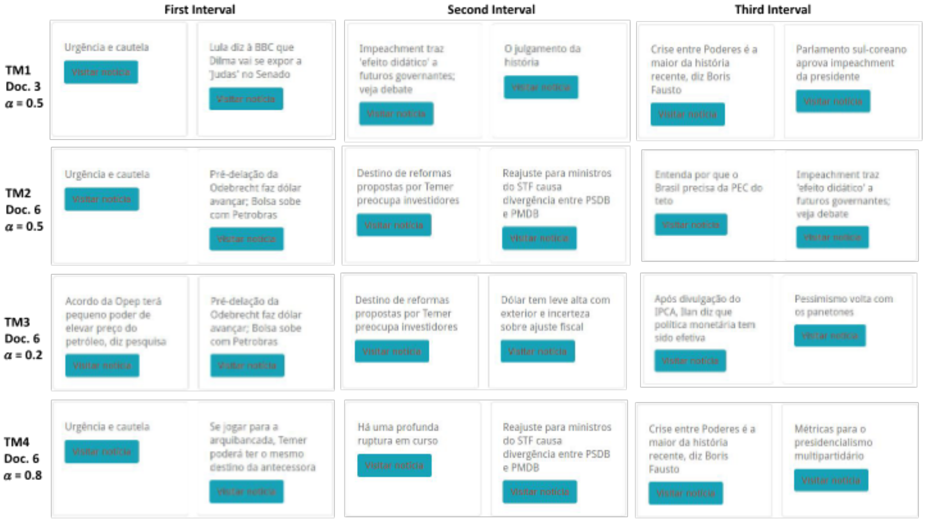
Given the *ranking* of news (Figure 4), we took for analysis the document three (D3)(*"Should the Senate approve the impeachment of Dilma Rousseff? NO"* and document six (D6)(*"BC President says private interests hinder fiscal adjustment"*) and built a *timeline* related to each document. We chose these documents for dealing with topics that share the *"impeachment Dilma Rousseff"* theme, with D3 having more emphasis on the *impeachment* process with political content and D6 with emphasis on the impacts of the *impeachment* on the Brazilian economy. As the basis of the discussion of this paper, we generated four *timelines*

TM1 and TM2 with $\alpha = 0.5$ for documents D3 and D6, respectively; and TM3 and TM4 for document D6 with $\alpha = \{0.2, 0.5\}$. For analysis, we selected the top two documents from three intervals of the *timelines* and analyzed the variations of these as the reference document and α threshold change as shown in Figure 5.

Figure 5 shows the documents from the TM1 *query* related to the D3. We observe that all the key documents of the three intervals deal with the main topic *impeachment*. Among the topics covered by the documents, we have themes such as "tax fraud", "corruption investigations", and "parliamentary coup". The themes share as their focus the process of *impeachment* using as background other processes involved. As an example, we can mention the second document in TM1 (*"Lula tells BBC that Dilma will expose herself to 'Judas' in the Senate"*). It deals with an interview of former president Lula to a newspaper, in which he defends himself against accusations regarding corruption involving his political party. Also, the third (*"Impeachment brings "didactic effect" to future leaders; see debate"*) and fourth (*"The judgment of history"*) documents deal with the event known as "pedaladas fiscais" (or pedaling fiscal). This event was used as an argument for the *impeachment* process. Comparing the documents in TM2 with those in TM1, it is noticeable that documents dealing with the *impeachment* process are still present. However, documents related to the economy show up as more relevant. As main examples of this change, we have the second (*"Odebrecht's pre-statement makes the dollar move forward; Stock Exchange goes up with Petrobras"*) and third (*"Fate of reforms proposed by Temer worries investors"*) documents in TM2. These documents deal with "variation in stock prices" and "worry of foreign investors" topics, respectively, which are topics that use the *impeachment* as background because of the economic instability caused in the country. Based on this analysis, we can see the impact caused by the alteration of the reference document, since with $\alpha = 0.5$ it was already possible to direct the construction of the *timeline* from the alteration of D3 to D6.

As described earlier, the second way to build different *timelines* is from the variation of the parameter α for the same tuple of *query* and reference document. Thus, the higher α, the greater the similarity of the documents to the *query* and vice versa. To analyze this parameter's impact, we use the D6 document to generate the T3 and T4 *timelines*. We chose this document to observe how much the variation of the α threshold distances and approximates the results of topics on economics. For testing, we set α to 0.2 and 0.8, so that the results are not radically dependent on the *query* or the reference document alone.

