Bachelor thesis

Event Detection from Text Data

Tomáš Kala

Supervisor: doc. Ing. Jiří Kléma, PhD.

Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
May, 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student:	Tomáš K a l a
Study programme:	Open Informatics
Specialisation:	Computer and Information Science
Title of Bachelor Project:	Event Detection from Text Data

Guidelines:

1. Get familiar with the topic of event detection from potentially large text collections.
2. Reimplement the method of He et al. in Python and test it on a dataset provided by the thesis supervisor.
3. Propose and implement modifications of this algorithm. Consider changes in the cost function, utilization of document clustering with consequent topic dependent event detection and application of word/document embedding.
4. Extend the algorithm to be able to annotate the individual events and organize them.
5. Compare the results reached in the previous steps. Employ the list of real events given by the thesis supervisor to make the comparison as objective as possible.

Bibliography/Sources:

[1] He, Qi, Kuiyu Chang, and Ee-Peng Lim. "Analyzing feature trajectories for event detection." Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007.
[2] Fung, Gabriel Pui Cheong, et al. "Parameter free bursty events detection in text streams." Proceedings of the 31st international conference on Very large data bases. VLDB Endowment, 2005.
[3] Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. "Efficient Estimation of Word Representations in Vector Space". In Proceedings of Workshop at ICLR, 2013.
[4] Zhong, Shi. "Efficient online spherical k-means clustering." In Proceedings of the IEEE International Joint Conference on Neural Networks, 2005., vol. 5, pp. 3180-3185, 2005.
[5] Atefeh, Farzindar, and Wael Khreich. "A survey of techniques for event detection in twitter." Computational Intelligence 31.1, pp. 132-164, 2015.

Bachelor Project Supervisor: doc. Ing. Jiří Kléma, Ph.D.

Valid until: the end of the summer semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic	prof. Ing. Pavel Ripka, CSc.
Head of Department	**Dean**

Prague, January 12, 2017

Author statement for undergraduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....

signature

Abstract

Event detection is a process of analysis of text documents aiming to uncover real events happening in the world. It is based on the assumption that words appearing in similar documents and time windows are likely to concern the same real-world event. Therefore, our method attempts to group together words with similar temporal and semantic characteristics while discarding noisy words, not contributing to anything of interest. This results in a concise event representation through a set of representative keywords. These are then used to query the document collection to retrieve the actual event-related documents. Finally, we extract short summaries from these documents and annotate the events in a human-readable fashion. The keyword retrieval phase of our method is based on an existing event detection system, which we modify by employing a word embedding model to measure semantic similarity. The method is evaluated on a collection of 2 million documents from Czech news over a 13 months period and compared to the original method, not depending on word embeddings.

Keywords: Document retrieval, event detection, multi-document summarization, word embedding.

Abstrakt

Detekce událostí je proces analýzy textových dokumentů za účelem odhalení událostí, které se během doby jejich vydání staly ve světě. Tento proces je založen na předpokladu, že sémanticky podobná slova se zvýšeným výskytem během stejného období se pravděpodobně vztahují ke stejné události. Námi zkoumaná metoda se tedy snaží shlukovat dohromady slova s podobnou časovou nebo sémantickou charakteristikou, a zároveň ignorovat slova nenesoucí žádnou informaci. To vede k jednoduché reprezentaci událostí pomocí skupin klíčových slov. Tato klíčová slova jsou následně použita k dotazu do zkoumané kolekce a získání dokumentů vztahujících se k jednotlivým událostem. Z těchto dokumentů jsou nakonec extrahována krátká shrnutí pro bohatší popis událostí. Fáze získávání klíčových slov je založena na existujícím postupu, který modifikujeme použitím modelu vnořování slov (word embedding) k měření sémantické podobnosti. Metoda je vyhodnocena na kolekci 2 milionů dokumentů z českých novinových serverů vydané za období 13 měsíců, a porovnána s původním postupem nevyžadujícím vnořování slov.

Klíčová slova: Získávání dokumentů, detekce událostí, sumarizace více dokumentů, word embedding.

Contents

Chapter 1

Introduction

As the number of news articles published each day grows, it becomes impossible to manually examine them all to learn about events happening in the world. The field of Event Detection arose as a subfield of Information Retrieval (Rijsbergen, 1979; Manning et al., 2008) and Topic Detection and Tracking (Allan et al., 1998; Allan, 2002) with a goal to aid the users by automatically discovering important events in document collections.

More precisely, given a stream of text documents published over a certain time period, the task is to analyze them and output a collection of events that happened in the world during the period. An event is loosely defined as *something happening in a certain place at a certain time* (Yang et al., 1998).

In this thesis, we chose to modify an approach introduced by He et al. (2007a), which is a retrospective [1] method relying on event representation through keywords. These keywords are semantically related words with a similar temporal characteristic. The assumption is that related words frequently co-occurring during the same time period are representative of the same events that happened at that time.

We attempt to modify this method in various ways to obtain events of higher quality. We aim for a small number of events comprised of highly relevant keywords without any underlying noise. These events should have a clear interpretation, and not be redundant of each other. To achieve this, we introduce a word embedding-based measure of word similarity, which will be discussed in Chapter 2 and Chapter 3 in more detail.

Once we obtain the events represented in terms of keywords, we query the document collection to also obtain the documents related to the events. We then use these documents and keywords together to generate human-readable annotations that reveal more information about the events.

The rest of the thesis is organized as follows. First, in Chapter 2, we discuss related work. Then, in Chapter 3, we describe the document collection used for evaluation and the preprocessing steps taken.

In Chapter 4, we describe the original paper's procedure used to extract temporal characteristics of the individual words. These characteristics are then examined to reveal a subset of words which may be related to certain events, as opposed to generally appearing noisy words, so called *stopwords.*

Then, in Chapter 5, we proceed to the event detection itself. Here, we describe the original method, its modification relying on word embedding and also propose

[1] Discussed in Section 2.2

an alternative algorithm for event detection.

Although a set of related keywords provides a concise event representation, it is not particularly readable to the user. In Chapter 6, we follow by interpreting each keyword set as a query to the document collection. This allows us to employ Information Retrieval techniques to obtain documents relevant to each event.

Since the number of documents may be still too high, we also generate a short annotation for each event. The user can quickly skim through these annotations to get an idea what the events are about, and decide which of them are worth a closer examination. This will be addressed in Chapter 7.

Finally, we evaluate our method and compare it to the original paper in Chapter 8. We then conclude the thesis in Chapter 9.

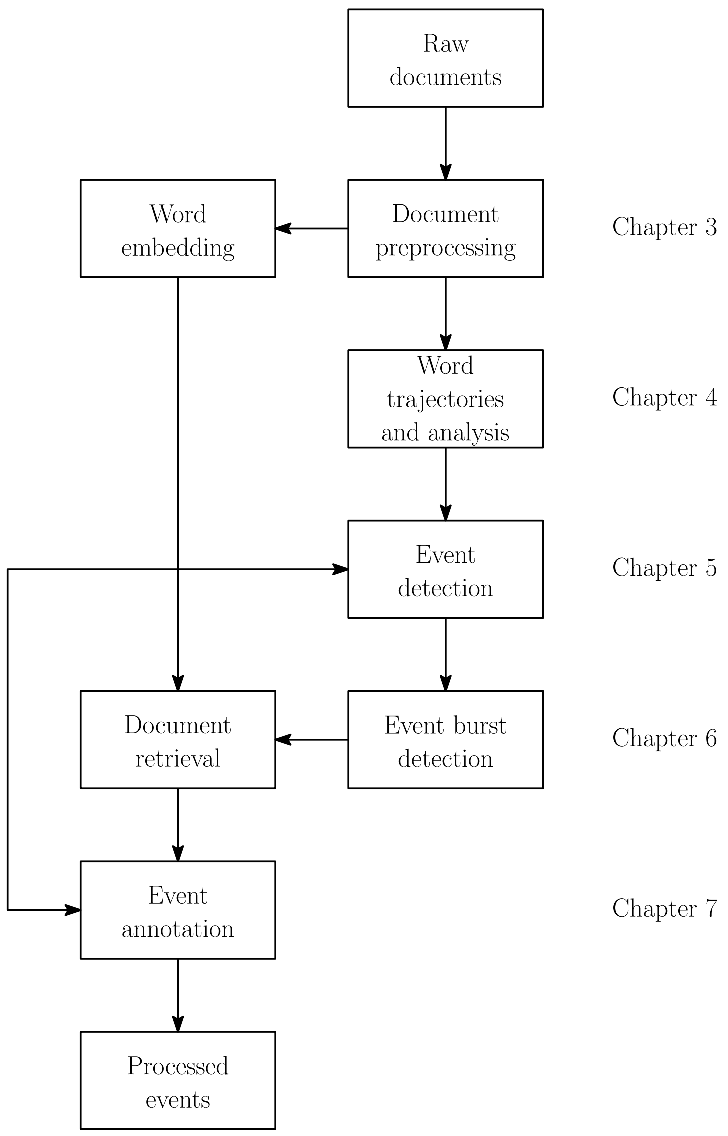


Figure 1.1: Schematic representation of our method.

Chapter 2

Related work

A system encompassing event detection, subsequent document retrieval and automatic event annotation needs to tackle several issues. In particular, we need to select a suitable word embedding model to be used during the detection. Furthermore, we must decide on the detection method itself and also specify the process of relevant documents retrieval. Finally, we need to find a suitable method of annotating the detected events in a human-readable fashion.

All of these concerns have been addressed in literature. Below, we provide a basic overview of the related work which was helpful for our approach.

2.1 Word embedding

Recently, a number of neural network models for vector space word embedding have been proposed. Perhaps the best known model is Word2Vec (Mikolov et al., 2013a) by Tomáš Mikolov. Additional methods include Stanford GloVe (Pennington et al., 2014), WordRank (Ji et al., 2015) and FastText (Bojanowski et al., 2016).

In this thesis, we use the Word2Vec model. The learned word vectors have useful semantical properties (Mikolov et al., 2013b,c), an efficient implementation exists (Řehůřek and Sojka, 2010), and it is a well documented and accepted method.

The Word2Vec model has additionally been modified to support embedding whole documents (Le and Mikolov, 2014).

2.2 Event detection

Although our method is evaluated on a news collection, the documents do not necessarily have to come from a formal news source. A lot of work has also been published in event detection by analyzing tweets, an overview can be found in Atefeh and Khreich (2015), other examples being Ifrim et al. (2014) and Brigadir et al. (2014). Atefeh and Khreich (2015) also distinguish between *retrospective* and *online* event detection. The former analyzes a given collection of documents to discover past events, the latter (also known as *First Story Detection*) tries to classify continuously incoming texts into “old” documents concerning events already known, and “new” documents concerning events not yet seen.

Further distinction can be made based on event representation. Some methods directly compare documents by their content and temporal similarity (He et al., 2007b), outputting an event as a set of documents. Others, such as Fung et al. (2005); He et al. (2007a); Fisichella et al. (2011) and our method included, represent the events by clusters of semantically and temporarily related keywords.

Additional work has also been done in event detection through topic modeling (Chaney et al.; Keane et al., 2015). Topic modeling will be briefly addressed in the next section.

2.3 Document retrieval

Retrieving relevant documents from a large corpus based on a user-given query is the main concern of Information Retrieval (Rijsbergen, 1979; Manning et al., 2008). A number of methods comparing similarity of document representation through vectors has been created. These methods range from a simple, yet precise binary weighting (Luhn, 1957; Salton and Buckley, 1988; Manning et al., 2008), to those utilizing term weighting to diminish common words (Sparck Jones, 1972) and approaches that attempt to discover a latent structure behind the documents, such as Latent Semantic Indexing (Deerwester et al., 1990).

Further work has been done in topic modeling, where the focus is to discover abstract topics behind the documents. Latent Semantic Indexing belongs to topic modeling as well. More complex methods, such as Latent Dirichlet Allocation (Blei et al., 2003) employ a generative probabilistic model to discover the topical structure. Document can then be compared in terms of their topical similarity.

Recently, a new similarity measure utilizing the Word2Vec model, an extension of Rubner et al. (2000), called Word Mover's Distance (Kusner et al., 2015) was introduced. This is a measure we are going to use and discuss in Chapter 6 in more detail.

2.4 Event annotation

For annotating the detected events, we consulted Gupta and Lehal (2010) and Nenkova and McKeown (2012). We aim to obtain a short summary for each event using the documents retrieved as relevant. The task of document summarization can be divided into *abstractive*, where the task is to generate new sentences or words not seen in the documents, and *extractive*, where the task is to extract parts of the document into a summary.

An example of the abstractive approach applied on news events is Alfonseca et al. (2013), extractive approach is addressed by e.g. Ferreira et al. (2014); Lin and Bilmes (2010, 2011). The abstractive methods are much more complex and still an active area of research, as it is necessary to generate sentences with a logical structure. We decided to employ the extractive approach, as the methods are generally better documented, simpler and more mature.

The method introduced by Lin and Bilmes (2010, 2011) supports multi-document summarization, which is suitable for our task, as we have multiple documents relevant to each event. Additionally, Kågebäck et al. (2014) examined various ways of how this approach could be improved by word embeddings. Their work led to a system presented in Mogren et al. (2015) which aggregates multiple similarity measures to perform summarization. We decided to adapt this system for our task.

Chapter 3

Document stream and preprocessing

The document collection we work with comes directly from webscraping various Czech news servers, and does not have any special structure. The documents consist only of headlines, bodies and publication days. Furthermore, there are some noisy words such as residual HTML entities, typos, words cut in the middle, etc. To make the most of the collection, we preprocess the documents to remove as many of these errors as possible, and also to gain some additional information about the text.

We first employ some NLP (Natural Language Processing) methods to gain insight into the data. Then, we train a model to obtain word embeddings, which we discuss next.

Our event detection method is keyword-based — the events will be represented by groups of keywords related in the temporal as well as semantic domain. To be able to measure the semantic similarity, we need to obtain a representation of the individual words that retains as much semantic information as possible while supporting similarity queries. There is a number of ways to do so — a simple TFIDF (Term Frequency-Inverse Document Frequency) representation (Sparck Jones, 1972; Manning et al., 2008) which represents the words by weighted counts of their appearance in the document collection. More complicated methods, such as Latent Semantic Indexing (Deerwester et al., 1990) attempt to discover latent structure within words to also reveal topical relations between them. This idea is further pursued by probabilistic topical models, such as Latent Dirichlet Allocation (Blei et al., 2003).

In this thesis, we use the Word2Vec model introduced by Mikolov et al. (2013a,b,c), which uses a shallow neural network to project the words from a predetermined vocabulary into a vector space. Vectors in this space have interesting semantical properties, such as vector arithmetics preserving semantic relations, or semantically related words forming clusters. A useful property of the Word2Vec model is that it supports online learning, meaning that the training can be stopped and resumed as needed. We can then train the model on one document collection, and only perform small updates when we receive new documents with different vocabulary.

Later on, we will need some sort of word similarity measure. This will come up several times in the course of the thesis — in the event detection itself, later when querying the document collection to obtain document representation of the events detected, and finally when generating human-readable summaries. The Word2Vec model is fit for all of these uses, as opposed to the other approaches mentioned above, some of which are designed only to measure document similarity, or, on the other hand, do not support document similarity queries very well.

3.1 Preprocessing

Some of the documents contain residual HTML entities from errors during web scraping, which we filter out using a manually constructed stopwords list.

We used the MorphoDiTa tagger (Straková et al., 2014) to perform tokenization, lemmatization and parts of speech tagging. Our whole analysis is applied to these lemmatized texts; we revert to the full forms only at the end when annotating the events in a human-readable way.

3.2 Word embeddings

Next, we train the previously mentioned Word2Vec model. Although the training is time-consuming [1], the word vectors can be pre-trained on a large document collection and then reused in following runs. In case the vocabulary used in these new documents differs, the model can be simply updated with the new words.

For the training, we only discard punctuation marks and words denoted as unknown parts of speech by the tagger. Such words are mostly typos not important for our analysis. We also discard words appearing in less than 10 documents.

The thesis was implemented using the Gensim (Řehůřek and Sojka, 2010) library. The project contains memory efficient, easy to use Python implementations of various topic modeling algorithms, Word2Vec included. In addition, we used the SciPy toolkit (Jones et al., 2001–) and Scikit-Learn (Pedregosa et al., 2011) for various machine learning-related computations.

