POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Extractive Timeline Summarization based on Unsupervised Techniques

Supervisors

Prof. Luca CAGLIERO

Dr. Moreno LA QUATRA

Candidate

Stefano MUNNA

December 2020

Abstract

With the increasing importance of internet during time, a huge amount of news article about several arguments are published in several websites: users that are interested in a particular event could use an automatic system that presents to him the most important aspects of the event and/or its evolution across time. These two problems are addressed by:

- Multi-document Summarization, with the aim of condensing news from several articles in a complete and non-redundant summary
- Temporal Summarization, with the aim of keeping the user informed about the developing of an event,providing non-redundand and significant updates as soon as new articles arrives (or at prefixed time intervals)
- Timeline Summarization, with the aim of providing the end-user with the history of a concluded event along a timeline that highlights the most significant dates for the event and the most significant happenings for each date.

We analyzed the State-of-the-Art in text, temporal and timeline summarization presenting several algorithms that have been proposed in the past years: the aim of this thesis is to create a framework that performs timeline summarizzation following a pipeline (date selection,date summarization,timeline visualization), extracting, from an input set of document about a specific topic, the most important dates and then applying text summarization to select the most important aspects (sentences) for each date. The aim is to explore the performance of several text summarization algorithms used in the date summarizzation step (in which a summary is extracted, for each date, from the input sentences associated to that date), comparing at the end the results obtained by testing the several algorithms on CRISIS and T17 datasets and evaluating the output timelines using some ROUGE variants. To ease the exploration of the generated summaries, we developed a Web application tailored to date summary visualization. It provides a visual extract of the summary content together with a representative image crawled by Google. It constitutes the last block of the pipeline,aimed at providing end-user with a more friendly tool to explore the timeline generated by the system.

Acknowledgements

Table of Contents

List of Tables VI

List of Figures VIII

Acronyms X

1 Introduction 1

2 Preliminaries 5

3 Related Works 12
- 3.1 Text Summarization 13
- 3.2 Temporal Summarization 30
- 3.3 Timeline Summarization 35

4 Timeline Summarization Framework 43

5 Visual Summary Exploration 51
- 5.1 Possible extensions of the visual interface 54

6 Experimental Results 55
- 6.1 Evaluation metrics 55
- 6.2 Datasets description 58
- 6.3 T17 results 59
- 6.4 CRISIS Results 65

7 Conclusion and Future Work 72
7.1 Future works 74

Bibliography 76

List of Tables

6.1 Results obtained testing algorithm on T17 and evaluating results using concat-Rouge-1. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result. 60
6.2 Results obtained testing algorithm on T17 and evaluating results using concat-Rouge-2. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result. 61
6.3 Results obtained testing algorithm on T17 and evaluating results using agreement-Rouge-1.* indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result. 62
6.4 Results obtained testing algorithm on T17 and evaluating results using agreement-Rouge-2. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result. 63
6.5 Results obtained testing algorithm on T17 and evaluating results using Date-content alignment many to one (align+ m:1) ROUGE 1. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result. 64
6.6 Results obtained testing algorithm on T17 and evaluating results using Date-content alignment many to one (align+ m:1) ROUGE 2. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result. 65

6.7 Results obtained testing algorithm on CRISIS and evaluating results using concat-Rouge-1. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result. 66
6.8 Results obtained testing algorithm on CRISIS and evaluating results using concat-Rouge-2. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result. 67
6.9 Results obtained testing algorithm on CRISIS and evaluating results using agreement-Rouge-1.* indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result. 68
6.10 Results obtained testing algorithm on CRISIS and evaluating results using agreement-Rouge-2. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result (We considered TextRankBM25 as the best performing in terms of Precision and F-Measure). 69
6.11 Results obtained testing algorithm on CRISIS and evaluating results using Date-content alignment many to one (align+ m:1) ROUGE 1. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result (We considered TextRankBM25 as the best performing in terms of F-Measure). 70
6.12 Results obtained testing algorithm on CRISIS and evaluating results using Date-content alignment many to one (align+ m:1) ROUGE 2.* indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result (We considered TextRankBM25 as the best performing in terms of F-Measure). 71

List of Figures

4.1 Timeline summarizaton pipeline of the proposed system 44
4.2 For each important date,the framework follow the illustrated steps to select, among the input sentences associated to the current date, the sentences that will form the summary for that date,using for each date the same summarization algorithm. 46
4.3 Graphical representation of the framework: for each important date,extracted in the date selection step, the same algorithm ALG is applied to create the final timeline by extracting sentences from the input ones using the date summarization block of Figure 4.2 50

5.1 textual file containing the list of sentences extracted for each date. 51
5.2 homepage of TimelineVisualization 52
5.3 final timeline visualization 53
5.4 popup showing the finanl summmary extracted for the selected date 53

Acronyms

NLP
Natural Language Processing

NLG
Natural Language Generation

DUC
Document Understanding Conference

TS
Temporal Summarization

TLS
Timeline Summarization

TF-IDF
Term Frequency - Inverse Document Frequency

KLD
Kullback Leibler divergence

LDA
Latent Dirichlet allocation

IR
Information Retrival

MDS

Multi-document Summarization

MSC

Multi-Sentence-Compression

AP

Affinity Propagation

SVM

Support Vector Machines

SVD

Singular Values Decomposition

EHDP

evolutionary Hierarchical Dirichlet process

TDT

Topic Detection and Tracking

CBSU

Cluster-based sentence utility

CSIS

Cross-sentence informational subsumption

CBS

Centroid-based Summarization

SOTA

State of the Art

Chapter 1

Introduction

Due to the increasing importance of Internet over the last decades, people have the possibility to access a large amount of news articles about several topics. The readers interested in a specific topic will be forced to read several articles to have a complete overview of specific events. It will be useful to provide readers with a concise summary that highlights the most important aspects of the event he is interested in by removing redundant information. Addressing this problem is the main task of automatic text summarization.

Text summarization is a branch of Natural Language Processing that comprises extractive techniques, aimed at extracting the most important sentences from the input documents and abstractive ones, whose goal is to generate new content according to end-user needs, combining sentences of the original text. Text summarization can be divided into several subtasks: Single-Document Summarization techniques, whose aim is to generate a shorter version of a document, and Multi-document Summarization techniques, whose goal is to condense information about a topic of interest from different documents (articles) into a short summary that cover all the important information contained in the original documents. Since the input documents typically evolve over time, text summarization has evolved into new tasks, namely temporal summarization and timeline summarization.

Temporal summarization aims at providing the user of updates about the developing of an event by selecting, as soon as new articles are

available or at specific time intervals, sentences from these articles that are relevant with respect to the event of interest and that are not redundant with respect to the already emitted updates.

Timeline summarization, instead, aims at summarizing the key information about an event by creating a timeline including the most significant dates and content for each date.

In this thesis we propose a new pipeline for timeline summarization task relying on a date-wise approach. Unlike the other two approaches, direct summarization (that consider the input as a unique set of dated sentences among which the most important ones are extracted along with their dates) and event detection (in which sentences are clustered based on events and dates and then the most representative sentences are extracted from the most important clusters) the proposed method consists of two main steps: the date selection step, which selects the most important dates, and the date summarization step, which extracts the important sentences from the text of the selected date. The framework takes as input a set of timestamped news articles ranging over a specific topic and extracts the most relevant sentences describing the key events occurred on the considered dates. The thesis work focuses on the overview of several state-of-the-art methods for date selection and news article summarization.

The "Date Selection" step receives as input a stream of documents annotated with their publication date and associates to each sentence, extracted from the input stream, a date that is the publication date or it is computed exploiting temporal reference in the sentence and using several tools (we used heideltime) to obtain the final date: then the importance of each date is computed: state-of-the-art methods use several criteria such as mention count, number of times the sentence is mentioned, or publication date, number of articles/sentences published on that date: our framework exploits techniques that are not object of this thesis. The "Date summarization" step receives as input the important dates, extracted in the previous step, and iterates each date to extract the important sentences among the ones associated to the current date: in first place, too short sentences and duplicated sentences are filtered out and the remaining sentences are processed.