As shown in Figure 5, varying α from 0.5 (TM2) to 0.8 (TM4), almost all the documents with the main topic related to the economy were replaced, leaving only one, the fourth document (*"Readjustment for STF ministers causes a divergence between PSDB and PMDB"*) in T4. Although this document is slightly related to economics because it contains several economic terms (e.g., salary, unemployment, debt), reading this document, we noticed that it focuses on the discussions between political parties caused by inappropriate salary adjustments during the impeachment process. On the other hand, varying α from 0.5 (TM2) to 0.2 (TM3), we can see that all the main documents of the intervals have an



	First Interval		Second Interval		Third Interval	
TM1 Doc. 3 $\alpha = 0.5$	Urgência e cautela Visitar notícia	Lula diz à BBC que Dilma vai se expor a 'Judas' no Senado Visitar notícia	Impeachment traz 'efeito didático' a futuros governantes; veja debate Visitar notícia	O julgamento da história Visitar notícia	Crise entre Poderes é a maior da história recente, diz Boris Fausto Visitar notícia	Parlamento sul-coreano aprova impeachment da presidente Visitar notícia
TM2 Doc. 6 $\alpha = 0.5$	Urgência e cautela Visitar notícia	Pré-delação da Odebrecht faz dólar avançar; Bolsa sobe com Petrobras Visitar notícia	Destino de reformas propostas por Temer preocupa investidores Visitar notícia	Reajuste para ministros do STF causa divergência entre PSDB e PMDB Visitar notícia	Entenda por que o Brasil precisa da PEC do teto Visitar notícia	Impeachment traz 'efeito didático' a futuros governantes; veja debate Visitar notícia
TM3 Doc. 6 $\alpha = 0.2$	Acordo da Opep terá pequeno poder de elevar preço do petróleo, diz pesquisa Visitar notícia	Pré-delação da Odebrecht faz dólar avançar; Bolsa sobe com Petrobras Visitar notícia	Destino de reformas propostas por Temer preocupa investidores Visitar notícia	Dólar tem leve alta com exterior e incerteza sobre ajuste fiscal Visitar notícia	Após divulgação do IPCA, Ilan diz que política monetária tem sido efetiva Visitar notícia	Pessimismo volta com os panetones Visitar notícia
TM4 Doc. 6 $\alpha = 0.8$	Urgência e cautela Visitar notícia	Se jogar para a arquibancada, Temer poderá ter o mesmo destino da antecessora Visitar notícia	Há uma profunda ruptura em curso Visitar notícia	Reajuste para ministros do STF causa divergência entre PSDB e PMDB Visitar notícia	Crise entre Poderes é a maior da história recente, diz Boris Fausto Visitar notícia	Métricas para o presidencialismo multipartidário Visitar notícia

Fig. 5. Matrix with two most relevant documents by reference document and threshold intervals

economic focus, in general, on topics like ”stock prices”, ”inflation”, and ”GDP growth”. As a visible example of the favoring of economic topics tangential to the D6 document occasioned by the reduction of the α threshold, we can use the third (*”Odebrecht’s pre-statement makes the dollar move forward; Stock Exchange goes up with Petrobras”*) document in TM2. This document became the most important document in its range in TM3. This article addresses the economic impacts occasioned by the suspected crimes of the Brazilian company Odebrecht [8] and its association with the Brazilian oil company Petrobras, [9] Brazil’s leading state-owned oil company. Investigations into the company at the time of the story’s publication have caused distrust and oscillations in Petrobras’ stock price, considerable economic impacts, which justifies the document’s increased importance in its range.

Finally, comparing the main documents presented in TM3 with the documents in TM4, we can observe that both *timelines* do not share any main documents, even addressing constantly related topics. Observing the last document in TM3 (economic document), we observed it has no direct mention to the *impeachment* process. However, we noticed it brings related sentences, such as ”The Congress may be in turmoil, running from the police” and ”The public banks, stressed in the Dilma Rousseff years”, despite the threshold $\alpha = 0.2$ drastically reducing the *query* importance. This behavior shows to be successful in capturing semantic information between documents by our models, allowing our method

[8] https://en.wikipedia.org/wiki/Odebrecht
[9] https://en.wikipedia.org/wiki/Petrobras

to correlate tangent documents from implicitly shared information. It also shows us the efficiency of the α threshold in directing the construction of *timelines*.