We use the skip-gram model defined in Mikolov et al. (2013a), which was shown in Mikolov et al. (2013b) to learn high quality word embeddings well capturing semantic properties. After experimenting with different settings on a smaller subset of the documents, we decided to embed the words in a 100-dimensional vector space and to allow 5 passes over the document collection. Allowing more passes slows down the training, while not improving the quality very much. Setting higher dimensionality also does not lead to significant quality improvement, and slows down the training as well as requires more memory.

In the thesis, we refer to the vector embedding of a word w as $\boldsymbol{v}_w \in \mathbb{R}^{100}$.

3.3 Document collection

The dataset used is a collection of Czech news documents from various sources accumulated over a period from January 1, 2014 to January 31, 2015. The collection contains 2,078,774 documents averaging at 260 words each, with 2,058,316 unique word tokens in total. However, majority of the words are rare words or typos of no importance, so the number of unique real words is much lower. This is confirmed

[1]See Chapter 8 for computation times.

after discarding the words appearing in less than 10 documents, with only 351,136 unique words remaining.

These words are further processed in the following chapter, where we uncover a small subset of words possibly representative of an event, and discard the rest.

3.4 Document stream formally

Formally, the input to the algorithm is a collection of N news documents containing full text articles along with their publication days and headlines.

If we denote t_i as the publication day of a document d_i, the collection can be understood as a stream $\{(d_1, t_1), (d_2, t_2), \ldots, (d_N, t_N)\}$ with $t_i \leq t_j$ for $i < j$. Furthermore, we define T to be the length of the stream (in days), and we normalize the document publication days to be relative to the document stream start; that is $t_1 = 1$ and $t_N = T$.

Chapter 4

Word-level analysis

Word-level analysis is the first phase of the event detection algorithm, focused on obtaining temporal characteristics of the individual words as well as a set of candidate words for event representation. We do not yet perform the actual event detection, which is addressed in the next chapter, but merely extract a subset of words carrying enough information to be considered representative.

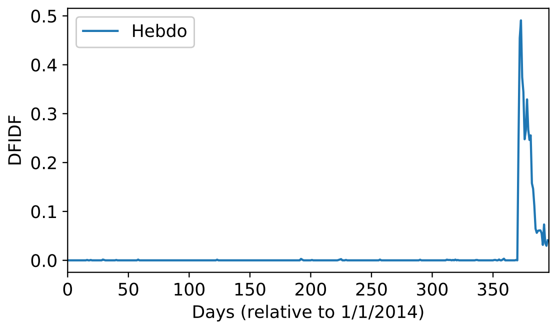
Here we work with the assumption that an event can be detected by observing the frequencies of individual words over time and grouping together those words which appear in similar documents during similar time periods (He et al., 2007a; Fung et al., 2005). This corresponds to an event being often mentioned in the text stream around the period when it actually occurred. Of course, not all words are representative of an event, so we will have to impose a criterion of a "word eventness".

He et al. (2007a) also distinguished between periodic and aperiodic words, where periodic words are mentioned with a certain period (these words are related for example to sport matches played every weekend, weather forecasts reported every day, etc.) Consequently, the authors divided the words into two groups by their periodicity, and detected events from each group separately. However, during our evaluation, some word periodicities were misclassified. This would cause an event represented by those words to be split into a "periodic part" and an "aperiodic part". Therefore, we detect events from all "eventful" words at once, and examine the periodicities of the events later on in Chapter 6.

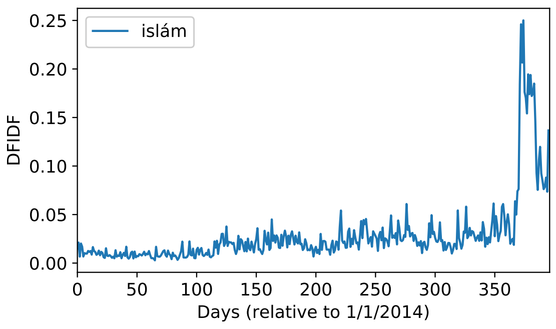
We show an example of such periodicity misclassification in Figure 4.1. There are two words representing the same real world event, the first of which in Figure 4.1a being correctly classified as aperiodic. The word in Figure 4.1b also has a distinct burst of activity typical for aperiodic words, though it is labeled periodic due to the noise present in its time trajectory. If we detected the events separately from the aperiodic and periodic categories, these two words could not be assigned together, even though they concern the same event.

At first, we construct a time trajectory of each word — a measure of word frequency over time. Then, we apply signal processing techniques to determine the eventness of each word. The same techniques will be later used to determine the event periodicities in Chapter 6. Once we have a notion of word eventness, we extract a small subset of words to be considered for further analysis, and discard the rest.

These word trajectories will then be examined for so called "bursts" in frequency,



(a) Aperiodic word



(b) Aperiodic word misclassified as periodic

Figure 4.1: Trajectories of the words *Hebdo* and *Islam*, respectively. Both of these words are related to the shooting in Charlie Hebdo offices in Paris on January 7, 2015. However, the original method classifies the word *Islam* as periodic with period of 198 days. If we detected events separately from periodic and aperiodic words, the event "Charlie Hebdo attack" would be split into at least two, causing redundancy.

where a word would suddenly start appearing in a large number of documents during a short time period. Should a number of words appear in similar documents with overlapping bursts, it may be an indicator that an event worthy of attention occurred.

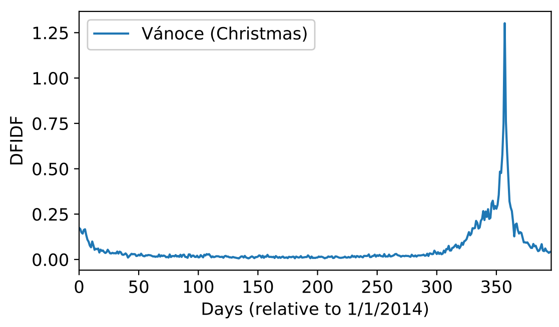
One thing to note is that the frequency of a word is, by itself, not a good indicator of a word importance. Stopwords appearing in most documents, such as conjunctions, prepositions, etc. do not carry any information and should be ignored. Therefore, we utilize the parts of speech tagging performed earlier and limit our analysis to Nouns, Verbs, Adjectives and Adverbs only. This limits the number of stopwords appearing in the stream, though some remain. These will need to be filtered by other means, as we will see later in this chapter.

We compare the trajectories of a typical eventful word and a stopword in Figure 4.2. The trajectory of the eventful word contains a distinct burst of activity, indicating a possible event happening. The stopword, on the other hand, does not contain such bursts, its trajectory oscillating as the word is mentioned in many documents over time. Although there are eventful *periodic* words with multiple bursts as well, as in Figure 4.3a, their trajectories generally reach higher values than those of stopwords. This difference in values will be used to distinguish stopwords from eventful words in Section 4.3.

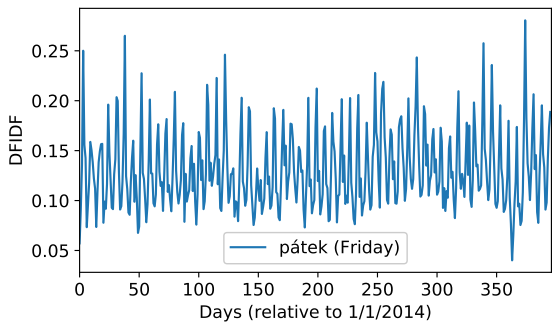
We ignore the documents and focus entirely on word analysis up until Chapter 6. There, we use the words assembled into events to query the document collection, obtaining the event-related documents. The core of the word analysis algorithm is taken from He et al. (2007a).

4.1 Binary bag of words model

To construct the word trajectories, we first need to know which words appear in which documents, as we are interested in the document frequency of each word. We create a binary bag of words model, which is represented by a binary matrix denoting the incidence of documents and words. This model completely ignores



(a) Eventful word



(b) Stopword

Figure 4.2: (a) Trajectory of an eventful word *Christmas*. (b) Trajectory of a stopword *Friday* with a period of 7 days.

word order, which is neglected in this analysis.

It might seem that using only a simple *binary* model, as opposed to one denoting, say, the frequency of words within the documents, results in a loss of information. While true, this model is not the target word representation — we only need to know the word-document incidence to construct the word trajectories, which are then the desired output.

We define a term-document matrix $\mathbf{B} \in \{0,1\}^{N \times V}$, where N is the number of documents and V is the total vocabulary size. The document collection can then be interpreted as a set of N observations, each consisting of V binary features. The matrix $\mathbf{B}$ is defined as

$$\mathbf{B}_{ij} = \begin{cases} 1, & \text{document } i \text{ contains the word } j; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

Because every document contains only a small fraction of the vocabulary, the matrix $\mathbf{B}$ consists mostly of zeroes. This allows us to store the matrix in a sparse format, which makes it possible to fit the matrix in memory. We use a sparse matrix instead of a more traditional inverted index (Manning et al., 2008), because this representation allows us to vectorize some further operations.

4.2 Computing word trajectories

He et al. (2007a) defined the time trajectory of a word w as a vector $\boldsymbol{y}_w = [y_w(1), y_w(2), \ldots, y_w(T)]$ with each element $y_w(t)$ being the relative frequency of w at time t ($1 \leq t \leq T,\ 1 \leq w \leq V$). This frequency is defined using the DFIDF (Document Frequency-Inverse Document Frequency) score:

$$y_w(t) = \underbrace{\frac{\mathrm{DF}_w(t)}{\mathrm{N}(t)}}_{\mathrm{DF}} \cdot \underbrace{\log \frac{N}{\mathrm{DF}_w}}_{\mathrm{IDF}}, \tag{4.2}$$

where $\mathrm{DF}_w(t)$ is the number of documents published on day t containing the word w (time-local document frequency), $\mathrm{N}(t)$ is the number of documents published on day t, N is the total number of documents and DF_w is the number of documents containing the word w (global document frequency).

The DFIDF score, defined by He et al. (2007a), is a modification of the commonly used TFIDF (Term Frequency-Inverse Document Frequency) score (Sparck Jones, 1972; Manning et al., 2008), which measures the importance of a word within a document collection. The purpose of this modification is to include temporal information and measure the word importance over time.

To be able to compute these word trajectories efficiently using the matrix $\mathbf{B}$, we define a utility matrix $\mathbf{D} \in \{0,1\}^{N \times T}$ mapping the documents to their publication days:

$$\mathbf{D}_{ij} = \begin{cases} 1, & \text{document } i \text{ was published on day } j; \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

Next, we sum the rows of $\mathbf{B}$ together to obtain $\boldsymbol{df} = [\mathrm{DF}_1, \mathrm{DF}_2, \ldots, \mathrm{DF}_V]$; $\mathrm{DF}_j = \sum_{i=1}^{N} \mathbf{B}_{ij}$. Similarly, we sum the rows of $\mathbf{D}$ to obtain $\boldsymbol{n}_t = [\mathrm{N}(1), \mathrm{N}(2), \ldots, \mathrm{N}(T)]$; $\mathrm{N}(j) = \sum_{i=1}^{N} \mathbf{D}_{ij}$.

Finally, we compute the word trajectory matrix $\mathbf{Y} \in \mathbb{R}^{V \times T}$, with trajectory of a word w, $\boldsymbol{y}_w \in \mathbb{R}^T$, being the w-th row of $\mathbf{Y}$.

The matrix $\mathbf{Y}$ is computed as follows:

$$\mathbf{Y} = \underbrace{\operatorname{diag}\left(\log \frac{N}{\boldsymbol{df}}\right)}_{\text{IDF}} \cdot \underbrace{\mathbf{B}^{\mathrm{T}} \cdot \mathbf{D} \cdot \operatorname{diag}\left(\frac{1}{\boldsymbol{n}_t}\right)}_{\text{DF}} \tag{4.4}$$

Now, having trained the Word2Vec model in Chapter 3 and constructed the word trajectories, we obtained temporal and semantic representation of the words. Every word w is represented by two vectors: $\boldsymbol{v}_w \in \mathbb{R}^{100}$ being its Word2Vec embedding, and $\boldsymbol{y}_w \in \mathbb{R}^T$ its time trajectory. The trajectories will be further analyzed in this chapter, while both trajectories and Word2Vec embeddings will be used in Chapter 5 to group words into events.

4.3 Spectral analysis

Having constructed the word trajectories, we still need to decide which words are eventful enough. He et al. (2007a) interpreted the word trajectories as time signals, which allowed them to analyze the trajectories using signal processing techniques. They performed the analysis both to decide word eventness and to discover the word periodicity.

Unlike the original paper, we only analyze the signal power to decide which words are eventful enough. We will detect events from both periodic and aperiodic words at once, and decide the periodicity of the whole events in Chapter 6.

We apply the discrete Fourier transform to the trajectories to represent each time series as a linear combination of T complex sinusoids. We obtain $\mathcal{F}\boldsymbol{y}_w = [X_1, X_2, \ldots, X_T]$ such that

$$X_k = \sum_{t=1}^{T} y_w(t) \exp\Big(-\frac{2\pi \mathrm{i}}{T}(k-1)t\Big), \ k = 1, 2, \ldots, T. \tag{4.5}$$

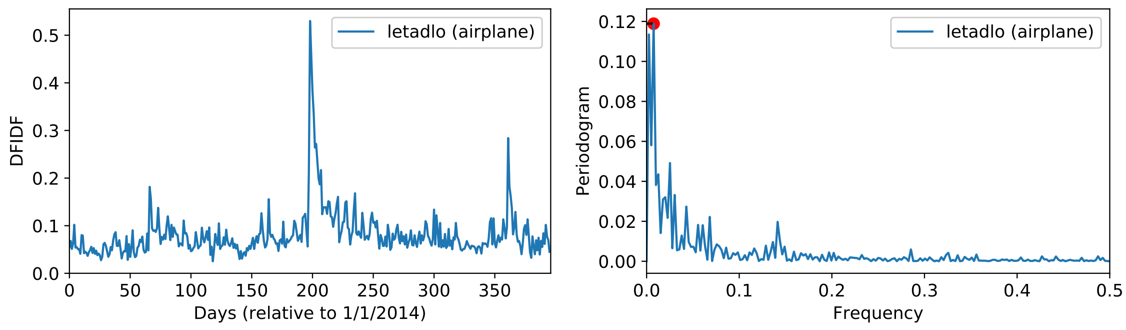
The measure of "eventness" of a word is simply its signal power. That can be determined from the power spectrum of each signal, estimated using the periodogram

$\boldsymbol{p} = \left[\|X_1\|^2, \|X_2\|^2, \ldots, \|X_{\lceil T/2 \rceil}\|^2\right]$.

To measure the overall signal power, we define the dominant power spectrum of the word w as the value of the highest peak in the periodogram, that is

$$\mathrm{DPS}_w = \max_{k \leq \lceil T/2 \rceil} \|X_k\|^2. \tag{4.6}$$

In Figure 4.3, we show the trajectory and periodogram of a periodic word *airplane*. In the periodogram plot, the DPS value of approximately 0.1192 is highlighted. This value is attained at frequency of about 0.0076, making the dominant period of the word $1/0.0076 \approx 132$. Although He et al. (2007a) further utilized these dominant periods to categorize the words by their periodicities, we postpone the periodicity analysis to event trajectories assembled in Chapter 6.



(a) Trajectory

(b) Periodogram with indicated DPS

Figure 4.3: Trajectory and periodogram of the word *airplane* with a period of 132 days.

Finally, He et al. (2007a) define the set of all eventful words (EW) as those words whose trajectory signal is powerful enough. This corresponds to their occurrence in a large number of documents in a noiseless pattern:

$$\mathrm{EW} = \left\{ w \mid \mathrm{DPS}_w > \textit{DPS-bound} \right\}. \tag{4.7}$$

where *DPS-bound* can be estimated using the *Heuristic stopword detection* algorithm described in He et al. (2007a). The algorithm computes the average trajectory value and DPS values from a given seed stopwords set. The DPS boundary is then defined as the maximum DPS value of the stopwords set.

Chapter 5

Event detection algorithms

In this chapter, we define the actual event detection algorithm. First, we describe the original method used by He et al. (2007a). Then, we make a change to incorporate semantic similarity through the word embeddings obtained in Chapter 3. Finally, we introduce an alternative algorithm that utilizes word clustering using a custom distance function.