The "Date summarization" step integrates various state-of-the-art text summarization algorithms: graph-based algorithms, such as the popular Text Rank and LexRank algorithms, the CoreRank algorithm, which relies on core decomposition to extract the most salient keywords from the input sentences, and clustering-based approaches; finally, we also integrates some methods based on Maximal Marginal Relevance: the main contribution of this thesis is exploring the performance of the employed methods and make a comparison between them. Another contribution of this thesis work concerns the visualization of the automatic timelines. To this purpose, we built a Web application, called Timeline Visualization, that takes as input the output timeline and processes the file to provide a visual explanation of the result: the timeline consists of several entries that contain the important dates and a preview of the summary for that date, which can be shown in a separate window containing the entire summary and a representative image. The proposed picture is obtained querying Google using the corresponding summary. We tested our methods on two benchmark datasets: Timeline17, which is a corpus of document constructed starting from 17 different timelines, and CRISIS, which is a corpus of documents covering several crises such as war and revolutions. Authors in both cases retrieved the articles that compose the datasets querying the Google search engine: for the Timeline17 corpus authors made a research based on the topic or event described by each timeline, while for CRISIS authors made a document research based on events (crisis, war, revolution) and locations (Syria, Egypt, Libya, Yemen), filtering out the retrieved documents that were not in the temporal range of any initial timeline. We evaluated the quality of the summarization process using the standard Rouge metrics. The results show that graph-based algorithms were the best performing, in particular the TextRank (employed along with BM25) algorithm was the best performing, on both datasets, among all the text summarization algorithms, according to the most part of used metrics, but also CoreRank and cluster-based algorithms obtained good performance according to some of the used metrics, in particular the algorithms that exploit k-means clustering. The future developments of this work could move to the employment of other unsupervised text summarization

algorithms based on the latest advancements of deep language models, trying to improve the performance of the already implemented ones. As future extension we could integrate into the date summarization step some supervised techniques. Those methods could exploit default pre-trained models or can exploit human annotated data to fine-tune the models for the summarization objective. Another possible extension of the system could involve the selection of images from the original dataset, providing them as input for the implemented web application. It could be modified to accept a separate file including relevant images for each selected date. In conclusion we can assert that we proposed a system for timeline summarization whose performance are fairly good and strongly depends on the algorithms used in the summarization phase; after exploring and comparing performances of the employed algorithms we identified TextRankBM25, i.e. the summarizer that exploit the graph-based TextRank algorithm integrating BM25 as similarity measure, as the algorithm that performed significantly better than all the other algorithms according to the most part of exploited evaluation metrics. The system could potentially benefit from the integration of new deep learning methodologies. Finally, we provide users with a user-friendly way of graphically exploring the timeline content.

Chapter 2

Preliminaries

With the passing of years the Internet has assumed an increasingly important role in every day life: people have now the possibility to access a huge set of information about several topic, written by several authors and published in several websites.

It would be useful for the reader, that is interested in a specific topic, to have several information and news about that specific argument gathered into a summary that is short enough to allow the user not to waste a lot of time in reading several articles and at the same time detailed enough to illustrate to the reader the complete development of the event that he is interested in. This is the aim of **Natural Language Processing (NLP)** and in particular of **Text Summarization (Document Summarization)**.

NLP is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data. Challenges in natural language processing frequently involve:

- **Speech recognition:** also known as Speech to Text (STT), it consists in recognition and translation of spoken language into text by computer.

- **Natural language understanding:** it deals with machine Reading comprehension (the ability to process text, understand its meaning, and to integrate with what the reader already knows) and finds applications in automated reasoning, machine translation from a language to another,question answering,news-gathering, text categorization and others.

- **Natural language generation (NLG):** it is a software process that transforms structured data into natural language. It can be used to produce long form content for organizations to automate custom reports, as well as produce custom content for a web or mobile application. It can also be used to generate short blurbs of text in interactive conversations (chatbot) which might even be read out by a text-to-speech system.

Among the tasks involved into NLG one of the most important is Text Summarization, starting point of this thesis.

Summarization is the task of condensing a piece of text to a shorter version preserving key informational elements and the meaning of content [1]: since in the big data era, given a specific topic, we can find an inestimable volume of information about that specific topic, automatic text summarization is assuming a very important role. The aim of Text Summarization is to extract a summary from one ore more input documents avoiding redundancy: text summarization processes the input dataset, i.e. a collection of documents about several topics, to obtain the final summary for a specific topic of interest: the **document** in text summarization is treated as a set of **sentences** that talks about the same topic, identified as the topic of the entire document: the input dataset could condense several collection of document, where a **collection** is a set of document speaking about the same topic. The aim of text summarization is to extract a summary given a specific topic: a **summary** is a short representation of the input text, which provide a shorter version of a text, extracting the most salient passages, or that condense information about the topic of interest extracted from different sources, maintaining the informational content and semantic

meaning of the original text while avoiding time consuming redundancies. To obtain the summary typically intermediate representation of sentences are computed: these representations see sentences as set of **terms**, from which are excluded most recurrent terms that don't add significant information to sentence representations, such as conjunctions, articles and punctuation, classified as stopwords. Usually the documents employed in text summarization are news articles and a collection is a stream of news article about a specific topic.

We can identify two macro subsets of Text Summarization:

- **Single-Document Summarization**: the input of all single document summarization algorithms is a set of sentences extracted from the same input document: the aim is to automatically generate a shorter version of the input document that contains the same information of the original text, or to automatically generate an abstract for the input document selecting the most significant sentences avoiding information repetition.
- **Multi-document summarization**: in this case the input is a set of sentences extracted from several documents about the same topic: the aim is to put in a single summary the information that can be found in several documents or articles, obtaining in this way a shorter text that contains information of several articles (so points of view and opinions of different authors about the same event) that cover all the most important sub-event about the event of interest, maintaining the information richness of the input text but decreasing searching and reading time avoiding redundancies in the final summary.

We can distinguish two approaches to text summarization:

- **Extractive Summarization:** sentences are directly extracted from the original text: it can be summarized in the following three steps:
  1. the original text is represented as a set of sentences that are transformed into their intermediate representation: typically a sentence is represented as a vector;

2. assign a score to each sentence (the way the score of each sentence is computed depends on the used algorithm, but usually sentence is represented as a set of terms and each term is a vector cell whose value is based on term frequency in the document or in the input collection);
3. extract the top-k most scored sentences that will form the summary (also in this case the score of each sentence is computed in several way, depending on the used algorithm).

In short, the summmary is generated by copying sentences from the original text,rearranging them to obtain the final summary[1]. In this way sentences could be extracted from different part of the input text or from different document so they are likely to be totally unrelated, causing the final summmary to appear grammatically incorrect, but this is the most simple approach to text summarization.

- **Abstractive Summarization:** Abstractive methods build an internal semantic representation of the original content, and then use this representation to create the final summary, with the goal of generating new content according to end-user needs.Abstraction may transform the extracted content by combining sections of the source document, to condense a text more strongly than extraction. This transformation, however, is computationally much more challenging than extraction but the final summary will look like a human-written summary. In short, abstractive summarization systems generate new phrases, combining input sentences and using words that were not in the original text [1], so sentences of the final summary probably will be combination of the input sentences that will appear related to each other.

Whatever the approach to text summarization is, the final summary should have some features to be considered a good summary [2]:

- it should be the more human readable as possible, so it should not contain datelines, system internal formatting or ungrammatical sentences;

- it should not contain unnecessary repetition of entire sentences or of noun where pronouns could be used to refer to a previously used noun
- nouns or pronouns references should be clear
- sentences must be related to each other in terms of informational content (as previously seen is unlikely to see sentences grammatically related if the extractive approach is used)
- summary should be well structured and organized and must provide a complete and informationally coherent representation of the topic of interest

Several algorithm that approach text summarization in several way have been presented in past years: among them we can find the graph-based algorithms that organize sentences in a graph and find relationship between sentences based on their similarity as we will see in TextRank [3],CoreRank [4], or LexRank [5]; cluster-based algorithm perform instead text summarization clustering sentences based on their similarity or topic and then selecting the most representative sentencs, as we will see more in detail in [6] and [7], or clustering by topic and summarizing simoultaneously, as in [8] and [9]; other techniques have been exploited such as MMR-based ([10]) or LSA-based ([11], [12], [13]) or text-mining based, in particular itemset-mining-based, such as ItemSum [14], MWISum [15] and ELSA [16]. We will get more in detail in the next chapter, in which we make a discussion about the SOTA methods and a comparison between them. Text summarizzation is moreover exploited in other tasks:

- **Update summarization:** presented in DUC2007, given a topic query and two (or more) set of document A and B related to that query (they may have been obtained through a filtering step) the aim is to extract a summary of A for the reader and then a summary of B assuming that the reader has already read A (so the extracted sentences must not express a concept already expressed in A: the concept of novelty is introduced).

- **Temporal summarization:** introduced in TREC2013, the aim of this task is to keep the reader informed about a specific topic or event across time: given a set of sentences already provided to the reader and a set of new document, the aim is to extract a set of sentences that are relevant to the topic and that have an high level of novelty with respect to the previously extracted sentences, in order to avoid redundancy, as soon as the new document arrives. In this case the summary is a set of already selected sentences that are available to the reader: as the event of interest is not concluded and the reader is interested in its developing the summary is dynamic, i.e. it is updated with new sentences that are extracted from the stream of documents that become available in the meanwhile and are emitted as updates: in this case text summarization is used to identify the most important sentences among new sentences and to identify the ones that are non redundant with respect to the already selected ones. In order to take into account temporal information, sentences are usually annotated with a timestamp which could be the publication date of the article in which the sentence is contained or it is computed through temporal references contained in the sentences: the timestamp is used to check if a sentence refers to a date that is before the last emission date: in this case sentence will probably contain information already described and so it is discarded.