5 Conclusion and Future Work

In this paper, we presented the StoryTracker tool, which can be integrated into any search engine so that users can easily organize and explore documents coming from a search by presenting the results through timelines. Our proposal is composed of three main parts: (1) Query Enrichment; (2) Timeline Construction; and (3) Summarization Interface. In the first part, we enrich the query by asking users to specify, besides the words, the document categories and a time interval. This query is then submitted to a search engine which returns several documents. The user chooses one s/he considers most relevant (reference document) to build a timeline. In the second part, our tool creates a vector representation using the *word embbedings* `doc2vec` model for all returned documents and the set of words that compose the query. The documents are then organized chronologically into different time subspaces. For each of these subspaces, we use the vectors representing the documents to identify the most similar documents to the query and the reference document, thus establishing a timeline. For the same set of documents returned by a search engine, the user can create different timelines, which present the same selected topic from different perspectives, varying the reference document. Finally, the third part comprises visual techniques to allow the user to explore the data.

To evaluate StoryTracker, we simulated a search engine using a database of articles published by the Brazilian newspaper Folha de São Paulo, from January 2015 to September 2017, composed of 167,000 documents. Simulating a search for the terms "the impeachment of Dilma Rousseff and its impacts", we build different timelines evaluating two different perspectives: political and economic. Our results show that our tool can trace different timelines using the original *query* and varying the reference document, covering different subtopics of the impeachment process (e.g., lava jato investigation, public manifestations). It would not be possible through a single *query* in a traditional search engine. As future work, our goal is to couple StoryTracker to different current search engines, performing an online evaluation of this combination under the usability perspective, considering different user profiles.

Acknowledgments

This project has partially supported by Huawei do Brasil Telecomunicações Ltda (Fundunesp Process 3123/2020), FAPEMIG, and CAPES.

Bibliography

[1] O. Alonso, M. Gertz, and R. Baeza-Yates. Clustering and exploring search results using timeline constructions. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 97–106, New York, NY, USA, 2009. Association for Computing Machinery.

[2] O. Alonso, M. Gertz, and R. Baeza-Yates. Clustering and exploring search results using timeline constructions. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 97–106, New York, NY, USA, 2009. Association for Computing Machinery.

[3] O. Alonso, J. Strötgen, R. Baeza-Yates, and M. Gertz. Temporal information retrieval: Challenges and opportunities. *TWAW Workshop, WWW 2011*, 707, 01 2011.

[4] R. Attar and A. S. Fraenkel. Local feedback in full-text retrieval systems. *J. ACM*, 24(3):397–417, July 1977.

[5] H. K. Azad and A. Deepak. Query expansion techniques for information retrieval: A survey. *Information Processing and Management*, 56(5):1698–1735, 2019.

[6] Y. Chang, J. Tang, D. Yin, M. Yamada, and Y. Liu. Timeline summarization from social media with life cycle models. In *IJCAI*, 2016.

[7] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. In *Information Processing and Management*, pages 779–840, 2000.

[8] N. Kanhabua and A. Anand. Temporal information retrieval. pages 1235–1238, 07 2016.

[9] P. Karvelis, D. Gavrilis, G. Georgoulas, and C. Stylios. Topic recommendation using doc2vec. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2018.

[10] S. Kuzi, A. Shtok, and O. Kurland. Query expansion using word embeddings. pages 1929–1932, 10 2016.

[11] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents, 2014.

[12] H. Lee and Y. Yoon. Engineering doc2vec for automatic classification of product descriptions on o2o applications. *Electronic Commerce Research*, 18:1–24, 08 2017.

[13] J. Li and C. Cardie. Timeline generation: Tracking individuals on twitter, 2014.

[14] M. Matthews, P. Tolchinsky, R. Blanco, J. Atserias, P. Mika, and H. Zaragoza. Searching through time in the new york times. 01 2010.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality, 2013.

[17] A. Qamra, B. Tseng, and E. Y. Chang. Mining blog stories using community-based and temporal clustering. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, page 58–67, New York, NY, USA, 2006. Association for Computing Machinery.

[18] J. Rocchio. relevance feedback in information retrieval. 1971.

[19] D. Roy, D. Paul, M. Mitra, and U. Garain. Using word embeddings for automatic query expansion, 2016.

[20] Y. Shao, S. Taylor, N. Marshall, C. Morioka, and Q. Zeng-Treitler. Clinical text classification with word embedding features vs. bag-of-words features. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2874–2878, 2018.

[21] J. Singh, W. Nejdl, and A. Anand. History by diversity. *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval - CHIIR '16*, 2016.

[22] S. Tahvili, L. Hatvani, M. Felderer, W. Afzal, and M. Bohlin. Automated functional dependency detection between test cases using doc2vec and clustering. 04 2019.

[23] L. Trieu, H. Tran, and M.-T. Tran. News classification from social media using twitter-based doc2vec model and automatic query expansion. pages 460–467, 12 2017.

[24] Y. Wang, H. Huang, and C. Feng. Query expansion with local conceptual word embeddings in microblog retrieval. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2019.