The original algorithm creates events as sets of related keywords by greedily minimizing a cost function combining temporal and semantical distance between words. However, the paper used only a simple notion of semantical distance, namely the document overlap between words. This demands that there exists at least one document containing all the words used to represent an event. This is a strong requirement, since the documents may use different vocabularies while conveying similar information. As a result, the events are split into multiple keyword sets, leading to redundancy.

In an attempt to solve this problem, we modify the cost function, replacing the document overlap by a Word2Vec-based similarity. This does not require the words to appear in exactly the same documents, only that they have similar semantics. We refer to this method as *embedded greedy approach*, as it is a modification to make the original greedy algorithm utilize word embeddings.

Realizing that the task of constructing keyword sets resembles the task of word clustering, we propose an alternative algorithm. Here, we apply a clustering algorithm to the words, using a modification of the cost function as a distance measure. This is a method referred to as *cluster-based approach*.

First, we briefly describe the original method for reference. This will make it clear which parts of the function we modify. It will also allow us to make reference to the original method in Chapter 8, where we compare all three algorithms.

5.1 Original method

As stated in the introduction, the original method performs greedy minimization of a cost function defined over sets of words. The cost function consists of trajectory distance measuring the word distance in temporal domain, and document overlap, standing for distance in the semantic domain. We first describe these two components and then combine them into the cost function.

5.1.1 Trajectory distance

Before measuring the trajectory distance, the trajectories are smoothened by fitting a probability density function to them. We adapt a similar technique in Chapter 6 where it is described in more detail. Our event detection modifications do not use the original smoothing though, and we refer the reader to the original paper for more details.

After normalization to unit sum, the (smoothened) trajectory $\boldsymbol{y'}_w$ of a word w can be interpreted as a probability distribution over the stream days. The element $y'_w(i)$ then denotes the probability that w appears in a random document published on day i. This interpretation allows to compare the trajectories using information-theoretic techniques, notably the information divergence (Cover and Thomas, 2012).

In the original paper, the authors first defined the distance between trajectories of two words v and w as $\mathrm{Dist}(\boldsymbol{y'}_v, \boldsymbol{y'}_w) = \max\{\mathrm{KL}(\boldsymbol{y'}_v \| \boldsymbol{y'}_w), \mathrm{KL}(\boldsymbol{y'}_w \| \boldsymbol{y'}_v)\}$, symmetrizing the Kullback-Leibler divergence KL (Kullback and Leibler, 1951).

Then, the distance is generalized to a whole set of words M as

$$\mathrm{Dist}(\mathrm{M}) = \max_{v,w \in \mathrm{M}} \mathrm{Dist}(\boldsymbol{y'}_v, \boldsymbol{y'}_w). \tag{5.1}$$

5.1.2 Document overlap

The document overlap is again first defined for a pair of words v and w as $\mathrm{DO}\,(v, w) = \frac{|\mathrm{M}_v \cap \mathrm{M}_w|}{\min\{|\mathrm{M}_v|, |\mathrm{M}_w|\}}$, where $\mathrm{M}_j = \{i \mid \mathbf{B}_{ij} = 1\}$ is the set of all documents containing the word j. The higher the document overlap, the more documents do the two words have in common, which makes them more likely to be correlated.

The overlap is again generalized to a set of words M as

$$\mathrm{DO}(\mathrm{M}) = \min_{v,w \in \mathrm{M}} \mathrm{DO}(v, w). \tag{5.2}$$

5.1.3 Cost function

The cost function is a combination of the trajectory distance and document overlap of a set of words M. It is defined as

$$\mathrm{C}(\mathrm{M}) = \frac{\mathrm{Dist}(\mathrm{M})}{\mathrm{DO}(\mathrm{M}) \cdot \sum_{w \in \mathrm{M}} \mathrm{DPS}_w}. \tag{5.3}$$

Since the algorithm attempts to minimize it, the intuitive result is a set of words with low trajectory distance and high document overlap. The algorithm will also prefer words of higher importance due to the last term of the denominator, counting in the power spectra.

5.1.4 Event detection algorithm

The algorithm, called *unsupervised greedy event detection algorithm* in the original paper, produces events as structured objects e consisting of:

- $e.KW$: The event keyword set.
- $e.Docs$: Documents concerning the event.

- *e.Bursts*: Bursty periods of the event.
- *e.DP*: Dominant period of the event.
- *e.Annotation*: Human-readable annotation of the event.

Only the event keyword set $e.KW$ is initialized in this section. The other fields will be properly defined and filled in the rest of the thesis.

The algorithm itself is defined as follows:

Algorithm 1 Unsupervised greedy event detection

```
Input: Word set EW obtained in Chapter 4, matrices $\mathbf{B}$ and $\mathbf{Y}$, word DPS
 1: Sort the words in descending DPS order: $\mathrm{DPS}_{w_1} \geq \cdots \geq \mathrm{DPS}_{w_{|\mathrm{EW}|}}$
 2: $k = 0$
 3: for each $w \in \mathrm{EW}$ do
 4:     $k = k + 1$
 5:     $e_k.KW = \{w\}$
 6:     $cost_{e_k} = \frac{1}{DPS_w}$
 7:     $\mathrm{EW} = \mathrm{EW} \setminus w$
 8:     while $\mathrm{EW} \neq \emptyset$ do
 9:         $m = \operatorname*{argmin}_m \mathrm{C}(e_k.KW \cup w_m)$
10:         if $\mathrm{C}(e_k.KW \cup w_m) < cost_{e_k}$ then
11:             $cost_{e_k} = \mathrm{C}(e_k.KW \cup w_m)$
12:             $e_k.KW = e_k.KW \cup w_m$
13:             $\mathrm{EW} = \mathrm{EW} \setminus w_m$
14:         else
15:             break
16:         end if
17:     end while
18: end for
Output: Events $\{e_1, \ldots, e_k\}$
```

The algorithm works by greedily minimizing the cost function (5.3). Once it is minimized, an event is produced, consisting of all the words found since last event.

The words are sorted in descending DPS order before entering the minimization loop, so that the most important words are processed first. This assures that the most eventful words are assigned together, without wasting them to appear with low quality words.

He et al. (2007a) did not provide the time complexity of the algorithm, which we attempt to estimate now. The execution time is dominated by the main loop on lines 3 through 18. The outer loop must execute $\mathcal{O}(|\mathrm{EW}|)$ times. In each of the iterations, the inner loop is executed at most $|\mathrm{EW}|$ times, making it $\mathcal{O}(|\mathrm{EW}|)$ as well. The argmin statement on line 9 must search through the whole remaining $|\mathrm{EW}|$ words, also making it run $\mathcal{O}(|\mathrm{EW}|)$ times.

If the number of eventful words is low enough, the pairwise trajectory distance and document overlap can be precomputed. This makes the cost function take $\mathcal{O}(|\mathrm{M}|^2)$ time when applied to a set M. If the distances are not precomputed, the cost function execution requires $\mathcal{O}(|\mathrm{M}|^3)$ time.

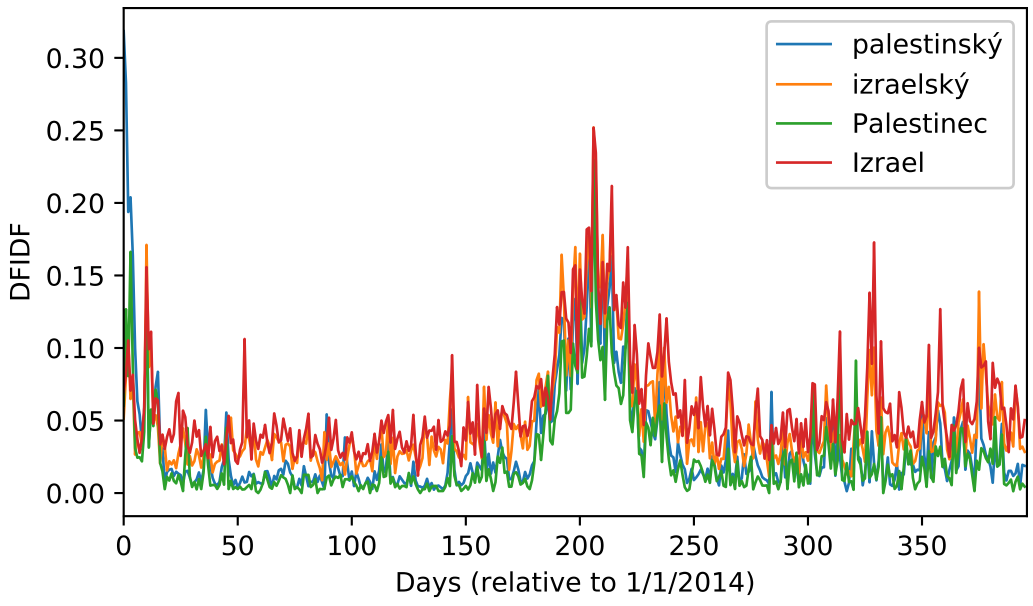


Figure 5.1: Example of an event detected using the original method. The event consists of the words *palestinian*, *israeli*, *Palestinian* and *Israel*, respectively.

We were unable to precisely determine the cost function's complexity with respect to the set EW, as it is always applied on the currently composed event. However, during our experiments, the number of words comprising an event never exceeded 10 in the original method. This makes the cost function's asymptotic complexity negligible compared to the main loop.

The resulting complexity of the algorithm is therefore $\mathcal{O}(|\mathrm{EW}|^3 \cdot c)$, where c is the complexity of the cost function.

In Figure 5.1, we show an example of an event detected by the original method. We can see that it consists of four keywords with overlapping trajectories, sharing a common burst of activity around day 210. The event is likely related to the tension in the Middle East, though the keywords by themselves do not allow any closer interpretation. For this reason, we provide longer annotations in Chapter 7, so that we can examine the event more closely.

5.2 Embedded greedy approach

In this section, we modify the original method to use the Word2Vec model to measure semantic similarity between words. Unlike the document overlap (5.2), this new similarity measure is able to distinguish semantically similar words even when they do not appear in the same documents. This may happen, for instance, when different authors each use distinct vocabulary when referring to the same event.

5.2.1 Semantic similarity

Some of the astounding results of the Word2Vec model arise from semantically similar words forming clusters (Mikolov et al., 2013c) in terms of cosine similarity, which is a standard measure used in information retrieval (Manning et al., 2008;

Huang, 2008).

We replace the document overlap in the cost function (5.3) by cosine similarity between Word2Vec embeddings, though with a small modification. The cosine similarity is bounded in $[-1, 1]$ with -1 denoting the least degree of similarity. This means that the cost function would reach negative values for highly dissimilar words. This would mean a problem, as Algorithm 1 attempts to minimize it. Consequently, we will transform the cosine similarity into $[0, 1]$, just like the document overlap (5.2).

The similarity between a set of words M and a word $w \notin \mathrm{M}$ is defined as

$$\mathrm{Sim}(\mathrm{M}, w) = \left(\frac{\langle \bar{\boldsymbol{v}}_{\mathrm{M}}, \boldsymbol{v}_w \rangle}{\|\bar{\boldsymbol{v}}_{\mathrm{M}}\| \cdot \|\boldsymbol{v}_w\|} + 1 \right) / \ 2, \tag{5.4}$$

where $\bar{\boldsymbol{v}}_{\mathrm{M}}$ is the mean of all vector embeddings of M and $\boldsymbol{v}_w$ is the vector embedding of w. Here, the mean vector virtually represents the central topic of words in M.

5.2.2 Cost function

We redefine the cost function (5.3) as

$$\mathrm{C}(\mathrm{M}, w) = \frac{\mathrm{Dist}(\mathrm{M} \cup w)}{\mathrm{Sim}(\mathrm{M}, w) \cdot \sum_{u \in \mathrm{M} \cup w} \mathrm{DPS}_u}, \tag{5.5}$$

where $\mathrm{Dist}(\cdot)$ is the original trajectory distance function (5.1).

In the original method, He et al. (2007a) defined the cost function (5.3) for a set of words. However, in Algorithm 1, it is always applied on the union of the keywords of an event constructed so far, and a newly added word. The new cost function must now be applied on such keyword set and word separately due to the nature of Word2Vec similarity definition.

Having constructed the cost function, we use Algorithm 1 to detect events once again.

In Figure 5.2, we show an event detected using the embedded greedy method. It is related to the same real-world event as the event in Figure 5.1, though it consists of more keywords. Generally, events detected by the embedded greedy method contain more keywords, as we will see in Table 8.1. The trajectory overlap is not perfect, the burst of the word *American* being off compared to the rest of the words. It is possible that the semantic similarity of the word was so high that the word was deemed relevant nonetheless.

5.3 Cluster-based approach

Realizing that the keyword-based event detection resembles word clustering and could be solved by an application of a clustering algorithm, we decided to investigate this idea. In the final method, we apply a clustering algorithm equipped with a custom distance function to the set of eventful words. The distance function is actually a modification of the cost function yet again, though some means have to be taken to make it usable in this context. First though, we need to consider a proper clustering algorithm.

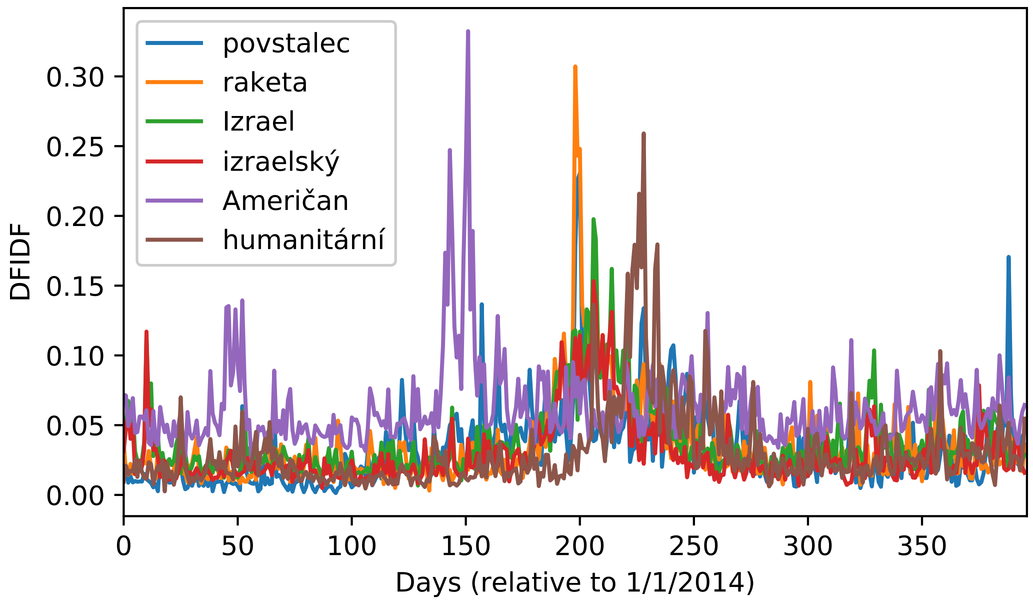


Figure 5.2: Example of an event detected using the embedded greedy method. The event consists of the words *rebel, missile, Israel, israeli, American* and *humanitarian*, and is related to the same real event as Figure 5.1.

The obvious requirement for the clustering algorithm is that it must require no prior knowledge of the desired number of clusters. Another requirement is that the algorithm must accept a custom distance measure.

We considered three candidate algorithms, all satisfying these requirements: Affinity propagation (Frey and Dueck, 2007), DBSCAN (Ester et al., 1996) and its modification, HDBSCAN (Campello et al., 2013).

During our experimentation, Affinity propagation performed poorly, its clusters being often seemingly random and of low quality. The quality of HDBSCAN clusters was considerably better, though the algorithm took longer to converge as the number of eventful words grew. It also required to tune multiple parameters, which was difficult to do without any annotated data. We decided to use the DBSCAN algorithm, which outperformed Affinity propagation as well, and does not require to tune as many parameters as HDBSCAN.

In addition to the previously stated requirements, DBSCAN is also capable of filtering out noisy samples (in our case words), not fit for any of the clusters. This property will prove advantageous for our task, as will become clear during the evaluation in Section 8.3.

5.3.1 Noise filtering

Before we apply clustering, we filter out the noisy parts from the word trajectories. Most words are on some level reported all the time, though only a fraction of these reportings corresponds to notable events. Unlike the greedy optimization described previously, clustering is prone to such noise, and would yield clusters of poor quality, often with trajectories being put together only due to their noisy parts being similar. Additionaly, with DBSCAN capable of filtering out noisy samples, some high quality words could be discarded precisely due to this noise in their (otherwise eventful) trajectories.