- **Timeline summarization:** the aim of this task is to create a timeline to show to the user the history of an already concluded event: this is the main difference from TS: user is interested in the main sub-event that characterized the developing of the event and so how the event started, how it evolved and how it concluded; for this purpose TLS aims to select dates on which the most significant events occurred and then highlight that events: timeline summarization can so be divided into two sub-task: the first is the date selection, in which text summarization can be used to select the dates that contain the most relevant sentences, and then, for each date, the date summarization sub-task aims to extract the most important sentences using a text summarizzation algorithm.

Also in this case sentences are annotated with a timestamp that in this case identify a specific date in the year,month,day format: from the input sentences are extracted the candidate dates, which are scored according to several criteria based on counting how much times a date is mentioned or how many sentences have been published on that date or other combination or variants of this criteria, obtaining in this way a set of dates and a set of sentences from each date, among which the most important ones, according to text summarization criteria, will be extracted: the final timeline will be composed by a set of date and a short summary for each selected date.Also in this case several algorithms have been presented in the past years: as we will see more in detail in the reserved section, all these algorithms approach the problem in three different ways: treating the dataset as an single set of dated sentences that are extracted using text summarization and whose dates will form the timeline (direct summarization), as in [17], [18], [19]; [20], [21], [22] use clustering to gather sentences into time-constrained event clusters and generate timeline selecting sentences from the most important clusters (event-detection approach); the last aproach is the date-wise approach, that separates date selection and text summarization for each date to generate the final timeline: it is the approach that we adopted in our framework, as we will see in the dedicated chapter.

In this thesis we will present in Chapter 3 several Text summarization, temporal and timeline summarization techniques (that we consider as the related work of the framework that we developed), then in Chapter 4 we will speak about the implemented framework, that performs the timeline summarization task, focusing on the summarization algorithms used in the framework; in Chapter 5 we present "Timeline Visualization", which is a minimal web application used to show the extracted timeline, then we will compare the results obtained by the several summarization algorithms, evaluated with some variants of ROUGE metrics (Chapter 6) and in the last chapter we will summarize our work, draw conclusions and briefly describe some possible extensions to our framework.

Chapter 3

Related Works

As described in the previous chapter, Text Summarization find application in several task, among them we will analyze the Temporal Summarization and Timeline Summarization tasks.

The standard Text Summarizzation fails in the description of unexpected event such as natural disaster or accidents: an event like these requires updates to be emitted during time, as soon as new and relevant information become available, to the reader that is interested in its development. Text summarization fails because even if a good summary is provided in output, if the event is developing, information that become available after the emission of the summary are not provided to the reader. Temporal Summarization (TS) has the aim of following the real-time development of the event: in particular, the goal is to develop systems which can detect useful, new, and timely sentence-length updates about the developing event to return to the end-user [23].

Unlike TS, Timeline Summarization (TLS) is instead a special case of multi-document summarization (MDS), which organizes events by date. Basically, TLS can infact be generated by MDS systems by using summarization algorithms on sentences, extracted from news articles, for every individual date to create a corresponding daily summary: the aim is to show to the reader the development of an event during time, selecting a subset of important dates to be considered as the major points of the timeline and generating a good daily summary for each of

these dates [18], after the event is concluded.

In this chapter we will present several algorithms for Text Summatization, TS and TLS that we consider as the related work for our framework that, as we said, performs the TLS task.

3.1 Text Summarization

It is impossible to present TS and TLS algorithm without before talking about Text Summarization and the several algorithms that perform this task. As previously said, text summarization has the aim to extract an abstract for a document (Single document summarization) or to condense information from different articles in a short summary avoiding redundancy (MDS). In this section we will present several text summarization algorithms, in particular extractive methods and among them we will introduce the algorithms that have been implemented for the text summarization step of the TLS framework, objective of this thesis work.

Analyzing the stat-of-the-art in text summarization, several classes of methods can be identified:

- **Graph-based**: these methods identify text units and consider them as vertices of a graph in which two vertices are connected by an edge if there is a relationship between the text units that they represent to compute the score of each text unit based on the "votes" expressed by neighbours: in text summarization text units are sentences (or terms in some cases) and the relationship between two sentences is their similarity.

- **Cluster-based**: these methods divide sentences into clusters, trying to gather sentences that talk about the same sub-event, relying on similarity between sentences and then extracting centroids or sentences near to centroids of each cluster: differences between cluster based methods consist of the used clusterization algorithm or the method used to compute similarity between sentences.

- **Itemset-based**: these methods consider the input documents

as a transactional dataset in which each sentence is a transaction composed by a set of item (usually terms): the aim of these methods is to identify the frequent k-itemsets, i.e. the set of k items whose support, total number of transactions in which the itemset is contained divided by the total number of transaction, is above a given threshold.

- **LSA-based**: latent semantic analysis is a technique of NLP aimed at identifying relationship between documents (in terms of described concepts) using a matrix that contains words count for each document and using SVD (Singular value decomposition) to reduce the matrix and to obtain a vector representation for each document that will be compared using cosine similarity: in text summarization LSA is tipically aimed at extracting all the latent concept from the input text, gathering sentence based on described concept (using similarity between sentences) and extracting the most representative sentences for each concept (that identify a sub-event or nugget of the original event).
- **Deep learning-based**: these methods exploit deep learning techniques, in particular neural network, and apply them to supervised techniques to predict the importance of a sentence based on the previously trained model, typically using a binary classification model that tries to predict if a sentence is important or not.

The following algorithms belongs to the graph-based methods class: one of the best text summarization algorithm in terms of performance is **TextRank**, introduced by authors of [3]: it is a graph-based algorithm that create a ranking for sentences using a graph in wich score/importance for a vertex (sentence) is not computed through information of the single vertex but combining information of the entire graph (set of sentences); the main concept of TextRank is that the score of a vertex in the graph is computed in terms of number of "votes"/"recommendations" the "neighbour" vertices cast for the vertex in analysis (when a vertex v_1 is linked to v_2 it is casting a vote to v_2). In order to apply this kind of algorithm to text summarization authors define four steps: "identify text units that best define the task" [3], in this algorithm they can be

Keywords or Sentences but we will focus on TextRank for sentences, "identify relations that connect such text units" [3], such as similarity, "iterate the graph based ranking algorithm until convergence" [3], "sort vertices based on their final score" [3]. Tipically graph-based algorithms (such as PageRank) use directed unweighted graph but authors, in order to apply the graph-based algorithm to text, use weighted and undirected graph in which if a vertex v_1 cast a "weighted vote" for v_2, also v_2 cast the same "weighted vote" for v_1. The aim of TextRank for sentences extraction is to identify the most important and "representative" sentences for the given text: for each sentence in the original text a vertex is introduced in the graph; two vertices/sentences S_i and S_j are linked by a weighted undirected edge if the similarity score computed is greater than 0; this similarity score, which will be the weight of the edge, constitutes the vote that S_i casts for S_j and viceversa: it can be computed using overlap of two sentences, i.e. number of common tokens, but also semantic overlap, i.e. sentences express the same concept, or cosine similarity between two sentences representation; after running the ranking algorithm to compute importance of each sentence in the graph as the dumped summation of the neighbour votes, sentences are sorted based on their score and the top-N ranked are selected to form the final summary.

Another graph-based algorithm for Text Summarization is LexRank presented by authors as a graph-based approach to compute sentence importance based on the concept of centrality of a sentence in a graph,in which each vertex represent a sentence, calculated using a connectivity matrix in which each entry $M_(ij)$ is computed as cosine similarity between S_i and S_j representation [5], i.e to create clusters of sentences and to calculate the centrality of each sentence in order to extract the most important (central) ones to include into the summary. A cluster of document is a set of sentences that are similar (more or less) to each other: the sentences that are similar to many of the other sentences are considered as central and so salient (Centrality-based Sentence Salience): the first step is to build the cosine similarity matrix where M_{ij} is the CosSim between S_i and S_j (it can be seen also as a undirected weighted graph like in TextRank): then, given the similarity

matrix, sentence centrality must be computed: authors introduce three different ways to do this: Degree Centrality only consider as edge of the undirected graph the similarity score that are above a chosen threshold: then Degree centrality of a sentence is the Degree of the corresponding vertex, i.e. the number of vertices (sentences) the vertex (sentence) is linked to with a weight over the threshold (each edge is considered as one vote,not depending on his weight); this score does not take into account where the vote comes from, i.e. sentences from documents that are not related to the interest topic could get an high score if they are central into that document: LexRank score take into account this problem and use a centrality propagation function based on eigenvectors to propagate the centrality of a sentence to the neighbours, considering in this way where a vote come from: this score take into account the Degree score and the Centroid score for each sentence; LexRank does not consider the weight of the edge of the graph so authors introduce another score measure that is continuous LexRank which add the weight of the edges of the graph in the computation.