We want to keep only those trajectory parts exceeding a certain frequency level, distinguishing notable bursts from the general noise. We do this by computing a cutoff value for each event trajectory and discarding the sectors falling under this cutoff. This procedure is adopted from Vlachos et al. (2004). The algorithm is based on computing a moving average along the trajectory, and works as follows:

Algorithm 2 Burst filtering

```
Input: Window-length $l$, word trajectory $\boldsymbol{y_w}$
1: $\boldsymbol{ma}_l$ = Moving Average of length $l$ for $\boldsymbol{y}_w = [y_w(1), y_w(2), \dots, y_w(T)]$
2: $\mathit{cutoff} = \text{mean}\,(\boldsymbol{ma}_l) + \text{std}\,(\boldsymbol{ma}_l)$
3: $\boldsymbol{bursts}_w = [y_w(t) \mid y_w(t) > \mathit{cutoff}]$
Output: $\boldsymbol{bursts}_w$
```

We use the window length $l = 7$, looking 1 week back in the trajectory.

5.3.2 Distance function

We now define the distance function used by DBSCAN. It conveys similar information as the cost function in the previous two algorithms. We still need to measure the trajectory distance as well as semantic similarity between words, though the distance will now be defined strictly pairwise.

For a measure of trajectory distance, we replace the Kullback-Leibler divergence by the Jensen-Shannon divergence JSD (Lin, 1991), which is symmetric in its arguments. This is a necessary property of the distance function.

Although He et al. (2007a) did symmetrize the Kullback-Leibler divergence, they did not provide any source for their symmetrization form. We were unable to find a case where that particular form was used, though we discovered the Jensen-Shannon divergence, which comes from stronger mathematical background (Lin, 1991; Fuglede and Topsoe, 2004). It also tended to improve the clustering quality during our experimentation, as opposed to the original symmetrization. We then decided to replace the original paper's KL-divergence symmetrization by the JS-divergence.

Instead of semantic *similarity*, we measure semantic *distance* as the Euclidean distance between two word vector embeddings. The reason is that Euclidean distance is unbounded, which makes it possible for the samples to be spread farther apart. Since DBSCAN is a density-based clustering algorithm, having high density areas consisting of words with low trajectory distance and similar cosine similarities would cause them to appear in the same cluster. This would cluster the words only in terms of their trajectories, not semantics.

The distance between two words v and w with (normalized and filtered using Algorithm 2) trajectories $\boldsymbol{y'}_v$, $\boldsymbol{y'}_w$ and Word2Vec embeddings $\boldsymbol{v}_v$, $\boldsymbol{v}_w$ is now defined as

$$\mathrm{d}(v, w) = \mathrm{JSD}(\boldsymbol{y'}_v \| \boldsymbol{y'}_w) \cdot \|\boldsymbol{v}_v - \boldsymbol{v}_w\|_2, \tag{5.6}$$

with $\mathrm{JSD}(\boldsymbol{p}\|\boldsymbol{q}) = \frac{1}{2}\left(\mathrm{KL}(\boldsymbol{p}\|\boldsymbol{m}) + \mathrm{KL}(\boldsymbol{q}\|\boldsymbol{m})\right), \ \boldsymbol{m} = \frac{1}{2}\left(\boldsymbol{p} + \boldsymbol{q}\right)$.

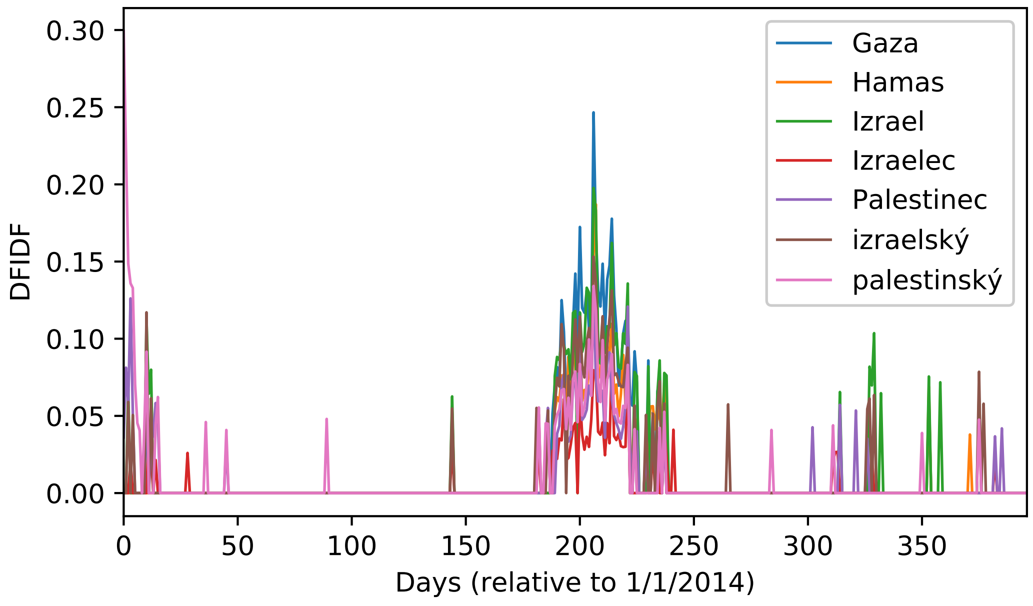


Figure 5.3: Example of an event detected using the cluster-based method. The event is related to the same real event as Figure 5.1 and Figure 5.2.

5.3.3 Event detection

Now, we describe the cluster-based event detection algorithm, which is a direct application of the DBSCAN algorithm and consequent noise filtering.

Algorithm 3 Cluster-based event detection

```
Input: Word set EW obtained in Chapter 4, matrix $\mathbf{Y}$, word embeddings for EW
1: Precompute a distance matrix $\mathbf{Dist} \in \mathbb{R}^{|\mathrm{EW}|\times|\mathrm{EW}|}$ with $\mathbf{Dist}_{ij} = \mathrm{d}(w_i, w_j)$
2: Apply DBSCAN to Dist, obtaining $k$ clusters and the noisy cluster
3: for each $(w, cluster) \in$ DBSCAN.clusters do
4:     if $cluster \neq noise$ then
5:         $e_{cluster}.KW = e_{cluster}.KW \cup w$
6:     end if
7: end for
Output: Events $\{e_1, e_2, \ldots, e_k\}$
```

In Figure 5.3, we show an event detected using the cluster-based method. It is again related to the same real event as those in Figure 5.1 and Figure 5.2. The main thing to note is that the word trajectories are clear of noise due to application of Algorithm 2. This made it possible to match words only based on their bursts, not any underlying noise. Compared to the event depicted in Figure 5.2, the cleaned trajectories overlap almost perfectly.

Chapter 6

Document retrieval

Having detected the events, we still have to present them to the user in a readable format. A set of keywords may be a concise representation for the computer, but it does not offer much insight into the event itself. We aim to generate short annotations for the events, based on which the user can decide to actually inspect the event more thoroughly and read some of the documents. Consequently, we need to retrieve a number of documents relevant to each event. These documents will then be used in Chapter 7 to generate summaries.

We can use each event's temporal and semantic information to query the document collection. The former is trivial – simply select the documents published within an event's bursty period. From these document, we can then select those document which relate to the event semantically. This will prove more complicated, and we will need to employ some more information retrieval techniques to obtain the documents.

As of now, an event e is described by a set of its keywords, $e.KW$. The goal is to convert this keyword representation to a document representation, $e.Docs$ consisting of documents related to e.

6.1 Event burst detection

First, we need to detect the period when the particular event happened, so that we can retrieve the documents published around that time. This part of the algorithm again follows from He et al. (2007a). In this paper, the period around an event's occurrence was called *bursty period*. The burst detection is done in five steps.

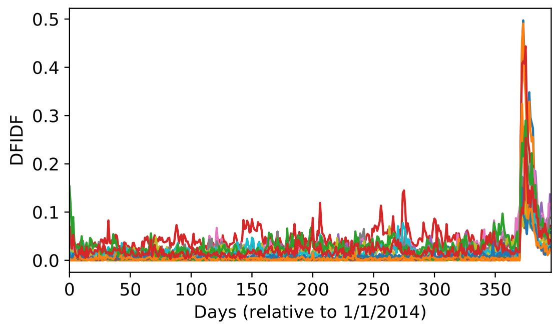
1. Construct the event trajectory from the trajectories of its keywords.

2. Clean the event trajectory.

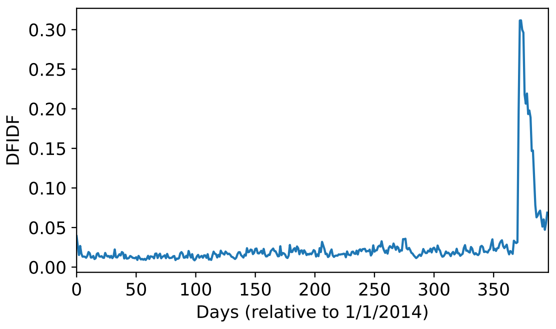
3. Determine the event's periodicity.

4. Fit a probability density function to the event trajectory.

5. Take the region(s) with the highest density as the bursty period(s).



(a) Event keywords



(b) Event trajectory

Figure 6.1: Keywords and trajectory of the event related to Charlie Hebdo attack. The keywords are *Charlie, Hebdo, Mohamed (Muhammad), Paříž (Paris), Islam, karikatura (caricature), Muslim, náboženství (religion), pařížský (Parisian), prorok (prophet), satirický (satirical), teroristický (terroristic), terorizmus (terrorism).*

6.1.1 Event trajectory construction

We first need to construct an *event trajectory* out of its *keyword trajectories.* We do this by computing a weighted average of the event's keyword trajectories, with weights being the keyword DPS. This ensures that less important words with slightly different time characteristic will not shift the trajectory away from the actual burst.

$$\boldsymbol{y_e} = \frac{1}{\sum_{k \in e.KW} \mathrm{DPS}_k} \sum_{k \in e.KW} \mathrm{DPS}_k \cdot \boldsymbol{y}_k \tag{6.1}$$

An example of such event trajectory can be found in Figure 6.1.

6.1.2 Trajectory filtering

Now, a typical event (shown in Figure 6.2) usually has a number of dominant bursts corresponding to the period(s) when the event actually occurred. Additionally, there are some milder, noisy bursts due to the keywords appearing elsewhere, independently of that particular event.

We aim to fit a probability density function to the event trajectory, as in He et al. (2007a). The noisy bursts would behave as outliers, shifting the fitted function away from the main event bursts. Once again, we apply the Burst filtering algorithm from Subsection 5.3.1 to filter out noise, this time from the event trajectories. The cutoff value computed by the Burst filtering algorithm is shown in Figure 6.2 as well.

6.1.3 Event periodicity

We apply the signal processing techniques described in Chapter 4 once more, this time to determine the dominant period $e.DP$ of each event e. After computing the periodogram, the dominant period is defined as the inverse of the frequency corresponding to the highest peak in the event trajectory:

$$e.DP = \frac{T}{\underset{k \leq \lceil T/2 \rceil}{\operatorname{argmax}} \|X_k\|^2}, \tag{6.2}$$

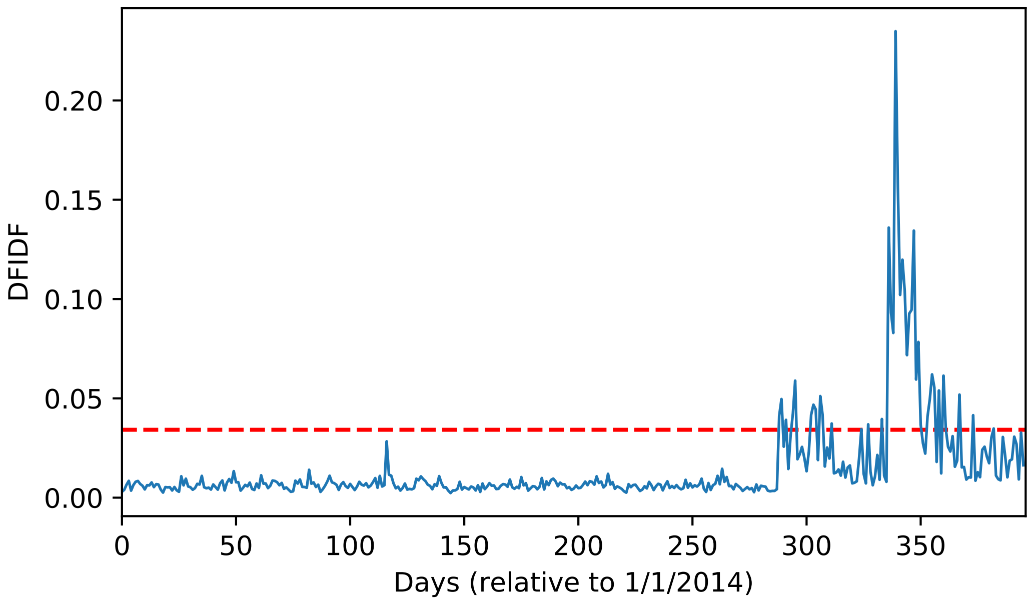


Figure 6.2: An event with a noisy trajectory and keywords *Vrbětice, muniční (ammunition), sklad (storage)* related to the explosions of the ammunition storage in Vrbětice on October 16 and December 3, 2014 (events 38 and 47 in Appendix A. The dashed red line is the cutoff value computed using window length 7. The parts of the trajectory under the cutoff will be discarded.

as the Fourier coefficient X_k denotes the amplitude at frequency $\frac{k}{T}$. We then consider an event e to be *aperiodic* if it happened only once in the stream, that is if $e.DP > \lceil T/2 \rceil$. Similarly, the event is *periodic* if $e.DP \leq \lceil T/2 \rceil$.

6.1.4 Density fitting

We normalize the event trajectories to have unit sums, so they can be interpreted as probability distribution over days. An element $y'_e(i)$ of the normalized trajectory $\boldsymbol{y'}_e$ can be interpreted as the probability of that event occurring on day i. This allows us to fit a probability density function to them. He et al. (2007a) adapted a similar approach, though only for word rather than event trajectories.

We describe aperiodic and periodic events separately, as different probability distributions must be used in case of a single burst than in case of multiple bursts.

1. **Aperiodic events**

   An aperiodic event trajectory $\boldsymbol{y'}_e$ is modeled by a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. We fit the Gaussian function to the trajectory $\boldsymbol{y'}_e$ and estimate the parameters μ and σ. He et al. (2007a) did not mention the method they used for aperiodic trajectories. As we are fitting the density to probabilities rather than observations, Maximum Likelihood Estimate of the parameters is not applicable. We decided to use non-linear least squares, namely the Trust Region Reflective algorithm (Branch et al., 1999) to estimate μ and σ bounded within the document stream period. An example of the Gaussian distribution fit to an event trajectory is shown in Figure 6.3.

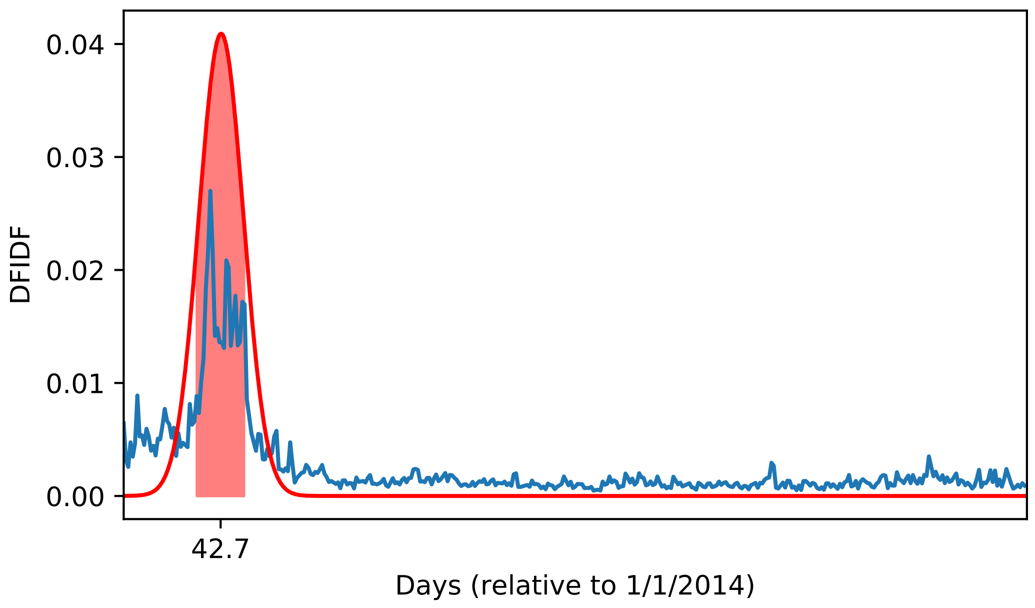


Figure 6.3: An aperiodic event consisting of the keywords *Sochi, ZOH (Winter Olympic Games), olympijský (olympic), olympiáda (olympiad)*. The Gaussian function modeling the trajectory and the event bursty period are shown in red.

2. **Periodic events**

   A periodic event trajectory $\boldsymbol{y'}_e$ is modeled using a mixture of $K = \lfloor T/e.DP \rfloor$ Cauchy distributions (as many mixture components as there are periods), as in Fisichella et al. (2011):

$$f(x) = \sum_{k=1}^{K} \alpha_k \frac{1}{\pi} \left(\frac{\gamma_k}{(x - \mu_k)^2 + \gamma_k^2} \right)$$

   The mixing parameters $\alpha_k \geq 0$, $\sum_{k=1}^{K} \alpha_k = 1$, location parameters μ_k and scale parameters γ_k are estimated using the EM algorithm.

   The Cauchy distribution has a narrower peak and thicker tails than the Gaussian distribution, which models the periodic bursts more closely. The individual bursts of a periodic event tend to be quite short, but even between two consecutive bursts, the frequency remains at a non-negligible level, which makes the Cauchy distribution a somewhat better choice. Figure 6.4 shows an example of such fit.

6.1.5 Burst detection

Using the fitted probability density functions, we define the bursty period(s) as the regions with the highest density. The bursty period of an aperiodic event e is now defined as $e.Bursts = \{[\mu - \sigma, \mu + \sigma]\}$. For a periodic event, there are $K = \lfloor T/e.DP \rfloor$ bursty periods defined as $e.Bursts = \{[\mu_k - \gamma_k, \mu_k + \gamma_k] \mid k = 1, \ldots, K\}$. The burst of an aperiodic event is highlighted in Figure 6.3, while a periodic event's bursts are shown in Figure 6.4.

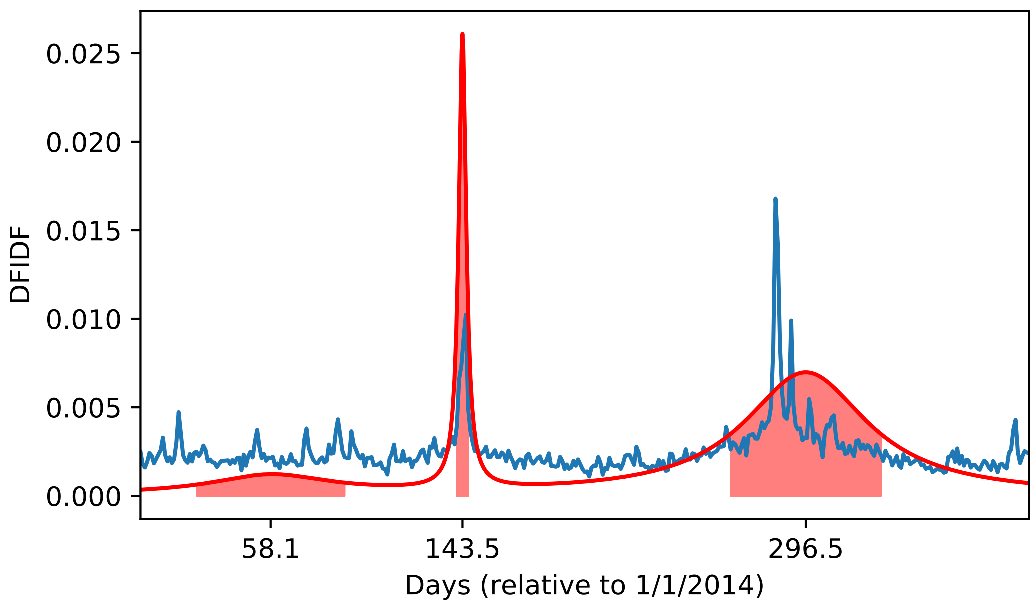


Figure 6.4: A periodic event with keywords *volba (election), volební (electoral), volič (voter)* and a period of 132 days. The event is modeled by a mixture of $\lfloor 396/132 \rfloor = 3$ Cauchy distributions. Each of the event's bursty periods is highlighted.

6.2 Document retrieval

We only describe the process for aperiodic events. The method is similar for periodic events, except applied on each burst individually.

We need to measure the relevance of individual documents published within an event's bursty period to the event. The only measure of semantics for an event we have is the event's keyword set $e.KW$. If we interpret $e.KW$ as a keyword query for the document collection, we arrive at the classical task of Information Retrieval. That is, to rate the documents in a given corpus by their relevance to the query (Manning et al., 2008).

In the original method by He et al. (2007a), the task was simple due to the cost function used. The only measure of semantic similarity was the degree of document overlap between all words in $e.KW$. If two words had no document overlap, they would not get assigned in the same event. That way, there was always at least one document in which all of the event's keywords appeared. It was a simple matter to compute the intersection of all documents containing either keyword within the bursty period. This is not the case in our method, and we will need to measure the document relevance in a more sophisticated way.

There are a few approaches we could take, such as project all documents and queries to a TFIDF (Term Frequency-Inverse Document Frequency) space (Manning et al., 2008) and sort the documents by their cosine similarity to the query. This simple approach does not go beyond a trivial keyword occurrence comparison, though after applying some weighting scheme. We could enrich it using Latent Semantic Indexing (Deerwester et al., 1990) to also take the document topics into account. This would require us to compute yet another model to be used for this part only, which would be computationally and memory-intensive.

Instead, we decided to further utilize the Word2Vec model and use the recently introduced Word Mover's Distance (Kusner et al., 2015), which is an application of the better known measure of Earth Mover's Distance (Rubner et al., 2000) to word embeddings.

The Word Mover's Distance (WMD) measures the similarity of two documents as the minimum distance the word vectors of one document need to "travel" to reach the word vectors of the second document. Since more similar words are embedded close to each other (Mikolov et al., 2013c), the farther apart the words lie, the less similar they are semantically. The formal definition of the WMD is rather lengthy, so we refer the reader to the original paper (Kusner et al., 2015) for the full derivation.

The WMD discards word order, which makes it suitable for our keyword queries. As the authors note, it achieves best results for short documents, in part due to the method being computationally expensive for larger pieces of text. Therefore, we apply the WMD to document headlines only.

In Information Retrieval, it is more traditional to work with document similarity rather than distance. In the Gensim framework (Řehůřek and Sojka, 2010) which implements the WMD, the similarity is defined as

$$\mathrm{Sim}_{WMD}(d_i, d_j) = \frac{1}{1 + \mathrm{WMD}(d_i, d_j)} \tag{6.3}$$

which is 1 if $\mathrm{WMD}(d_i, d_j) = 0$ and goes to 0 as $\mathrm{WMD}(d_i, d_j) \to \infty$.

We now describe the algorithm to compute the document representation of an event.

Algorithm 4 Document representation of an aperiodic event

```
Input: Event $e$, $burst \in e.Bursts$, number of documents $n$, document stream $D$
1: $burst\_docs = \emptyset$
2: for each $doc \in D$ do
3:     if $doc.publication\_date \in burst$ then
4:         Compute $\mathrm{Sim}_{WMD}(e.KW, doc.headline)$
5:         $burst\_docs = burst\_docs \cup doc$
6:     end if
7: end for
8: Sort $burst\_docs$ by the computed $\mathrm{Sim}_{\mathrm{WMD}}$ in descending order
Output: first $n$ elements of $burst\_docs$
```

The set of event documents *e.Docs* is then a union of the outputs of Algorithm 4 over all bursts in *e.Bursts*.

In our experiments, we chose the number of documents n as the square root of total number of documents within the particular event burst.

Chapter 7

Event annotation

The final step of our method is to annotate the detected events in a human-readable way. We aim to generate short summaries so that the user does not have to process a large quantity of text, and can just skim through a few sentences to decide whether he is interested in that particular event. If so, then he can examine the event more closely and go through the actual documents, which we have retrieved in chapter Chapter 6.

Although the keyword set discovered in Chapter 5 provides a concise representation of an event, it can lead to ambiguities or simply not reveal enough information. The keywords should be considered an internal representation used in the detection process, not a feature presentable to the user.

An example of such event whose background is unclear from the keyword set is shown in Figure 7.1. After manual examination, we discovered that the event concerns Viktor Yanukovych being ousted from Ukraine's presidency, though it is unclear from the keyword set.

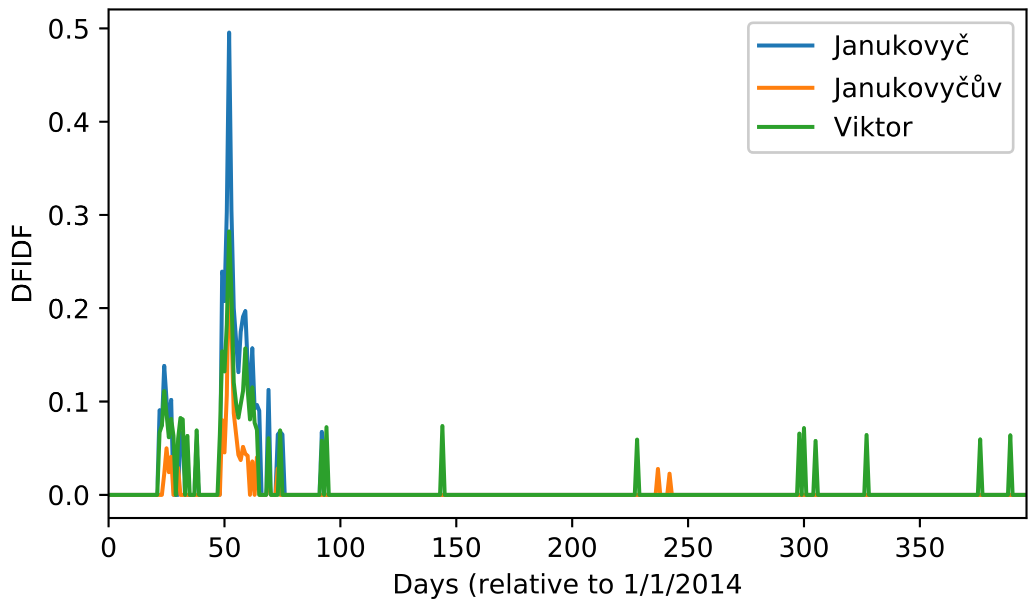


Figure 7.1: An event whose meaning is not clear from the keyword set.

A simple method is to annotate an event by the headline of the most relevant document in terms of Word Mover's Similarity. This may give insight of the general topic of the particular event, but it is unlikely that a whole event will be well characterized by a single document. For this reason, we also investigate a more complex method.

To obtain richer annotations, we apply multi-document summarization techniques to generate a short summary of an event's document set. More specifically, we attempt to extract a subset of sentences out of the event documents, which cover the general topic of the event without providing redundant information. As the documents come from different sources and describe the events from different perspectives, the result will not generally be a continuous paragraph, but more of a set of characteristic sentences. Still, a longer piece of text will likely provide a better insight into an event than a single headline.

We examined the multi-document summarization system presented in Lin and Bilmes (2010, 2011). This system was later improved by Kågebäck et al. (2014), who evaluated the usage of different word embedding techniques for sentence similarity measures. Their work led to the system presented in Mogren et al. (2015) which aggregates several different similarity measures to obtain a better quality summary. We adapt their system and combine together several measures of sentence similarity suitable for the event detection task.

7.1 Multi-document summarization

In Lin and Bilmes (2010), the authors formulate the task of multi-document summarization as a constrained combinatorial optimization problem, where the goal is to retrieve a subset of sentences maximizing a monotone submodular function $\mathcal{F}(\cdot)$ measuring the summary quality.

A submodular function $\mathcal{F}(\cdot)$ on a set of sentences U satisfies the property of *diminishing returns*; that is, for $A \subseteq B \subseteq U \setminus \{v\},\ \mathcal{F}(A \cup \{v\}) - \mathcal{F}(A) \geq \mathcal{F}(B \cup \{v\}) - \mathcal{F}(B),\ v \in U$. This has an intuitive interpretation for text summarization, namely that adding a sentence v to a longer summary does not improve the summary as much as adding it to a smaller one. The reason is that the information carried by v is likely already present in the longer summary.

Even though solving the task exactly is NP-hard, a greedy algorithm is guaranteed to find a solution only a constant factor off the optimum, as discussed by the authors.

The summary quality is measured in terms of how representative it is to the whole set (coverage) and how dissimilar the sentences are to each other (diversity). The constraints limit the summary to a reasonable length by bounding the total number of words.

In Lin and Bilmes (2010), basic submodular functions to be used in multi-document summarization are described. In Lin and Bilmes (2011), these functions are further developed to better capture the semantic properties of sentences.

Mathematically, the task is formulated as

$$\begin{aligned} \max_{S \subseteq U} \quad & \mathcal{F}(S) = \mathcal{L}(S) + \lambda \mathcal{R}(S) \\ \text{s. t.} \quad & \sum_{i \in S} c_i \leq \mathcal{B}, \end{aligned} \tag{7.1}$$

where U is the set of all sentences from the document set being summarized,

c_i is the number of words in sentence i and $\mathcal{B}$ is the total budget, i.e. the desired maximum summary length.

A feasible set S maximizing $\mathcal{F}(\cdot)$ will provide a reasonable number of sentences well capturing the overall topic of the whole document set, no two of which being redundant. What remains is to define the coverage function $\mathcal{L}(\cdot)$ and diversity function $\mathcal{R}(\cdot)$, whose influence can be controlled by the parameter $\lambda \geq 0$. Additionally, the functions must be defined in a way that the submodularity conditions from Lin and Bilmes (2010) are not violated, so that a greedy algorithm can still be used with performance guarantee.

7.2 Coverage function

In Lin and Bilmes (2011), the coverage function $\mathcal{L}(\cdot)$ is defined in terms of pairwise sentence similarity $\text{Sim}(\cdot, \cdot)$ as

$$\mathcal{L}(S) = \sum_{i \in U} \min \Big\{ \sum_{j \in S} \text{Sim}(i, j), \alpha \sum_{j \in U} \text{Sim}(i, j) \Big\}. \tag{7.2}$$

The first argument of the minimum measures the similarity between the sentence i and the summary S, while the second argument measures the similarity between the sentence i and the rest of the sentences U. The number $\alpha \in [0, 1]$ is a threshold coefficient controlling the influence of the overall similarity.

In Lin and Bilmes (2010), the authors further prove that if $\text{Sim}(i, j) \in [0, 1]\ \forall i, j \in U$, the whole function remains submodular.

Originally, only a simple cosine similarity between TFIDF sentence vectors (Manning et al., 2008) was used as $\text{Sim}(\cdot, \cdot)$. Kågebäck et al. (2014) examined various methods of word embeddings to obtain a finer measure of similarity. This alone outperformed the original method. In Mogren et al. (2015), a more complex system aggregating multiple similarity measures was built, further improving the summary quality. The authors compute the sentence similarity $\text{Sim}(i, j)$ as a product of these individual similarities, all bounded in $[0, 1]$:

$$\text{Sim}(i, j) = \prod_{l} \text{M}^{l}_{s_i, s_j}. \tag{7.3}$$

We use this method with several different similarity measures fit for the event detection task. Next, we describe the individual sentence similarities used.

7.2.1 TFIDF similarity

The first measure used is the standard cosine similarity between TFIDF (Term Frequency-Inverse Document Frequency) vectors (Manning et al., 2008) of two sentences s_i and s_j. Such method is a simple measure of document similarity often used in information retrieval.

If we denote the frequency of the word w in sentence s_i as $\text{tf}_{w,i}$ and the inverse document frequency of w as idf_w, the similarity is written as

$$\text{M}^{TFIDF}_{s_i, s_j} = \frac{\sum_{w \in s_i \cup s_j} \text{tf}_{w,i} \cdot \text{tf}_{w,j} \cdot \text{idf}^2_w}{\sqrt{\sum_{w \in s_i} \text{tf}^2_{w,i} \cdot \text{idf}^2_w} \cdot \sqrt{\sum_{w \in s_j} \text{tf}^2_{w,j} \cdot \text{idf}^2_w}}. \tag{7.4}$$

The term frequencies are always non-negative, and so the whole cosine similarity is in $[0, 1]$.