Zheng and Lapata[24] present an unsupervised approach to Single Document Summarization which relies on a graph-based algorithm and modifies how sentence centrality is computed in two ways, i.e. using BERT or using **tf-idf** to calculate how two sentences influence their respective centrality. The purpose of this system is to modify LexRank by using BERT to better capture sentence meaning and compute sentence similarity and then to use a directed graph, starting from the assumption that a sentence A could give more centrality to a sentence B than B could give to A: this verifies for example with two sentences that are linked because A contains an opinion about an event described in sentence B, that, different from A that would not have any sense without B, could be a central sentence also without A: according to authors, this is influenced by sentence position in the original text. Authors aim is to identify "nuclei" sentences and "satellites" sentences: in order to do this authors assume that sentences that come first in the document order are candidate to be "nuclei" so a sentence should receive an higher score from the sentences that follow it in the original text and that is similar to it and a lower score frome sentences that

preceed it in the text. A sentence score is given by the following formula: $W_1 \cdot Sum_{j<i}(e_{ij}) + W_2 \cdot Sum_{j>i}(e_{ij})$ where e_{ij} is the similarity between S_i and S_j and $W_1 < W_2$ (authors experimented that the optimal W_1 tends to be negative, so sentence that appears before seems to have negative impact referencing the sentence in analysis). In order to compute similarity between two sentences can be used BERT, which use a pretrained model to predict if a sentence is likely to refer to another sentences, or Cosine similarity applied to sentence representations obtained through tf-idf score for each non-stopword contained in the sentence.

Tixier et al. [4] propose an unsupervised system that performs extractive text summarization using a greedy approach to preserve near-optimal performance. The system is based on a "Graph-of-words" representation in which each node is a unique unigram contained in the original text, two nodes are linked by an edge if the unigrams represented by that nodes co-occur in the text in a window of W words: the edge is undirected and its weight represents the times the unigrams that it connects co-occur in the input text. The algorithm is based on k-core decomposition of the graph: a k-core of a graph G is a maxumal connected subgraph whose vertices has at least degree (i.e. the sum of the weights of the incident vertex) k. The k-core decomposition is obtained extracting all the core of the graph from k= 0 (or 1) to k= k_{max}. The proposed system is divided into four steps:

- **Text preprocessing**: the input of the system are utterances, i.e. fragments of speech transcrippts or traditional documents: utterances that are shorter than a threshold are removed and also incomprehensible sounds: stemming and stopwords removal are performed (copies of the original sentences are kept to be selected in the last step). The remaining words will be the nodes of the graph.
- **Graph building**: the graph is built as previously described.
- **Keyword extraction and scoring**: CoreRank is used to compute the score of each node of the graph, summing the core numbers (the highest order k of a core that the node belongs to) of the

neighbours of the node in analysis: at the end a fixed percentage of nodes are selected (the ones that have the highest scores).

- **Extractive summarization**:a submodular and monotone function is used to compute the concept coverage of a candidate summary S (set of sentences from the original input text: several sets are generated in a combinatorial way combining the input sentences): the score of a candidate summary is computed as the summation of the score of each keyword contained in S, multiplied by the number of times that keyword appears in S. To promote non redundancy a Diversity reward is computed for each candidate summary: it is computed as the number of contained unique keyword divided by the total number of keywords (the more keywords are contained the more a summary cover several concepts and so it will receive an higher score).

As we can see from the description of the previous algorithms, differences between them, and in particular between their performance, can be identified in the methods used to commpute sentence representation, the method used to build the graph and to compute sentence importance using relationship between vertices of the graph.

The authors in [25] start from a simple model based on word frequency and construct a sequence of models that inject more structure into the text representation. This method can bee seen as a combination of LSA based and deep learning based techniques. The aim is to extract a summary, long at most L words, composed by sentences extracted from the original documents set. Authors introduce several models for MDS: SumBasic algorithm select sentences iteratively by scoring them in terms of number of high-frequency non-stop words that they contain: in order to discourage redundancy the set of "already included words" is updated with words contained in the just selected sentence; **KLSum** select sentences greedyly in order to minimize the Kullback-Lieber divergence between the document set unigram distribution and the final summary unigram distribution; TopicSum uses LDA, useful to find the latent topics contained in the input corpus, to identify the significant content in the original set of document that the final

summary should contain: the aim is to apply KLSum between the learned content distribution (in place of unigram distribution) and the final summary distribution; in all these models the content of the document set is treated as a single unigram (or topic) distribution: HierSum, the last model that authors introduce, allow to model the several sub event: so instead of drawing a single general CONTENT distribution, also several SPECIFIC distribution are extracted for each sub-story. HierSum can extract general summary by plugging the GENERAL distribution into KLSum criterion or it can produce a summary for each topic by extracting a summary for each SPECIFIC distribution.

Carbonell et Al. approach the text summarization problem introdicing MMR, Maximal Marginal Relevance, a criterion that select sentences from the original text combining query-relevance and information-novelty, combination called by the authors "relevant novelty" [10] : the aim is to maximize the "marginal relevance" of the input documents, i.e. the linear combination between the similarity of the current document to the input query (positive influence in the final score of the document) and the maximum similarity of the document to a document of the previous selected ones(negative inflience in the final score): "a document has high marginal relevance if it is both relevant to the query and contains minimal similarity to previously selected documents" [10]. This criterion is applied to Single Document Summarization considering passages of the input document (sentences) as "document" in the MMR algorithm and calculating for each passage its relevance to the input query and its similarity to the previous selected sentences, trying to maximize the linear combination of these two factors.

Among the LSA based algorithms we can find [11] presents a term-based summarization system that uses LSA to identify the most important information of the text, represented by terms and their combination ("topics"): for this purpose a term to sentence matrix is built, in which each column is the representation of a sentence, given by the weighted frequency of each one of the contained sentences: then SVD is used to extract the singular vetors, representing each one a latent topic, whose importance is given by their magnitude: the sentences

that have the longest terms pattern expressed in the final matrix are selected to be part of the final summary: this algorithm is applied both in single-document and in multi-document summarization, with the difference that in multi-document summarization sentences that are considered into the matrix come from several documents and so important sentences could be similar to each other: cosine similarity is used to evaluate similarity of a candidate sentence to the already selected ones. The disadvantages of SVD-based summarization are highlighted by authors of [13]: according to authors there are two different disadvantages: the first is that the higher is the dimension of the reduced space, the less is the coverage of the selected topics (this could become an advantage if the total number of topics of the original documents are known and then the number of sentences to sentences to select is made equal to this number); the second disadvantage is that sentences that have an high single value score could be discarded by the process just because they have not the longest pattern of terms: authors propose an enhanced LSA summarization system to remove these disadvantages by multiplying each sentence to corresponding singular values to favour sentences that correspond to the highest singular values. Moreover authors in [12] use LSA in Update summarization, building the previous mentioned matrix for the set of old document and new document separately: SVD is applied to both the matrices to obtain topics of the old set of document and topics of the new set of document (as combination of terms, or terms patterns as in LSA based summarization): the novelty score of each new topic is computed as $1 - red(t)$, where $red(t)$ is the similarity between the new topic and the most similar old topic: the score of each sentence is the combination of the longest contained pattern lenght and the novelty score: each time a new sentence is selected novelty scores of the remaining sentences are updated.

Now we will introduce some cluster-based state-of-the-art methods:

Radev et Al. [7] introduce MEAD, a system that applies TDT (Topic Detection and Tracking) to generate clusters of sentences, given an input corpus of documents, and to extract centroids to be part of

the final summary. The proposed technique, called **CBS (centroid-based summarization)**, receive as input the centroids of the clusters, generated gathering documents regarding the same event. Each cluster produced by the TDT system is a set of chronologically ordered news articles describing the development of the same event, so that the centroids of the cluster are central sentences for all the articles of the cluster; to determinate the relevance of a sentence in the cluster CBSU (cluster-based sentence utility) is used to determinate the degree of relevance of the sentence to the general topic of the cluster; CSIS (Cross-sentence informational subsumption) is used to gather sentences in equivalence classes: two sentences that contain the same information, i.e. two sentences subsuming each other, belong to the same class. MEAD uses CBSU in the Centroid-based algorithm in which a centroid is considered as a pseudo-document, i.e. a set of words, extracted from the document of the cluster (and updated each time a document is inserted into the cluster), that have an IDF score above a given threshold, and CBSU score is computed based on the number of "centroid words" that the sentence contains: sentence are scored and then the top N sentences are returned as output in the document order. MEAD uses instead CSIS in the Redundancy-based algorithm in which CSIS between two sentences constitutes a "redundancy penalty" to subtract from the SCORE of the sentence (obtained by the centroid-based algorithm): this two algorithm are used as step of the general algorithm to extract the final summary avoiding redundancy.

Miller [6] proposes a RESTfull service that performs extractive text summarization receiving as input transcripts of lectures and using BERT for sentence embedding and K-Means clustering to identify sentences that will be part of the summary, i.e. sentences that are near to centroids of the generated clusters. Also in this case the algorithm follow different steps: the first is "Textual Tokenization" in which NLTK python library is used to split the input text in sentences, then sentences that begins with conjunction (that tipically are not significand if not preceded by other sentences) are removed and so sentences that are too small or too large. Then sentence embedding is performed and embeddings are computed for the input sentences using BERT, exploiting the default

pretrained model to generating an NxE matrix in which N is the number of sentence and E is the embedding dimension (so each row represent the embedding of a sentence S_i). The output matrix is passed to the next step in wich K-Means clustering is performed (in wich K is the desired dimension of the output summary in terms of sentences): K clusters are generated and from each cluster the closest to centroid sentence is selected to be part of the summary.

Authiors in [26] propose a system that take as starting point the Markov Random Walk Model, in which sentences are vertices of a graph and each sentence score is computed as the "votes" expressed by neighbours (as we have seen i graph-based described methods), and extend this method by clustering sentences by described topic and considering cluster-level information (in particular the importance of the topic described by the cluster): the proposed system is based on link analysis, usually used to compute relationship between web pages but considered in this case as link between two documents, which is the similarity between two documents. The system is based on two main concepts: a sentence in an important theme cluster should have an high score and an important sentence in an iportant theme cluster should have an higher score: the algorithm is so divided into subtasks:

- Theme cluster detection: in this step sentences are grouped into topic clusters, using a clusterization algorithm (the authors used K-means, agglomerative and divisive clustering)

- Sentence score computation: for this purpose two methods have been proposed by authors, both combining graph-based and cluster-based score for each sentence: the first method, Cluster-based Conditional Markov Random Walk Model, is a two-layer method in which the first layer is the sentence graph and the second layer contains the theme clusters: the weight of an edge is computed as the combination of similarity between two sentences, importance of each sentence in the belonging cluster, and similarity between the two cluster (considered as probability to go from a cluster to the other): the final score of each sentence is computed through Markov Random Walk. The second method is instead based on

relationship between each sentence and each cluster and the score of each sentence is given by the combination between the AuthScore of the sentence and the HubScore of the cluster it belongs to.

- Sentence selection: as in all the summarization algorithms the last step is sentence selection in which the most important sentences are selected avoiding redundancies.

Authors in [8] propose a system that perform text summarization by performing simultaneously document clustering, exploiting a document-term matrix and a sentence-term matrix, and text summarization with the aim of obtaining a better document clustering method with more meaningful interpretation of sentences, and a better summarization method that takes into consideration document context information. The system receives as input the document set: stop words removal is performed and then the document-term and sentence-term matrices are obtained,using the unigram language model, in the preprocessing step: the sentence-term matrix is used as basis for the proposed FGB (Factorization with given basis) model: the model uses nonnegative factorization on the document-term matrix, using the sentence-term matrix as basis, to obtain, upon convergence, the document-topic(for each document a cell is the probability of the document to belong to a topic) and sentence-topic matrices(for each sentence a cell is the probability of the sentence to belong to a topic): at this point clusterization and summarization of each cluster are performed simultaneously assigning each document to the topic to which the document belongs with the highest probability and then extracting the sentences the with highest probability for each topic to form the final summary.

A similar approach is proposed by authors in [9] in which they perform sentence clustering and ranking simultaneously, based on the spectral analysis: the aim is to explore the "clustering structure" of each sentence, "structure of beams", extracting the spectral features of sentence similarity networks: the network is constructed used a classical graph approach in which each vertex is the TFISF, term frequency inverse sentence frequency, representation of a sentence and an edge's weight is the cosine similarity between two sentences; this process is

aimed at revealing the natural relationship between sentences that is useful to clustering and ranking at the same time; each sentence is then projected on each beam(representing a cluster) and then only the beam with the highest projection length is kept as belonging cluster of the sentence in analysis: so sentence projected (highest projection length) to the same beam will belong to the same cluster ant at the same time sentence importance is computed as the length of the projection on the corresponding bean: so in this way clustering and ranking are obtained: sentence extraction is performed extracting the most ranked sentences from each clustering, iterating clusters starting from the one with the highest order of size.

As previously mentioned differencies between these algorithms can be identified in the used clusterization method and in the way sentences are compared, scored and so extracted: it is different instead the approach proposed in [9] and [8] in which sentence extraction is performed simultaneously with clustering.

All the previous algorithms are the classic algorithm used in automatic text summarization: now we will introduce some interesting and more recent system that perform the text summarization task.

The first is called SummaRuNNer [27], which is a system that perform extractive text summarization applying a Recurrent Neural Network (RNN) to a single document. It can be classified as a deep learning based algorithm: the algorithm is a classification algorithm based on a bidirectional GRU-RNN which has two hidden layers: the first operates at word level within the input sentences, computing the hidden state word representation based on the current word embedding and the previous hidden state, doing this from the first to the last word and then from the last to the first word (bidirectionally). The second hidden layer receive as input the concatenated hidden states of the previous level and operates bidirectionally at sentence level, encoding the representations of the sentences of the input document, obtaining in this way the encoded representation of the input document. In the top layer the classification algorithm "predict" whether to select a sentence or not according to its content richness, salience with respect to the document, novelty with respect to the already selected sentences

and positional features; in order to apply this classification algorithm a model must be trained in order to classify sentences to select and sentences to discard: two training method has been proposed by authors: the first, the "Extractive Training", is based on the idea that selected sentences should maximize the Rouge score, so sentences from the training set are selected greedly as long as they could maximize the Rouge score and then return this set of sentences as ground-truth to train the model; the second method, the "Abstractive Training",uses a decoder to model the generation of abstractive summaries at training time only.

The last category of algorithm to be described is the itemset-based: we present for this purpose the description of the following algorithms.

Authors in [14] propose a system named ItemSum that performs text summarization based on frequent itemset mining selecting sentences with an high level of coverage with respect to an itemset-based model and with an high sentence relevance score based on tf-idf score. The algorithm performs stopwords ans ULR removal and then stemming to obtain a set of sentences represented through the classic BOW (Bag of words) representation in which each sentence j belonging to document k is the set of stems occurring in that sentence; sentence are then seen as transaction of a transactional dataset (the union of all the sentences of all the documents in the original dataset) composed by items, i.e. unique stems contained in the sentence, whose importance is at the same time computed as the tf-idf score of the stem against the whole collection. ItemSum generates an itemset-based model of size ms that contains the ms most representative and non-redundant frequent itemset, where a frequent itemset is a set of items (stems), co-occurring in at least one of the input sentences, whose support (given by the number of transactions,i.e. sentences, in which the itemset is contained divided by the total number of transaction) is above a given threshold; sentences are selected combining the sentence relevance score, computed as the summation of the tf-idf score of each distinct term that is contained into the sentence, and sentence model coverage score, whose computation is a set covering problem: for each transaction (sentence) a coverage binary vector is built: the vector has dimension

ms and each cell represent an itemset of the model and it has value 1 if the corresponding itemset is supported by the current transaction (i.e. it is contained into the sentence), 0 otherwise. ItemSum applies a greedy strategy, that prefer sentences with the highest coverage level (high number of 1's into the coverage vector) and the highest relevance score, to solve the set covering problem to obtain the minimal set of high-scored sentences whose logic OR of the coverage vectors of each sentence gives as output a binary vector with the maximum number of 1's.

Authors of [15] presents a multilingual summarizer, called MWI-Sum, based on "frequent weighted Itemsets": different from the previous described ItemSum [14], MWI-Sum consider weighted itemsets and is applicable to any language for which a stemmer and a stop words list can be found. A frequent weighted itemset is a set of terms, having an high relevance score in the analyzed collection, that co-occur many times into the original collection: the algorithm is divided into several steps:

- Document preprocessing: stop words removal (using NLTK) and, optionally, stemming (to reduce terms to their root form) are performed and a relevance score is assigned to each term: for this purpose tf-df (term frequency-document frequency) is used: it is a variant of tf-idf, proposed by authors, that assign a score to a sentence term of a specific document directly proportional both to frequency into the document and frequency into the collection. At the end of this step a set of transaction (sentences) is obtained and each sentence will be a set of weighted item.

- Sentence Filtering: in this step authors assume that sentence that are more relevant will appear at the beginning of each document and that other sentences will contain repeated information: only the top-K (in terms of order into document, K user-driven) sentences reach the next step;

- Itemset-Based Model Generation: in this step an itemset model is generated: the model will be composed by frequent itemset

that contain relevant terms; starting from the transactional representation of the dataset (each sentence is a transaction of scored terms/items) the weighted support value of each transaction is computed as the summation of the matching values of all the itemset contained in the transaction (i.e. the lowest tf-df value of each contained itemset) divided by the summation of the maximum item weights of each transaction: in this way support increases with the number of "high scored itemsets" contained in the transaction; the transactions whose weighted support is above or equal to a given threshold are kept as part of the model.

- Frequent Weighted Itemset Mining: a variant of the FP-Tree index is proposed in which the weighted support of each transaction is stored instead of the classic support: then each sentence is associated to other sentences having the same support weight and an FP-growth-like itemset mining is performed to obtain the frequent weighted itemsets.