The major setback of TFIDF similarity is that it does not go beyond simple word overlap, though weighted to diminish stopwords and amplify important words. That means that if two sentences convey essentially the same information through different vocabulary, they will not be ranked similar due to having only a few words in common. That can be a problem in larger document collections from different sources and authors.

7.2.2 Word2Vec similarity

We attempt to solve this problem by considering the word embeddings of the individual words, as first examined by Kågebäck et al. (2014).

We represent a sentence s_i by summing together the vector embeddings of its words, $\boldsymbol{v}_i = \sum_{w \in s_i} \boldsymbol{v}_w$. The similarity of two sentences is then the cosine similarity of these vectors, transformed to $[0, 1]$:

$$\mathrm{M}^{W2V}_{s_i,s_j} = \left(\frac{\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{\|\boldsymbol{v}_i\| \cdot \|\boldsymbol{v}_j\|} + 1 \right) / \ 2. \tag{7.5}$$

This similarity brings a finer distinction of word-level semantics. This means that even if two sources reporting the same event use fairly different vocabularies, the sentences will still be ranked similar.

7.2.3 TR similarity

The next measure uses Text Rank (TR) similarity, as defined by Mihalcea and Tarau (2004). Each sentence is represented by a set of words, and the overlap of these sets is measured. Mogren et al. (2015) achieved best results by combining the TFIDF similarity, word embeddings and the TR similarity, which is defined as

$$\mathrm{M}^{TR}_{s_i,s_j} = \frac{|s_i \cap s_j|}{\log|s_i| + \log|s_j|}. \tag{7.6}$$

7.2.4 Keyword similarity

In addition to the three previously described similarities, Mogren et al. (2015) considered a keyword similarity, which measures the overlap between two sentences and a predefined keyword set. Having previously obtained the event keyword representation $e.KW$, we use this measure to make sure the sentences actually concern the particular event.

The similarity is defined as

$$\mathrm{M}^{KW}_{s_i,s_j} = \frac{\sum_{w \in (s_i \cap s_j \cap e.KW)} \mathrm{tf}_w \cdot \mathrm{idf}_w}{|s_i| + |s_j|}. \tag{7.7}$$

The measure effectively chooses only those sentences having non-zero word overlap with the keyword set. It also breaks the summary fluency, making the summary more of a set of sentences characteristic for the given event. On the other hand, the resulting sentences will be highly relevant to the event, often revealing important information about it.

This tradeoff between fluency and quality is more worth it when summarizing a large number of documents. Even without the keyword similarity, the chance that two summary sentences will make sense consecutively is quite small, when they come from different documents.

If an event consisted of only one or two documents, it would make sense to use a different measure to reach better fluency.

7.3 Diversity function

Now, we can define the diversity function, which will positively reward a summary consisting of non-redundant sentences. Lin and Bilmes (2011) first applied the K-Means clustering algorithm to the TFIDF vectors of the sentences in U. The diversity function then positively rewards summaries whose sentences come from different clusters. If the clustering well separates the sentences in the semantic sense, the sentences chosen from different clusters will not carry redundant information.

The diversity function is defined as

$$\mathcal{R}(S) = \sum_{k=1}^{K} \sqrt{\sum_{j \in S \cap P_k} r_j}, \tag{7.8}$$

where $P_k,\ k = 1, \ldots, K$ is a clustering of the sentence set U. The value $r_j = \frac{1}{|U|} \sum_{i \in U} \mathrm{M}^{TFIDF}_{s_i, s_j}$ is the singleton reward for adding the sentence i into the summary S.

This function positively rewards diverse sentences in a sense that once an element i from a cluster P_k is chosen, other elements from the same cluster will have diminishing gains due to the square root function (see Lin and Bilmes (2011) for a concrete example). The summarization will then prefer sentences from yet unused clusters.

7.4 Optimization

Having defined the cost function $\mathcal{F}(\cdot)$, we can use the greedy algorithm defined in Lin and Bilmes (2010) to obtain the desired summary $e.Annotation$ for an event e.

For the experiments, we set a budget of 50 words. In the diversity function the number of clusters K was set to $\frac{|U|}{10}$, putting 10 sentences into each cluster on average. As for the other parameters, we used the values from the original papers (Lin and Bilmes, 2010, 2011). Additionally, we need to specify which documents we will use for the summarization. In theory, there is no limit to the number of documents, though it would make sense to use only a few most-relevant documents. In Chapter 8, we will limit the number of documents for efficiency reasons.

7.5 Results

In Table 7.1, we show the annotation for the event depicted in Figure 7.1. Though the keywords do not reveal much, the event's topic is fairly clear from the longer summary. We also include the headline of the most relevant document for comparison. In this case, the headline does not provide much insight into the event.

Janukovyč, Janukovyčův, Viktor (Feb 15 - Mar 2, 2014)
Kdo stojí za Viktorem Janukovyčem? Kyjevská úřadovna prezidenta Viktora Janukovyče je bez stráží. Režim Viktora Janukovyče se zhroutil. Moc ukrajinského prezidenta Viktora Janukovyče se o víkendu zhroutila. Odvolaného ukrajinského prezidenta Viktora Janukovyče stíhá policie. “Já , Viktor Janukovyč, se obracím na lid Ukrajiny.” Svržený ukrajinský prezident Viktor Janukovyč se voleb zúčastnit nechce.

Table 7.1: Annotation for the event depicted in Figure 7.1

In Table 7.2, an event with highly redundant summary is shown. In this summarization, the diversity function (7.8) failed to diminish similar sentences, and most of the summary simply repeats the same information. On the other hand, the most relevant document’s headline represents the event very well, and would suffice to make sense of it.

Adriano, Krnáčová, primátorka (Oct 12 - Dec 1, 2014)
Pražskou primátorkou bude Adriana Krnáčová z ANO Novou pražskou primátorkou bude Adriana Krnáčová. Primátorkou Prahy bude Adriana Krnáčová (ANO). Novou pražskou primátorkou bude Adriana Krnáčová z ANO. Novou pražskou primátorkou bude Adriana Krnáčová z ANO. Adriana Krnáčová je původem ze Slovenska. Pražskou primátorkou byla zvolena Adriana Krnáčová z hnutí ANO. Bratislavská rodačka Adriana Krnáčová je primátorkou Prahy.

Table 7.2: Example of a summary with high degree of redundancy.

Neither of the summaries is fluent enough to be read as an article. In both cases, the summaries resemble unordered sets of sentences, most of which give some insight into the underlying event. The user can stil gain some information from these summaries and consequently decide whether he is interested in the events enough to read the documents.

Further examples of the generated summaries can be found in Appendix B.

Chapter 8

Evaluation

In this chapter, we compare the three event detection methods from various standpoints. We will compare the original method ("original"), its modification using word embeddings ("embedded-greedy") and the method using clustering algorithm to group similar words together ("cluster-based").

Most of these evaluations rate the quality of the detected events on the keyword level. We will be referring to the average number of keywords per event, which we provide in an overview in Table 8.1.

We discarded trivial events only consisting of a single keyword.

Method	Events detected	Keywords	Keywords/event
Original	217	451	2.08
Embedded-greedy	46	473	10.28
Cluster-based	77	761	9.88

Table 8.1: Overview of the detected events

As we can see from the table, the original method's events are not very rich, majority of them consisting of only 2 keywords. Since the original method is applied to the same words as the other methods detecting fewer events, we can expect a considerable level of redundancy, as we will see in Section 8.2.

Furthermore, the keywords are used to query the document collection for event documents. Having only a few keywords makes it difficult to unambiguously retrieve related documents, leading to a poor document set. This will become clear in Section 8.4.

The other two methods detect considerably fewer events which generally consist of a higher number of keywords. We can expect these methods to rank higher in the evaluations, provided that the events themselves are meaningful and not comprising of noisy words. This is an issue we will address in Section 8.3.

8.1 Precision, Recall, F-measure

First, we evaluate the precision and recall, which are metrics commonly used in information retrieval (Manning et al., 2008; Rijsbergen, 1979). The evaluation is done with respect to a reference set of real events that occurred during the examined period. The list can be found in Appendix A.

Given a set of detected events *Detected* and the set of reference events *Reference*, the Precision (P) and Recall (R) are defined as

$$\begin{aligned} P &= \frac{|\text{Detected} \cap \text{Reference}|}{|\text{Detected}|} \\ R &= \frac{|\text{Detected} \cap \text{Reference}|}{|\text{Reference}|}. \end{aligned} \tag{8.1}$$

The Precision and Recall scores are then merged together using the F_1 score, which is their harmonic mean scaled into $[0, 1]$:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}. \tag{8.2}$$

Although the reference list of real events is not exhaustive, it does provide a way to objectively compare the individual event detection methods. While there are many more events that happened during the examined time period, a sample of events important enough provides a picture of the methods performance.

We manually inspected the detected events and matched them with the reference events. Out of this assignment, we calculated the precision, recall and F-measure. The results are shown in the table below.

Method	**Precision**	**Recall**	**F$_1$ score**
Original	16.35%	**28.57%**	20.80%
Embedded-greedy	8.70%	10.20%	9.39%
Cluster-based	**25.97%**	**28.57%**	**27.21%**

Table 8.2: Precision, Recall and F-measure comparison (manual evaluation)

The original method's precision was poor due to high recurrence of events not appearing in the reference list, which will be more clear in redundancy evaluation later. As the average number of keywords per event is low, the real events are scattered among many detected events. On the other hand, the cluster-based method attained the highest precision due to events consisting of more keywords, meaning lower recurrence.

The embedded greedy method's precision and recall were poor both for the same reason as the original method, as well as some events consisting of keywords unrelated to each other. This made them difficult to assign to their real world counterparts.

In addition, we attempted to measure precision and recall in a more automatic way, so that the evaluation does not entirely depend on a manual input.

A real event, consisting of occurrence date and a headline, was considered detected if its date was found within a bursty period of some detected event, and if its headline had nonzero intersection with the detected event's keyword set.

Method	Precision	Recall	F_1 score
Original	7.14%	16.33%	9.94%
Embedded-greedy	**26.09%**	22.45%	**24.13%**
Cluster-based	20.78%	**28.57%**	24.06%

Table 8.3: Precision, Recall and F-measure comparison (automatic evaluation)

Unlike the manual evaluation, the automatic one favors the embedded greedy approach. The reason is that its events generally contain more keywords, so there is a higher chance the keywords intersect a headline of some real event, marking the event as detected.

On the other hand, the original method's keyword sets usually consist of only two words that may not appear in the real event headlines at all. This makes it score rather poorly in the automatic evaluation.

The cluster-based method's results are similar to the manual evaluation.

8.2 Redundancy

Next, we evaluate redundancy — the tendency to scatter a real event among several detected events. We noticed that the original method tends to scatter a real event among several detected events. This is an undesirable behavior which we attempt to suppress in the other two methods by using a less-strict word similarity measure. This is an evaluation that must be performed manually, since if we had an automatic way of recognizing redundancy, we would be able to avoid it entirely.

We assembled the detected events into groups, where each group is a set of events related to the same real-world event. If it was unclear which real event does a detected event refer to, we considered two events equal if they had similar burst characteristics and semantically similar keywords.

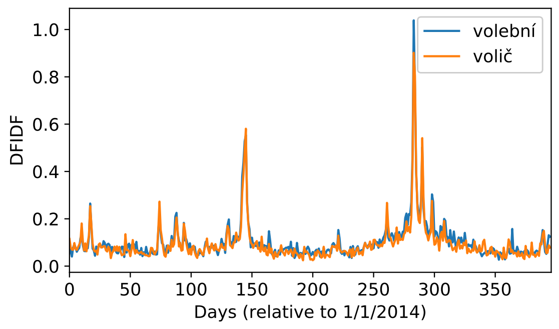
The redundancy is then computed as $1 - (|\text{groups}| \,/\, |\text{events}|)$. An ideal event detection method would result in singleton groups, where each real-world event was covered precisely by one detected event. This would result in numerator being the same as the denominator, and the total redundancy of 0.

In Figure 7.1, we show an example of two redundant events which should be merged into one. They were detected using the original method. The main burst around day 290 is related to elections to the Czech senate (events 36 and 40 in Appendix A). It is reasonable to assume that the four words appeared in some documents together, so their document overlap must have been nonzero, and they should comprise one event only.

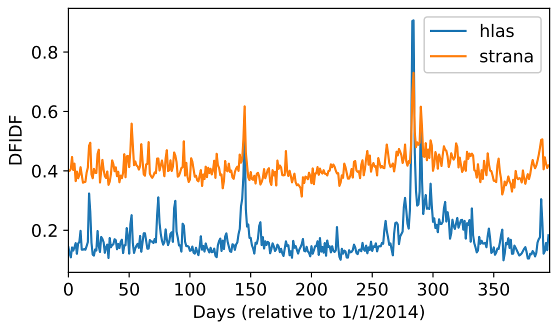
Method	Redundancy
Original	77.99%
Embedded-greedy	65.22%
Cluster-based	**42.86%**

Table 8.4: Redundancy comparison

Large redundancy of the original method is to be expected with events consisting of only 2 keywords on average.



(a) An event with keywords *election-related, voter*



(b) An event with keywords *vote, political party*

Figure 8.1: Example of two redundant events detected by the original method.

8.3 Noisiness

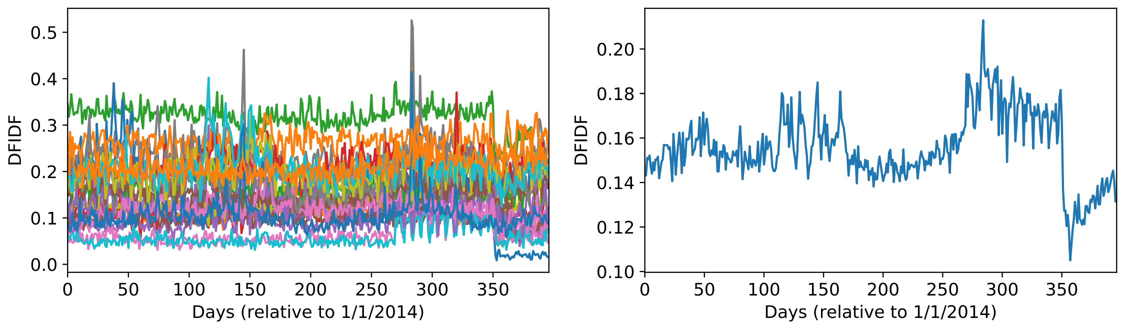
When we checked the detected events manually, we noticed that some events are formed of keywords unrelated to each other, or do not show any clear bursts in their trajectory. Outputting such events is clearly undesirable, as they are difficult to make sense of.

Motivated by this, we seeked to evaluate the methods in terms of how many events they output share such noisy characteristic. We manually examined the events detected for signs of noise in trajectories or keyword sets. An event is considered noisy if its trajectory does not contain any distinguishable burst of activity, or if it consists of keywords unrelated of each other.

We realize that this examination is largely subjective. However, we did not find any simple way of automating the evaluation without losing interpretability of the results. It would be possible to impose a threshold on the overall event trajectory variance or keyword similarity, under which the event would be considered noisy. Such threshold would be an arbitrary value though, and the method could misclassify some events.

In Figure 7.2, we show a typical noisy event that comes from the embedded greedy algorithm. Not only are the keywords mostly unrelated to each other, they also do not carry much information about what real event could possibly be happening. The event trajectory does not contain many notable bursts. Perhaps except around day 290, the overall trajectory value does not vary greatly to reveal any burst of interest.

Although the keyword trajectories are not all similar, there could be an overlap high enough for them to be put together. We suspect that the main level of similarity comes from the Word2Vec model, though. It is well possible that these words appeared in similar context, even though they are not representative of any real event. The Word2Vec model would then rate them similar, and they would end up in one event.



(a) Event keyword trajectories (b) Trajectory of the same event

Figure 8.2: (a) Event with noisy keywords and trajectory. The keywords are *top, autor (author), podnik (business), test, fotogalerie (photogallery), reklama (advertisement), výsledek (result), komentář (commentary), zpravodajství (reporting), hra (game), informace (information), novinka (hot news), akce (action), návrh (proposition), Martin, klíčový (key, adj.), vedení (leadership), projekt (project), program, účast (participation), ČTK (Czech News Agency).* (b) Trajectory of the same event constructed from the keywords.