- Sentence Evaluation and Selection: sentences are selected considering their relevance score, computed as the sum of the tf-df score of the contained terms, and coverage of the generated model, computed as the total number of 1's contained in the coverage vector, computed as specified in [14]'s description. Also in this case the sentence selection problem is approached as a set covering problem aimed at selecting the minimal set of sentences with maximal coverage score, i.e. maximal number of 1's in the summary coverage vector, i.e. vector obtained from the logic OR between all the coverage vector of all the sentences in the "summary": this problem is solved using a greedy approach that prefers sentences with an high number of 1's in the coverage vector and an high relevance score: at the beginning the summary coverage vector is initialized as vector of 0's : at each step the sentence with the highest number of one's in the coverage vector and with the highest tf-df score is selected and added to the summary, updating the summary coverage vector and the coverage vector of the remaining sentences, setting to 0 the bits of the already covered itemsets:

in this way one's in remaining coverage vectors will represent the covered itemset that has not been covered in the summary yet and so at each step, selecting the sentence with the highest number of one, the best complementary vector with respect to the summary coverage vector is selected: the algorithm stops when the summary coverage vector becomes a vector of one'1, i.e. all the itemset of the model have been covered by the summary.

Authors of "ELSA: A Multilingual Document Summarization Algorithm Based on Frequent Itemsets and Latent Semantic Analysis"[16] propose ELSA, a system for multilingual document summarization that exploits Latent Semantic Analisys (LSA), which derives concept by modeling them as a combination of single-document (limitation of this approach) terms and applying Singular Value Decomposition to a term-by-sentence matrix that stores per-sentence terms frequency, and then exploits frequent itemset (combination of terms that co-occurs together) mining to extract sentences that contain the largest number of frequent itemsets. In ELSA frequent itemset is used to describe all of the latent concepts contained in the input documents and LSA to avoid redundancy in the itemsets: the summarizer aims to select sentences,trying to maximize the latent concepts coverage while minimizing redundancy [16]. The aim of ELSA is to exploit itemset to consider the relationship among multiple terms and LSA to summarize textual content to obtain meaningful concepts, avoiding the limitations of the two technique. ELSA receive as input a collection of multilingual textual document and extract a summary following four main step:

- **Document Preparation**: in this step the textual content is prepared for the itemset mining by removing stopwords, i.e. words such as articles, preposition, conjuctions that are the most frequent but that don't represent significant information,stemming words, bringing them to their original form in order to not differentiate for example verbs based on their tens or the same term in singular or plural form, and by trasforming documents into a transactional form, i.e. all documents are gathered in a single dataset and each sentence is represented as a transaction formed by a set of "stems".

- **Frequent Itemset Mining**: the aim of this step is to identify the itemsets that potentially describe significant concepts, i.e. the most frequent itemsets: "a k-itemset is a set of k stems that co-occur in any transaction in the transactional dataset" [16], and frequent itemsets are the itemsets whose support, i.e. the number of transaction (sentences in the transactional dataset) in which the itemset is contained divided by the total number of transactions, exceed a given threshold minsup: in order to identify the frequent itemset an FP-Growth algorithm is used: output itemset are presented in a Itemset-by-Sentence matrix in which M_{IS} is equal to 1 if I is contained in S (S covers I) or 0 on the contrary.

- **Singular Value Decomposition**: the aim is reducing IS matrix to a CS (Concept-by-Sentence) matrix using SVD to obtain for each frequent itemset a subset of latent concepts and for each sentence the subset of correlated concepts, associating to each concept a degree of importance.

- **Sentence Selection**: the summary will include the sentences that cover the most important and relevant concepts identified through SVD (sentence significance) and that are not similar (in the space of concepts identified by LSA) to already selected sentences (sentence redundancy); the significance of a sentence is calculated as the combination of relevance of the corresponding concepts and relationship degree between the sentence and each of the related concepts; two sentences will have an high score of redundancy if their coverage of all LSA concepts is similar: according to this assumption, $Sim(S_x, S_y)$ will be equal to 1 if the similarity score is over a pre-defined threshold, it will be 0 otherwise; sentence selection step can be summed-up as follow: for each sentence the significance is computed, the sentence with maximal significance is included in the summary and sentences that are similar (according to Sentence redundancy measure) to the already included sentences are discarded from the set of sentences; these two passages are repeated until maximal summary size is reached or no more sentences are available.

The analyzed algorithms perform text summarization: as previously specified the framework proposed for this thesis performs, instead, timeline summmarization: nevertheless, the main contibution of this thesis is to introduce text summarization into timeline summarization task, to extract the most significant sentences for each one of the selected dates and to explore performances of state-of-the-art summarization algorithms in terms of quality of the generated timeline: for this purpose we employed some of the previously described algorithms.

3.2 Temporal Summarization

Presented as a challenge in TREC2015, the aim of this task is to emit a series of sentence updates, relevant to the input topic and with an high degree of novelty with respect to the already emitted sentences, over time about a named event, given a high volume stream of input documents. In particular, the temporal summarization task focuses on large events with a high impact, such as protests, accidents or natural disasters. Each event is represented by a topic description, a textual query that represent that event is provided, along with start and end timestamps defining a period of time within which to track that event [23]; this task could contain a sub-task that is filtering (given a set of document, only the ones related to the topic in analysis are selected). A lot of paper and algorithms have been published, approaching in different ways the problem of TS: Wan [28] propose an algorithm based on the classical TextRank: it is a graph-based algorithm that "makes use of the relationships between sentences and selects sentences according to the “votes” or “recommendations” from their neighboring sentences" [28]: the documents are split in a set of sentences, then each sentence is represented as a vector using tf-idf to calculate the weight of each term that is contained in the sentence itself: sentences are considered as the vertices of a weighted undirected graph in which two vertices/sentences S_i and S_j are linked by an edge (weighted w_{ij}) only if $w_{ij} = CosSim(S_i, S_j) > 0$: this weight represent the "vote" that S_i "express" for S_j and the score of S_i is the summation of all the "votes" expressed for it by all the other sentences; TimedTextRank

is based on the concept that sentences in more recent document are more important, and so their "expressed vote" is more important, than the earlier ones: so a weight is introduced into the summation in order to take account of the temporal information and the more the sentence that express a vote is recent (it belongs to a more recent document) the higher is the expressed vote; a postprocessing step is performed in order to remove redundancies.

Authors of "Online Temporal Summarization of News Events" [29] propose a system that applies temporal summarization to a stream of news document, with the aim to detect real-time if a document is relevant to a topic query and to extract updates as soon as possible: authors approach this task dividing it into three steps: in the first step each document that arrives in the stream is filtered (only documents that are relevant to the selected topic are kept) applying BM25 to the document and the input query (query espansion is applied to the query); then in the "Relevant Update Retrival" step, sentences that are relevant to the query are identified: in order to do this, a dynamic language model LM_Q is calculated for the query (aggregating the terms of the top-100 BM25 scored article extracted from wikipedia for the topic query in analysis) which is updated with the terms of the already selected sentences, and a language model LM_S is calculated for each Sentence: then KLD-score between LM_Q and LM_S of each sentence extracted from a relevant document is calculated in order to detect the topic-relevant sentences; in the last step only the novel sentences are chosen as updates: a sentence is selected if it is a "Novel Update by Number Occurrence", i.e. it contains at least one "Number-ContextTerm" pair (e.g. 40 death, 100 Km/h, 30 wounded) that appears in the current document enaugh times- greater than a chosen threashold T_{num} - to be considered as an update, or if it is a "Novel Update by Term Frequency", i.e. it contains at least one term that is novel and significant ($IDF_{now}(term) < IDF_{withoutCurrDoc}(term)$, it appears at least t_{term} times into the document, it has never been selected as novel and significant before).

BJUT [30] at TREC2014 proposed a multi-level system that receives as input a set of temporally-ordered documents and a topic query and

performs temporal summarization through two modules: the "Information Retrival Module" exploits Lemur and BM25 to extract sentences that are relevant to the expansed (in order to take account that in the text synonims of the terms of the original query could be used) input query; these sentences are processed by the "Information Process Module" that in an initial step select only the sentences that are in the temporal range of the event, then k-means clustering is applied and from the central cluster the top-100 scored sentences (score based on time and similarity) are extracted to compose the final summary and obtain temporal summarization.

The representatives of university of Glasgow presented at TREC2014 a system [31] that performs, among several tasks, temporal summarization with three principal aim: issue updates as soon as a new document arrives (unlike the rank-than-select systems that issue updates each hour), increase the coverage of the nuggets (sub-events of the original event), avoid to issue sentences that have semantic intersection with the query (obtained through "sentence proximity"); the algorithm is divided in three step: in the first step a machine learned document classifier is used in order to compare the document to the expansed topic query and decide if the document is on-topic: in this case the document reach the next step in which sentences are extracted from the document and a series of classification heuristics and a supervised sentence classification model (trained on a set of sentence extracted from TREC TS topics, uses emergency related terms an quality features, such as the presence of capital letters) in order to selected the relevant sentences, i.e. sentences that are neither too long neither too short, sentences that contain named entities, sentences that are "well written" according to the previously trained classification model: these candidate sentences are compared, in the last step, with a greedy cosine similarity heuristic, to the already extracted ones and only the most different are issued as updates.