Method	**Noisiness**
Original	50.94%
Embedded-greedy	19.57%
Cluster-based	**19.48%**

Table 8.5: Noisiness comparison

Here, the cluster-based method performed the best, as the clustering algorithm chosen (DBSCAN) is capable of automatic filtering of noisy samples. With the distance function measuring both trajectory and keyword similarity, it filters out words unrelated to any event.

Surprisingly, the embedded greedy method performed only slighly worse than the cluster-based method. Although the embedded greedy method reached considerable redundancy, manual check revealed that the events mostly consist of important keywords, and most of the trajectories contain distinguiushable bursts.

Poor performance of the original method is partially caused by its large redundancy. Even if a number of words with noisy trajectories appears in similar documents, the method would not group them together to a single event, but split into several noisy events. All of these noisy events then negatively contribute to the noisiness score.

8.4 Purity

All previous evaluations concerned the events on the keyword level. The purity measure will evaluate the event document sets in terms of topical consistency. This is a metric used by Aggarwal and Subbian (2012), and the definition can be found in Manning et al. (2008).

We interpret each event as a cluster of documents. The evaluation is then an application of the standard measure of cluster purity, which measures the consistency of class labelling within each cluster. Clearly, a high quality event should contain documents concerning similar topics. The problem is that our documents do not have any notion of class labels denoting their topics, which we will have to supplement.

Similarly to Aggarwal and Subbian (2012), we first assembled a list of 50 words from 1000 most often occurring Nouns and Verbs in document headlines. The words are *Ukrajina (Ukraine), Rusko (Russia), policie (police), soud (court), Zeman, EU, Sparta, festival, Babiš, Putin, Google, ekonomika (economics), letadlo (airplane), východ (east), politika (politics), zabít (to kill), poslanec (deputy), armáda (army), Kyjev (Kiev), Škoda, hokejista (hockey player), fotbalista (football player), doprava (traffic), vražda (murder), Vánoce (Christmas), Francie (France), sport, NATO, Moskva (Moscow), ropa (petroleum), turnaj (tournament), Obama, referendum, ebola, parlament (parliament), koalice (coalition), Paříž (Paris), automobil, mistrovství (championship), elektrárna (power plant), Sýrie (Syria), islamista (islamist), Brusel (Brussels), olympiáda (olympics), sníh (snow), průmysl (industry), revoluce (revolution), výbuch (explosion), finance, terorista (terrorist).* All documents that contained any of these words in their headline were tagged with the corresponding class label.

Then, for each event, we computed the number of documents tagged by the most frequent label in the event. These values are then summed over all events and divided by the total number of tagged documents from all events.

Mathematically, this is formulated as

$$\text{Purity} = \frac{1}{|\text{Tagged}|} \sum_{e \in \text{Events}} \max_{t \in \text{Tags}} |\{d \in e.Docs \mid d.tag = t\}|, \tag{8.3}$$

where Tagged $= \bigcup_{e \in \text{Events}} \{d \in e.Docs \mid d.tag \text{ is defined}\}$ and Tags is a set containing the 50 manually selected words. The results are shown in Table 8.6.

Method	**Purity**
Original	30.53%
Embedded greedy	44.42%
Cluster-based	**61.08%**

Table 8.6: Purity comparison

Poor performance of the original method can be explained by the events consisting mostly of 2 keywords. Such short query for the document collection does not distinguish the documents very well, often retrieving unrelated texts.

The other two methods produce events generally containing more keywords, so the queries are more specific. This allows us to retrieve more relevant documents and reach higher purity. The better results may also be caused by the Word Mover's Distance used as the document similarity metric. However, it is arguable whether the WMD could beat the selection of precisely those documents that contain all the event keywords, provided there were more than 2 keywords in general.

8.5 Computation time

Finally, we evaluate the computation time. We measure the execution time of the individual detection steps, so it is clear which parts are the bottlenecks. All experiments were performed on a laptop computer with a 64bit operating system, quad-core processor and 8GB RAM.

In the embedded-greedy and cluster-based methods, we did not retrieve documents for periodic events with a period of 7 days or lower. The reason is that such short periods will essentialy cover the entire stream with short bursts and the Word Mover's Distance, computationally expensive on its own, will need to be computed to almost all documents. This might take a day's worth of time, and the short period events mostly concern unineresting events such as sport matches, weather forecasts, etc.

In addition, the summaries were extracted only using 50 most relevant documents instead of all documents retrieved, so that the overall process would not take too long.

Unit	**Original**	**Embedded**	**Cluster**
Word2Vec embedding	N/A	3h 50min	
Bag of words model construction	$\longleftarrow$	37min	$\longrightarrow$
Word trajectories & spectral analysis	$\longleftarrow$	8s	$\longrightarrow$
Event detection	2min 12s	38s	4min 50s
Document retrieval	7min 30s	6h	7h 40min
Event annotation	3h 22min	3min 38s	7min 30s
Total	4h 9min	10h 31min	12h 20min

Table 8.7: Computation time comparison

The original method's document retrieval took considerably less time than the other two methods. The reason is that the original method does not use the Word Mover's Distance as a similarity measure. Due to word semantic similarity being measured only in terms of their document overlap, there is always at least one document containing all the event keywords. It is sufficient to take the documents containing either of the event keywords and intersect these sets, which is much faster than calculating the distance.

This is also the reason why the event annotation took longer in the original method. Having also retrieved documents for events with short period (7 days and less), it is necessary to summarize each period independently. When we summarized only aperiodic events and those with period higher than 7 days, summarization took 1h 12min.

Chapter 9

Conclusion and future work

In this thesis, we addressed retrospective event detection from text streams. The task is to analyze a given document collection and uncover real events that happened over the stream period. These events are described by semantically related keywords whose occurrence in the stream has similar temporal characteristic. Once the events have been found, the relevant documents can be recovered by querying the documents with the event keywords. We focused on various ways to augment the event detection by using Word2Vec model (Mikolov et al., 2013a) to measure the word semantic similarity as well as retrieve the documents.

First, we attempted to augment an existing method by He et al. (2007a) to use a Word2Vec-based similarity function to match semantically related words together. This led to an improvement over the original method, mainly in the average number of keywords per event. The modified method reached lower detection redundancy and the found events were less noisy. However, the method declined in precision and recall.

Then, we explored a different approach, where we interpreted the keyword-based event detection as a literal clustering task. We defined a custom distance function also utilizing the Word2Vec model as a semantic measure. We then applied a clustering algorithm equipped with this distance function to words previously selected as eventful. Our evaluation suggests that this method was more successful than both the original method and its Word2Vec modification. The cluster-based method reached even lower redundancy and noisiness than the previous methods while surpassing them in precision and recall.

The disadvantage of both our methods is the necessity to train the Word2Vec model, which is time consuming. However, the Word2Vec model supports online learning; the training can be stopped and resumed as necessary. When we are about to detect events from a different document collection with a distinct vocabulary, we can simply embed the new words prior to starting the algorithm.

We also examined how the Word2Vec model could be used to retrieve documents concerning the detected events. We applied the Word Mover's Distance (Kusner et al., 2015) to documents within each event's bursty period as a measure of their relevance to that particular event's keyword set. We then selected the most relevant documents as the event's document representation. Although the documents were of high quality and represented the events well, the process took an unbearable amount of time. In the original method, the retrieval process was more straightforward and much more efficient.

Finally, we applied multi-document summarization techniques to the documents to obtain a short annotation describing each event. These summaries, along with the event's occurrence dates and document sets, are the outputs of our method presented to the user. The summaries serve the purpose of giving a quick reference of the event's topic, based on which the user may decide to examine the event further and go through the retrieved documents.

In future work, it would be beneficial to use a more efficient way of computing the documents relevant to each event. Traditional information retrieval techniques, such as Latent Semantic Indexing (Deerwester et al., 1990) could be used here, perhaps with some domain specific knowledge of the underlying events, such as their bursty periods.

Also, we would like to examine how an event could be represented directly as a set of documents, rather than keywords. Although there are attempts to do so (He et al., 2007b), they require to fine-tune even more parameters than our method, and the document representation is again constructed using word trajectories. The Doc2Vec model (Le and Mikolov, 2014), a generalization of Word2Vec able to embed whole documents in a vector space, could be used to obtain the semantic representation.

Instead of computing a cutoff value to clean a word or an event trajectory, as we did in Subsection 5.3.1 and Subsection 6.1.2, further signal processing techniques could be applied on the trajectories to separate the dominant bursts from the underlying noise. The result would be a somewhat cleaner trajectory devoid of any milder bursts of no interest. This could lower the noisiness, since words would be matched together based on only the dominant activity, not any underlying influence, which still eludes the cutoff value method.

Bibliography

C. C. Aggarwal and K. Subbian. Event detection in social streams. In *Proceedings of the 2012 SIAM international conference on data mining*, pages 624–635. SIAM, 2012.

E. Alfonseca, D. Pighin, and G. Garrido. Heady: News headline abstraction through event pattern clustering. In *ACL (1)*, pages 1243–1253. Citeseer, 2013.

J. Allan. Introduction to topic detection and tracking. *Topic detection and tracking*, pages 1–16, 2002.

J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study final report. 1998.

F. Atefeh and W. Khreich. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164, 2015.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

M. A. Branch, T. F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

I. Brigadir, D. Greene, and P. Cunningham. Adaptive representations for tracking breaking news on twitter. *arXiv preprint arXiv:1403.2923*, 2014.

R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 160–172. Springer, 2013.

A. J. Chaney, H. Wallach, M. Connelly, and D. M. Blei. Detecting and characterizing events. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1142–1152.

T. M. Cover and J. A. Thomas. *Elements of information theory.* John Wiley & Sons, 2012.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

R. Ferreira, L. de Souza Cabral, F. Freitas, R. D. Lins, G. de França Silva, S. J. Simske, and L. Favaro. A multi-document summarization system based on statistics and linguistic treatment. *Expert Systems with Applications*, 41(13):5780–5787, 2014.

M. Fisichella, A. Stewart, A. Cuzzocrea, and K. Denecke. Detecting health events on the social web to enable epidemic intelligence. In *Proceedings of the 18th International Conference on String Processing and Information Retrieval*, SPIRE'11, pages 87–103, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-24582-4. URL `http://dl.acm.org/citation.cfm?id=2051073.2051083`.

B. J. Frey and D. Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, page 31. IEEE, 2004.

G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. Parameter free bursty events detection in text streams. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 181–192. VLDB Endowment, 2005. ISBN 1-59593-154-6. URL `http://dl.acm.org/citation.cfm?id=1083592.1083616`.

V. Gupta and G. S. Lehal. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268, 2010.

Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 207–214, New York, NY, USA, 2007a. ACM. ISBN 978-1-59593-597-7. doi: 10.1145/1277741.1277779. URL `http://doi.acm.org/10.1145/1277741.1277779`.

Q. He, K. Chang, E.-P. Lim, and J. Zhang. Bursty feature representation for clustering text streams. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 491–496. SIAM, 2007b.

A. Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

G. Ifrim, B. Shi, and I. Brigadir. Event detection in twitter using aggressive filtering and hierarchical tweet clustering. In *Second Workshop on Social News on the Web (SNOW), Seoul, Korea, 8 April 2014*. ACM, 2014.

S. Ji, H. Yun, P. Yanardag, S. Matsushima, and S. Vishwanathan. Wordrank: Learning word embeddings via robust ranking. *arXiv preprint arXiv:1506.02761*, 2015.

E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed 2017-03-07].

M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pages 31–39. Citeseer, 2014.

N. Keane, C. Yee, and L. Zhou. Using topic modeling and similarity thresholds to detect events. In *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT*, pages 34–42, 2015.

S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

M. J. Kusner, Y. Sun, N. I. Kolkin, K. Q. Weinberger, et al. From word embeddings to document distances. In *ICML*, volume 15, pages 957–966, 2015.

Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920. Association for Computational Linguistics, 2010.

H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. Association for Computational Linguistics, 2004.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013a. URL `http://arxiv.org/abs/1301.3781`.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Hlt-naacl*, volume 13, pages 746–751, 2013c.

O. Mogren, M. Kågebäck, and D. P. Dubhashi. Extractive summarization by aggregating multiple similarities. In *RANLP*, pages 451–457, 2015.

A. Nenkova and K. McKeown. A survey of text summarization techniques. In *Mining text data*, pages 43–76. Springer, 2012.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979. ISBN 0408709294.

Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

J. Straková, M. Straka, and J. Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P14/P14-5003.pdf`.

M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 131–142, New York, NY, USA, 2004. ACM. ISBN 1-58113-859-8. doi: 10.1145/1007568.1007586. URL `http://doi.acm.org/10.1145/1007568.1007586`.

Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 28–36, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5. doi: 10.1145/290941.290953. URL `http://doi.acm.org/10.1145/290941.290953`.

Appendix A

Real events used for evaluation

This is a list of confirmed events which occurred in 2014 that was used to evaluate Precision and Recall.

#	Date	Headline
1	June 2	Španělský král Juan Carlos I. abdikoval a za svého nástupce určil svého syna Filipa.
2	June 4-5	Konal se 40. summit G8 v Bruselu.
3	June 7	Petro Porošenko složil prezidentskou přísahu a stal se prezidentem Ukrajiny.
4	June 8	Abd al-Fattáh as-Sísí složil prezidentskou přísahu a stal se prezidentem Egypta.
5	June 10	V izraelských prezidentských volbách byl zvolen Re'uven Rivlin.
6	June 12	V Brazílii začalo 20. mistrovství světa ve fotbale.
7	June 15	Andrej Kiska složil prezidentskou přísahu a stal se prezidentem Slovenska.
8	June 18	Vůdci vojenského převratu v Turecku z roku 1980 Kenan Evren a Tahsin Şahinkaya byli odsouzeni na doživotí.
9	June 19	Filip, asturský kníže složil přísahu a stal se králem Španělska jako Filip VI. Španělský
10	July 1	Itálie se ujala předsednictví EU.
11	July 8	Armáda České republiky utrpěla největší ztrátu v novodobých dějinách, kdy při sebevražedném útoku poblíž letecké základny Bagram zemřeli čtyři čeští vojáci, spolu s dalšími 12 tamními obětmi. Pátý český voják byl těžce raněn a 14. července zemřel.
12	July 13	Mistry světa ve fotbale se stala německá fotbalová reprezentace.
13	July 15	Novým předsedou Evropské komise se stal lucemburský politik a bývalý premiér Jean-Claude Juncker.
14	July 17	V oblasti bojů na východní Ukrajině se zřítil Boeing 777 malajsijských aerolinií. Zemřelo všech 295 osob na palubě.

15	July 21	Vláda Bohuslava Sobotky vybrala nového eurokomisaře. Stane se jím ministryně pro místní rozvoj Věra Jourová, která ve výběru porazila Pavla Mertlíka.
16	August 10	V historicky první přímé prezidentské volbě v Turecku byl zvolen premiér Recep Tayyip Erdoğan.
17	August 16-28	Letní olympijské hry mládeže 2014 v čínském Nankingu.
18	August 19	Americký novinář James Foley byl popraven v syrské poušti neznámým islámským radikálem, jeho smrt vyvolala v západním světe vlnu pobouření.
19	August 24	Meziplanetární sonda New Horizons prolétla blízko L5 soustavy Slunce–Neptun.
20	August 25	Ve sporu o amnestii Václava Klause soud schválil smír, podle něhož se bývalý hradní právník Pavel Hasenkopf na vyhlášeném znění amnestie nepodílel.
21	August 28	Recep Tayyip Erdoğan složil prezidentskou přísahu a stal se prezidentem Turecka.
22	August 30	Polský premiér Donald Tusk byl na summitu Evropské unie zvolen předsedou Evropské rady.
23	September 1	Pavel Hasenkopf podal na Vratislava Mynáře trestní oznámení pro pomluvu ohledně Mynářova výroku, že Hasenkopf je jedním z autorů amnestie Václava Klause.
24	September 2	Další americký novinář Steven Sotloff byl popraven v syrské poušti neznámým islámským radikálem, stejně jako James Foley v srpnu.
25	September 4	Ve Vilémově se zřítil most, na kterém probíhala rekonstrukce. Zemřeli čtyři dělníci, další dva byli zraněni.
26	September 6	Počet nakažených ebolou při celoroční epidemii se přehoupl přes 4 000.
27	September 8	Britský následník trůnu Princ William a jeho manželka Kate oznámili, že čekají druhé dítě.
28	September 10	Kandidátka na českou eurokomisařku Věra Jourová získala portfolio spravedlnosti, spotřebitelské politiky a rovnosti pohlaví.
29	September 13	Islámští radikálové popravili dalšího západního zajatce, tentokrát jím byl britský humanitární pracovník David Haines.
30	September 18	Ve Skotsku proběhlo referendum o nezávislosti na Spojeném království. Pro odtržení od Británie hlasovalo 44,7% lidí, proti 55,3% lidí, Skotsko tak zůstane její součástí.
31	September 20	Náčelník Generálního štábu Armády ČR Petr Pavel byl zvolen předsedou vojenského výboru NATO.