Tabitz et Al. [32] presents a system that has the aim of covering several (the more the better) aspects (nuggets) of the topic of interest: the algorithm receives as input a set of document E, a topic query Q and a temporal range that goes from the start date to the end date of

the event and that is divided in intervals $H_{ts} -> H_{te}$; the algorithm iterates the time interval and perform summarization as follow: given the current interval H, it iterates all the documents and select only the documents that are on-topic (BM25) and in the range $H(t_s, t_e)$; at this point relevant sentences are selected as candidates updates for interval H according to the following criteria: lenght of the sentence ($4 < l < 40$),similarity with the query, presence in the sentence of terms that are contained in the language model generated for the event of interest (that contains terms that are typical of the kind of event in analysis): this step is called "double filtering"; then, given the candidate sentences, LDA is applied to extract from the sentences the set of topics $T(H)$ and to compute for each sentence the probability to belong to each topic $P(s, T_i)$ (soft clustering); then topics are sorted in order of importance (number of sentences contained, i.e. sentences that have $P(s, T_i) > 0$) and sentences are sorted in order of importance (probability to belong to the topic) into the topic; in the last step the N_t most important topic are selected and for each topic the N_s most important sentences are extracted to form the summary for that interval H.

In [33] the authors presents a system whose aim is to select, at a certain time instant T sentences that are considered "Vital Sentences" and that are novel with respect to the already extracted ones: given a topic query Q, a summary composed by the sentences extracted in the previous time instants $T_0...T_{n-1}$, a new time instant T_n and a set of new documents arrived between T_{n-1} and T_n the proposed algorithm extracts the documents that contain "vital sentences" and extract that vital sentence from the previously selected documents: a sentence is considered a "Vital" one if it has a $CosSim(S, Q)$ higher than a threshold or/and it contains at least one "trigger word" (e.g. for a natural accident these words could be "deahts","wounded"...); at this point not novel sentences are filtered out and only novel sentences are kept: novelty score is computed as the combination of Text Divergence score (computed using Cosin Similarity) and presence in the sentence of event-related entities that can be considered as updates.

In [34], Kedzie and the other authors present a system that combine

"Salience prediction" and clustering in order to extract, given a stream of temporally-ordered documents, a topic query, a time interval of interest, sentences as updates: in the first part of the algorithm "Salience prediction" is applied: the salience score of a sentence is calculated using a Gaussian Process Regression Model, influenced by four kind of "Features": first, the "Query features" i.e. the similarity with the query based on the number of terms that the sentence and the query have in common or the number of terms in the sentence that are synonyms of the ones in the query; second, the "Language Model Features", i.e the likelihood that a sentence has been generated from a Language Model that contains terms specific for the topic of the input query (the topic of interest of the analysis); then "Geo-location" (the sentence refers to a geographic location that is near to the event location) and "Time" features are considered; "Salience Prediction" is combined with the Affinity Propagation clustering algorithm and the most salient sentences are extracted: at this point the extracted sentences are compared with the sentences extracted from the previous documents and only the less similar are selected and "emitted" as updates.

In [35] the authors describe their system which try to issue updates from a stream of news articles periodically (e.g. each hour, at the beginning of each day...) and before the next group of news article arrives: it receive as input a stream of news and a topic of interest, that is considered as a set of topic-related events: each sentence is analyzed and considered on-event (related to one of the event that compose the topic) or off-event; then the system aim to recognize the "Useful" sentences by scoring them using IR and computing the probability of the sentence to be generated from a Language Model containing typical words for the events of the topic of interest(expressed as the percentage of word of the sentences that belong to the LM); in order to compute the "Novelty" of a sentence, all the sentences (even the off-topic ones) are processed: if a sentence generate a new cluster it will have an high score of novelty, if it belongs to an already existing cluster (with a certain probability) his score will be lower: doing that, the system will assign an high score to the off-topic sentences that will probably generate a new cluster: at the end this score is neutralized becouse only

sentences that have an high total score ($score(s) = U(s) \cdot N(s)$) will be selected.

In [36] Zhang et Al. approach the TS problem as a sequential update summarization problem: the algorithm receives as input a stream of document chronologically ordered (from T_s to T_e of the event) and issue useful, novel and time-related updates using three modules and an input set of keyword K for the topic: the "Preprocessing and IR Module" exploit Indri to perform IR between documents and the expansed query and select only the important (on-topic) documents; the "Keyword Mining Module" uses LDA to find the latent topics in the document and to find the representative words for these topic and update K with these keywords; the third module, the "Sentence Scoring Module", uses three different approaches to score sentences: (1) KLP: an update is a long sentence which contains keywords for the topic and that is at the begining of the paragraph; (2) SKD: an update is a short sentence with an high degree of keyword diversity; (3) KS: an update is a sentence with an high degree of keyword diversity; in the post-processing step sentences with the best score (according to one method between KLP, SKD, KS), that are in the temporal range (T_s, T_e) of the event and that are not redundant with respect to the already issued updates are selected to be issued as updates.

3.3 Timeline Summarization

In the previous section we spoke about Temporal summarization, now we will put the focus of the discussion on **Timeline Summarization**: there is an abundance of reports on events, crises and disasters, timelines summarize and date these reports in an ordered overview; the aim of TLS task is to create a timeline, an overview of a long-running event, to follow the evolution of that event across time via dated daily summaries (generated from a corpus of dated documents) for the most important dates; so the TLS task can be divided into two sub-task: the first is the "date selection" sub-task, in which text summarization can be used to select the dates that contain the most relevant sentences: the importance that this sub-task assumes in TLS costitutes the main difference from

MDS (a date can be selected as important one even if the summary for it is shorter than other date's summaries); in the second sub-task, "date summarization", for each date, the most important sentences are extracted through text summarizzation. As we will see this two step could be united in a single step or performed sequentially. This thesis will focus on the "date summarization" step, but before talking about it several TLS methods, considered as the related work for this thesis, will be presented in this chapter.

In "Examining the State-of-the-Art in News Timeline Summarization"[37] the authors examinate several TLS state-of-the-art methods and classify the several approaches used in TLS in three classes:

- **Direct Summarization Approach**, in which the set of time-ordered news articles is treated as a single set of dated sentences and most important sentences are extracted and their dates will form the timeline.

- **Date-wise Approach**, in which l dates are selected and then for each date k sentences are selected to form the summary for that date; in the first step a set of candidate dates are extracted from the articles selecting the publication dates of the several article and resolving date references (e.g. 'last Friday', 'October 21th'...) that are found in the text of the articles; then the l most important dates must be selected: in order to do this several method have been used, we will analyze them in the framework description chapter.For the date summarization step authors propose an heuristic that select as candidate sentences for each date D_i all the sentences of an article published in that date and all the sentences that refer to that date and try to find the sentence that are likely to mention important events for D_i (sentences are represented as vector,applying TF-IDF, and a date vector is generated, then CosSim between each sentence vector and date vector is computed); then each date is "summarized" separately using a MDS algorithm (TextRank, Centroid, Submodular are analyzed by authors in this paper).

- **Event Detection Approach**, in which a timeline is considered as a sequence of event, so clustering is used to gather documents in

"event", adding temporal constraint (each cluster is built around a representative date) and then select the l most important events/-clusters (by Size of the cluster, Date Mention Count i.e. how often the date of the cluster is mentioned in the input articles, Regression to predict the importance of the cluster); then each cluster among the l selected ones is summarized separately (using a MDS algorithm).

Chieu and Lee in their "Query Based Event Extraction along a Timeline" [17], use a Direct Summarization approach, performing TLS as extraction of events relevant to a query extracting one representative sentence for each event, starting from the assumption that an important event is widely cited into the input corpus of document. In order to perform this task authors propose an algorithm that can be divided in five steps: in the first step the algorithm try to get the set of sentences from the corpus C that are relevant to the input query q: authors define the "interesting events" as events that are reported in many sentences and so "interesting sentences" the sentences that talk about "interesting events": the algorithm consider a sentence "important" if it is an interesting one and if it is related to the query, more precisely if it reports an event that is query related; in the second step a date is assigned to each sentence with a date resolution process (that for example assign to "Yesterday" the day before the publication date of the article which the sentence belongs to, assign to "Sunday" the date of the Sunday that preceed the publication date and so on): the authors assume that "the first time expression detected in a sentence is the date of the event mentioned in s" [17] and if no temporal reference is detected into a sentence the date of that sentence will be the publication date of the article to which it belongs (reasonable assumption for news article). In the third step the sentence are ranked according to their importance expressed in terms of "Interest" which is considered as the number of "interesting sentences" that reports the same event of the sentence in analysis: to do these sentences are represented as terms vectors and each term is scored with iDf which is a variation of traditional idf based on date instead of document: CosSim is used to determinate if two sentences speak about the same event ("could

be paragraphed"); then the most "Interesting" sentences are selected (duplicated are removed in the fourth step) and in the last step sentences are ordered in chronological order to form the final timeline, according to the dates previously assigned to the sentences.