32	September 24	Na Pražský hrad se dostal výhružný dopis adresovaný prezidentovi Miloši Zemanovi s bílým práškem. Případ šetří policie.
33	October 3	Prezident Miloš Zeman přijal demisi ministryně pro místní rozvoj Věry Jourové.
34	October 3	Islámští radikálové popravili dalšího západního zajatce, stal se jím opět britský humanitární pracovník Alan Henning.
35	October 7	Evropský parlament schválil nominaci Věry Jourové na post eurokomisařky pro spravedlnost, spotřebitelskou politiku a rovnost pohlaví.
36	October 10-11	Proběhly volby do Senátu Parlamentu České republiky, volby do zastupitelstev obcí a volby do Zastupitelstva hlavního města Prahy. Ve volbách uspěly především vládní strany ČSSD, ANO a KDU-ČSL.
37	October 14	Žena trpící schizofrenii pobodala na obchodní akademii ve Žďáru nad Sázavou tři studenty a zasahujícího policistu. Jeden ze studentů útok nepřežil.
38	October 16	Ve Vrběticích došlo k výbuchu muničního skladu č. 16. Na místě zahynuli dva zaměstnanci skladu, došlo k evakuaci obyvatel přilehlých obcí.
39	October 16	Zanikla europarlamentní frakce Evropa svobody a přímé demokracie, 20. října byla opět obnovena.
40	October 17-18	Proběhlo druhé kolo voleb do Senátu Parlamentu České republiky. Ve volbách uspěly především vládní strany ČSSD, ANO a KDU-ČSL.
41	November 9	V Katalánsku začalo symbolické hlasování o nezávislosti na Španělsku.
42	November 12	Přistál modul Philae jako historicky první lidský stroj na kometě.
43	November 15	Islámští radikálové popravili dalšího západního zajatce, stal se jím americký humanitární pracovník Peter Kassig.
44	November 15-16	Summit G20 v Brisbane.
45	December 1	Ledovková kalamita ochromila hromadnou dopravu v ČR a dodávky elektřiny v mnoha regionech. Tramvajová doprava v Praze dokonce poprvé ve své historii zažila úplné zastavení provozu. Do normálu se dopravní i energetická situace vrátila až 3. prosince.
46	December 1	Druhým předsedou Evropské rady se stal Donald Tusk.
47	December 3	Ve Vrběticích došlo k dalšímu výbuchu muničního skladu č. 12. Opět proběhla evakuace obyvatel přilehlých obcí, oba dva výbuchy jsou vyšetřovány jako úmyslný trestný čin.

48	December 16	Ozbrojenci ze skupiny Tahrík-e Tálibán-e Pákistán spáchali masakr v péšávarské vojenské škole škole. Útok si vyžádal 141 obětí většinu z nich tvořili děti.
49	December 28	Na cestě ze Surabaje do Singapuru se ztratilo letadlo malajsijské společnosti AirAsia se 162 lidmi na palubě.

Appendix B

Annotated events

Below are several examples of events detected by the three methods. For each method, we provide examples of Well-formed event summaries as well as those of poor quality. The full outputs for all detected events can be found on the DVD. These examples serve purely to illustrate the various events found using the individual methods.

For each event, the first few keywords are shown to save space. Then, for each event burst, we list its date, headline of the most similar document, and the generated summary with a maximum length of 60 words.

B.1 Original method

Well-formed events

1. let, mha
Mar 11 - Apr 2, 2014
Pátrání po malajském boeingu pokračuje Což měl údajně být i případ jednoho z lidí v malajsijském letu MH 370. Jsou však značně vzdálené od plánované trasy letu MH 370. Neřekl jsem, že to byl let MH 370. Záhada letu MH 370 pokračuje. Záhada letu MH 370 tak pokračuje. Záhada letu MH 370 trvá.
Jul 18 - Jul 28, 2014
Na Ukrajině byl nejspíš sestřelen malajsijský boeing, více než 300 obětí Zpravodajství o tragickém letu MH 17 najdete v listě. Sestřelení letu MH 17 nadále sledujeme online. Redaktor CNN ukazuje dráhu letu MH 17. Byl MH 17 skutečně sestřelen? Na palubě letu MH 17 bylo podle malajsijského ministra dopravy 298 lidí. Sestřelení letu MH 17 by mu tedy takovou cestu mohlo otevřít.
Aug 14, 2014 - Jan 5, 2015
První zpráva o tragédii letu MH 17 na Ukrajině bude v září První zpráva o tragédii letu MH 17 bude v září. Boeing MH 370 také několikrát prudce změnil výšku. Bratr mi zmizel při letu MH 370, dcera zahynula v MH 17, pláče. Ukrajinská armáda prokazatelně BUKy měla a byly v dobu letu MH 17 aktivní. Separatistka se nalíčila řasenkou oběti letu MH 17.

2. KDU-ČSL, lidovec

Jan 4 - Feb 22, 2014

Koalice Náměstci, třeste se!
Vašimi koaličními partnery jsou ANO a KDU-ČSL. Vláda bude mít 17 členů. Již před Vánoci zveřejnili své nominanty lidovci. Kandidáti hnutí ANO a KDU-ČSL na ministry jsou již známi. KDU-ČSL je stranou, která se často ohání tradicí. ANO i KDU-ČSL budou mít po jenom vicepremiérovi. ANO: Lidovci, rozhodněte se!

May 16 - Jul 27, 2014

Eurovolby 2014 s Janem Kellerem (ČSSD) a Pavlem Svobodou (KDU-ČSL)
KDU-ČSL zveřejnila program, se kterým jde do voleb. Proti návrhu je opozice i lidovci. Má podporu zelených a lidovců. To je ale podle lidovců málo. Nové slevy na druhé dítě jsou podle lidovců slabé. Pro lidovce je to však málo. Lidovci nepodpoří Babišovy daňové změny. Lidovcům se to ale zdá málo.

Oct 11 - Oct 31, 2014

SedmiDeník Uplynulý týden očima Kateřiny Perknerové
KDU-ČSL kandidují v hlavním městě v Trojkoalici. Zvítězila zde jediná volební kandidátka KDU-ČSL. Věří, že lidovci v obou hlasováních posílí. Lidovci kandují ve dvaceti volebních obvodech. Kandidáti KDU-ČSL vítězili ve třech obvodech, ve dvou pak vedou společní kandidáti lidovců a Strany Zelených. Po sečtení poloviny hlasů vede v komunálních volbách KDU-ČSL.

Poor quality events

1. bodovat, děkovat

Oct 19 - Dec 31, 2014

Beránek Začali jsme hrozně. Nevěřil jsem vlastním očím
Hemingway bar neboduje jen ve světě. Odpověď zní "Ne děkuji. V anketě bodovalo divadlo všech žánrů. Pardubice naopak počtvrté za sebou nebodovaly. Východočeši nebodovali už počtvrté za sebou. Réway: Za výhru můžeme děkovat Rasťovi. Mají sílu a teď několikrát za sebou bodovali. JESENÍK BODOVAL V SOUTĚŽI ELEKTROOSKAR.

2. Čína, čínský

Apr 19 - Jun 11, 2014

Po přidružení k EU se Ukrajina stane jen vývozcem pracovní síly...? (Břetislav Olšer)
Nad Čínou se stahují mračna. Na vině je dovoz z Číny. Výsledek je tedy čínskou kopii jiné čínské kopie amerického vozu. McLarenu se v Číně nedaří. V Číně začíná mezinárodní autosalon. Prodej aut v Číně dál poroste. V Číně v sobotu začal mezinárodní autosalon. Fiat bude vyrábět modely Jeep v Číně.

Oct 10 - Dec 23, 2014

Nobelovu cenu za literaturu získal francouzský spisovatel Patrick Modiano
Ta bude pod čínskou dominancí a řízená z Pekingu. Vháníme Rusko do náruče Číny. Komunismus je i v Číně, proto je špatná. A doplatí na to výrobci. Ti všichni věří v potenciál růstu čínského akciového trhu. Tento problém už ale bude řešit čínská strana. Ta se v Číně odehraje již dnes.

B.2 Embedded-greedy method

Well-formed events

1. sankce, konflikt, Putin, Moskva, západ, východ, armáda, vojenský...

Jul 26 - Sep 15, 2014

Moskva a NATO se hádají o ruské vojáky na Ukrajině
Demonstranti v Moskvě žádali Putina o vojenský zásah na Ukrajině. Merkelová: NATO posílí na východě svou vojenskou přítomnost. Vladimir Putin dohlíží na vojenské manévry. Putin chce posílit armádu kvůli údajné hrozbě Západu. Putin prověřuje bojeschopnost armády na Dálném východě. NATO pořádá manévry na západě Ukrajiny, Moskva protestuje.

2. předplatné, vložit, Václav, Havel, číslo, náměstí, pražský, výročí...

Oct 26 - Dec 15, 2014
Knihovna Václava Havla slaví 25. výročí sametové revoluce
Pro mě to nebyla až taková událost. Hodnotí nejsoučasnější události v Evropě. StB navrhovala zabít Václava Havla. Knihovna Václava Havla slaví 25. výročí sametové revoluce. Oslavy sametové revoluce pohlídají v Praze stovky policistů. Kroměříž si na náměstí připomene výročí revoluce i Karla Kryla. Psali jsme: Čas pracuje pro Václava Havla.

Poor quality events

1. únor, březen, loňský, vloni

Feb 1 - Mar 25, 2014

V březnu
Co se stalo 20. února? Při loňské razii u Nagyové našla policie šperky za miliony. Tak máme pomalu konec února, jemuž je autor těchto řádků zvyklý říkat úmor. Vyšlo nové vydání Solidarity na únor/březen. Podle nejnovějších zpráv (z 1. března) město Brno povolilo výstavbu skladu Amozonu.

2. útok, oběť, zemřít, boj, těžký, dítě, skončit, Německo...

Apr 2 - Nov 4, 2014

Příměří v Gaze vydrželo dvě hodiny, Izrael pokračuje v bojích. Zemřelo 27 Palestinců
Bakterie mají hodiny, které je probudí, když skončí útok antibiotiky. Bakterie mají hodiny, jež je probudí, když skončí útok antibiotiky. Posádka druhého vozu skončila s těžkými zraněními v nemocnici. Těžká nehoda autobusu: Řidič zemřel, těhotná žena skončila popálená! V nemocnici skončilo 33 lidí, tři z nich jsou děti.

Jan 6 - Jan 26, 2015

OBRAZEM Francie vzpomíná na oběti útoků
Podle informací deníku Le Monde zemřelo 12 lidí. Zemřelo nejméně 12 lidí, z toho dva policisté. Obama se poklonil obětem pařížského útoku. Útočníci byli Francouzi, narození ve Francii. Část z nich v bojích padla, ale asi 180 radikálů se vrátilo do Německa. Boj proti terorismu: Francie se zbláznila.

B.3 Cluster-based method

Well-formed events

1. Gaza, Hamas, Izrael, Izraelec, Palestinec, izraelský, palestinský

Apr 28 - Sep 20, 2014

Izrael vs. Hamás 3.0
Palestinský Hamas oznámí příměří s Izraelem. Izraelci zaútočili na deset cílů v palestinském Pásmu Gazy. Izraelské nálety na Gazu zabily devět Palestinců. Izraelské tanky vjely do Gazy. Hamas prý v pásmu Gazy zadržel izraelského vojáka. Podle Izraele padl do rukou palestinského Hamasu. Izraelci při útoku na Gazu zabili tři velitele Hamasu.

Dec 19 - Dec 21, 2014

Izrael v pásmu Gazy zaútočil na Hamas
Izrael v pátek odpoledne oznámil, že Palestinci odpálili z Gazy raketu Kásam. GAZA - Izraelská armáda provedla letecký úder v pásmu Gazy, jehož cílem bylo palestinské hnutí Hamas. Autor: ČTK. Gaza - Izraelská armáda provedla letecký úder v pásmu Gazy, jehož cílem bylo palestinské hnutí Hamas. Přesto s Hamásem nic neudělá.

2. Ebola, Guinea, Leone, Libérie, Sierra, ebola, epidemie

Sep 28 - Oct 29, 2014

Epidemie eboly
Ebola nejdříve propukla v Guineji. Nejvíce postižené země jsou Sierra Leone, Guinea a Libérie. Libérie je spolu se Sierrou Leone a Guineou nejpostiženější ze západoafrických zemí. Zatím epidemie eboly hlavně postihovala východ Sierra Leone. Libérie patří k zemím nejvíce zasaženým epidemií eboly. WHO: Epidemie eboly v Libérii začala ustupovat.

Poor quality events

1. extra, podnik, reklama, siga, zpravodajství

Oct 10 - Dec 3, 2014

Maturus o. p. s.
Jaké reklamy na vás působí nejvíce? Reklama na mě nemá téměř žádný vliv. Televize Barrandov v listopadu rozšíří zpravodajství. Některé reklamy jsou hodně odvážné. Jaké se ti líbí reklamy? 4. Jaké se ti líbí reklamy ? 9. Zajímavá jsou zjištění ohledně televizní reklamy. Tyto reklamy se mi líbí. 3.

2. Rusko, ruský

Feb 21 - Apr 24, 2014

RUSKÝ POSTKOLONIALISMUS - 1.
Jaké jsou skutečné možnosti Ruska? Rusko potřebuje investory jako sůl. Na budově zavlál prapor Ruska. Tak to je pro Rusko dárek! Lid žádá návrat do Ruska. Rusko, největší stát světa, se může radovat. Tím se dostala i západní Ukrajina pod ruskou vládu. Rusko nemělo evidentně zájmy ve střední Evropě.

Jul 28 - Sep 8, 2014

Hurá na Rusko
Bát bychom se neměli Ruska, ale USA a EU. Rusko se vyhnulo klasickému otroctví. Za Janukoviče byly zájmy Ruska v bezpečí. Upozornil na to ruský server newsru.com. Rusko se obává, že se EU zřekne ruských raket. Válčí Rusko na Ukrajině? Vše na ruské straně proběhlo dobře.

Nov 30 - Jan 8, 2015

Rusko
Ne že by na to Rusko nemělo. Reálně není Rusko nikým vojenský ohrožováno. Odvolejme naše sankce proti Rusku! Zhruba v polovině léta se k ní přidal i ruský rubl. Rusko nejen přežije, ale bude mnohem silnější, řekl. Ruská centrální banka zahájila intervence na posílení rublu. Rusko tato nařčení opakovaně odmítlo.

Appendix C

DVD contents

The attached DVD contains the Python scripts used for experimentation, logs and outputs of the event detection. Additionally, we provide the preprocessed data for running the event detection. Due to space limitations, we do not include the whole document collection in text format. Instead, we provide the serialized matrix **B** defined in Section 4.1 and the relevant documents retrieved for all events. This will allow to run the detection with the settings as defined in the scripts. In addition, we provide the trained Word2Vec model.

To run, the project requires Python 3.6, NumPy 1.11.3, SciPy 0.18.1, Scikit-Learn 0.18.1, Matplotlib 2.0.0, Wordcloud 1.2.1 and Gensim 1.0.1.

The DVD structure is as follows:

- /logs – Log files produced during the experiments with information about running times.
- /plots – Trajectories of all detected events.
- /scripts – Python scripts used for experimentation.
  - /scripts/event_detection – The main project files. Run the detection through the main.py file.
  - /scripts/notebooks – Jupyter notebooks used for testing.
  - /utils – Simple utility scripts used to collect word counts and train the Word2Vec mode
- /summaries – Summaries of all detected events.
- /thesis – Source codes for this thesis and its PDF version.
- event_detection_data.zip – Input data and trained Word2Vec model compressed into a ZIP format. Unzip to /scripts/event_detection before running the project.