Tran et Al. [21] see the TLS task as the task of "extracting important points of the story, both in temporal and content dimension" and propose a supervised approach that takes as input a set of article A_q related to a specific topic q, applies machine lerning to predict importance of each date and of each sentence to return as output the m most important sentences for each of the n most important dates previously selected. The importance of a date d is calculated computing the number of articles published on d (PUBCOUNT) or befor/after d but containing reference to d (MENTIONCOUNT) and the number of sentences that refer to d (MENTIONCOUNT) or published on d (PUBCOUNT). Sentences importance (for each of the selected dates) is calculated according to several features: Surface (length, position, presence of stop-words/non stop-words), coherence (temporal and logical reference), topic and time-related features: the score of each sentence is calculated by measuring their semantic similarity to the manually created summary.

Steen and Markert [22] propose an unsupervised abstractive system to perform the TLS task: the main objective of this system is to generate a timeline by selecting a fixed number of date and generating a summary (defining a maximum number of sentence) for each date, combining the sentences of the original text to generate new sentences (abstractive) with the same semantic content, differently from the systems analyzed so far that extract sentences directly from the text. The algorithm perform the task in three steps: in the first step (Clustering) sentences that describe the same event are gathered in a cluster: AP (Affinity Propagation) clustering is used for this purpose: it automatically and dynamically choose the number of clusters to generate for the input dataset, select an exemplar sentence for each cluster (it will be cluster's center) and then for each non-exemplar sentence select one of the exemplars to form a cluster with. Sentences of the same cluster must refer to the same date: each sentence can refer to the document creation

date, if it has some time reference it can refer to more dates and if it refers to a range of date it may refer to any date of the range. A sentence S_2 may select S_1 as exemplar to cluster with if S_2 refers to an exact date d_2 that contains d_1 referenced in S_1 (i.e. d_1 and d_2 refer to the same day or d_2 refers to a range that contains d_1 that must be an exact date); at the end of this step, for each cluster, the date that is the most referenced by the sentences of the cluster is selected as representative date for the cluster. In the second step Multi-Sentence-Compression (MSC) is used to generate new sentences for each cluster; in the last step a score is assigned to each sentence of each cluster using three scoring function: a linguistic quality score is computed to encourage a readable output (assigning an higher score to the sentences generated from a shorter path in the MSC algorithm); then a date importance score is computed, i.e. the nummber of times the date that the sentence in analysis refers to is mentioned in the input; the last score is informativeness, calculated using TextRank; the score of a sentence is the product of theese three score. In the second part of this last step sentence selection is performed: sentence are greedily selected starting from the one with the highest score, selecting at most one sentence from each cluster (so a cluster may not be included in the final timeline), skipping sentences with an high CosSim with the previously selected one and sorting them, when the maximum number of sentences has been reached, in chronological order along the final timeline.

Authors in [18] propose a TLS system that focus on generating a good daily summary starting from news article's headlines: the main reason of this choice is that extracting sentences from the whole text of the huge amount of articles related to the topic of interest, according to the authors, does not guarantee good results in terms of "understandability" (due to the fact that often there is not continuity between the selected sentences) and also "relevance" (because it is hard to extract the right sentences, i.e. the sentences that are actually the most important, from the huge amount of sentences that are received as input by a TLS system); moreover headlines are good candidates for timeline generation because they are complete (in terms of time

and contained information about the event that the article talks about) and "comprehensible to the reader without requiring too much reading time" [18]: the aim is to identify informing headlines, i.e. headlines that tells what happens and not background opinions about the event, then remove duplicate headlines and at the end extract the most relevant ones to build informative daily summaries; starting from a set of headlines H_d, extracted from articles published on d, the system compute for each headline the Informing value, the Spread and the Influence. An headline is informing if it describes what happened, so it contains event related information: an SVM(support vector machines) machine learning algorithm (trained on a set of relevant headlines and a set of non-relevant headlines) is used to determinate if an headline is relevant (Score=1) or not (Score=0); the influence of an headline $I(h)$ compute "how much" an event described by the headline is influential in the event that will happen in the future: it is determined by calculating the similarity between the word representation of the headline published in d and the word representation of the set of sentences of articles published in $d_1 > d$ that refer to date d; the Spread score represent the relevance score of an headline based on the number of headlines that are likely to report the same event of the headline in analysis (considered as duplicated): a weighted graph is built in which vertices are headlines and they are linked by an edge if they are likely to be duplicated (the weight is the similarity score); then a random walk approach based on a Logistic Regression model is used to compute the Spread of the headlines. At the end the headlines that have high Informing, Influence and Spread score are selected to form the daily summary for a date d.

Jiwei Li and Sujian Li [20] see Timeline Summarization as showing to the reader the evolution of topics in an event of interest, by extracting the most important topics from articles published in different "epochs": authors propose a evolutionary Hierarchical Dirichlet process (EHDP) for TLS: an HDP is built at each epoch (the time aspect makes it evolutionary) and topic popularity and topic-word distribution are inferred from a Chinese Restaurant Process: sentences will be selected in the timeline considering relevance,coverage and coherence. For each

epoch is selected a Corpus of query related documents and the CRP is applied: each document is a "restaurant", each topic is a "dish" associated to a "table" and each sentence is a "customer" that sit in the "corresponding table": so for each epoch sentences are gathered according to the topic they talk about: in order to select sentences KL is used to compute score for each topic that will influence three sentence scoring criteria: Relevance, i.e. the summary should be related to the input query, Coverage, i.e. for each epoch all the important topic happened in that epoch should be included in the summary, Coherence, a good summary should be coherent do the neighbour summaries in the timeline: in the last step of the algorithm of sentence selection MMR is used to avoid aspect redundancy.

Martschat and Markert [19] in their system highlight the importance of date selection in timeline summarization and how it is the main difference between TLS and standard MDS: however the aim of this paper is to show how MDS techniques can be extended and adapted to TLS adding time constraints and designing objective functions through submodular functions and if the result is a scalable, well-performing TLS model that keeps the advantages of MDS. The aim of the proposed system is to generate a timeline $(d_1, s_1)...(d_n, s_n)$ starting from a set of dated (by a date expression or the publication date) sentences extracted from the corpus associated to the input query. Authors first define summarization as an optimization (in a greedy algorithm that tries to build a good summary) of an objective function that must be, in order to guarantee performance and good results for the generated summary, monotone and submodular and that, tipically in MDS, tries to maximize Coverage and Diversity in the summary; moreover constraints, such as the number of sentences, are introduced to define the summary's structure and so to increase the performance (in terms of quality of the summary) of the greedy algorithm. This model is applied to TLS by temporalizing coverage function, looking at the temporally local neighborhood of the date d in analysis, by temporalizing diversity functions, i.e. passing from MDS semantic criteria for sentence partitioning to TLS temporal criteria to obtain date-based partitions (a partition will contain sentences related to

a date that probably will be related to the same sub-event of the principal event), and then adding date selection function, integrating date importance, calculated using the number of reference to a date in the original corpus, into the objective function: the final objective function is the sum of the coverage, diversity and date importance criteria. Moreover constraints are introduced in terms of cardinality of dates to select and cardinality of sentences to select for each date to obtain the final timeline.

As we can see from related works description, the most part of the state-of-the-art methods performs timeline summarization trying to extract important sentences and then sorting them in temporal order based on the associated date (Direct timeline summarization) or trying to identify the most important events by clustering sentences based on their date and/or content (event-detection based timeline summarization): the contribution of our work is a system that instead performs the timeline summarization task using the date-wise approach: as we will see more in detail in the next chapter, our system divides the main task in two sub-task, date selection, in which the most important dates are selected according to counting and mentioning critaria, and date summarization, in which for each date the most representative sentences are selected as daily summary,unlike [17],[18],[19] that use the direct summarization approach or from [22] and [21] that use an event detection approach: unlike [20] and [19], that use a MMR-based heuristic to select sentences and avoid redundancies, in our method we applied (and compared) several summarizer based on several summarization algorithms and techniques; moreover,differently from [21] we use the unsupervised approach in text summarization and the extractive summarization approach, unlike [22] that performs abstractive summarization.

Chapter 4

Timeline Summarization Framework

In this chapter we will present the implemented framework, focusing on the "Date Summarization" step: the picture below show an high level representation of the proposed system: as we can see the proposed pipeline consists of three blocks: the first block aims to select the most important dates from the input sentences obtaining as output a list of candidate sentences for each important date; the second block performs, for each important date, sentence selection, applying a text summmarization algorithm: the output file will contain the timeline, as we will see, in a specific format: the user will upload the timeline file into the final block, TimelineVisualization, that will show to the user the final timeline in a user-oriented way.

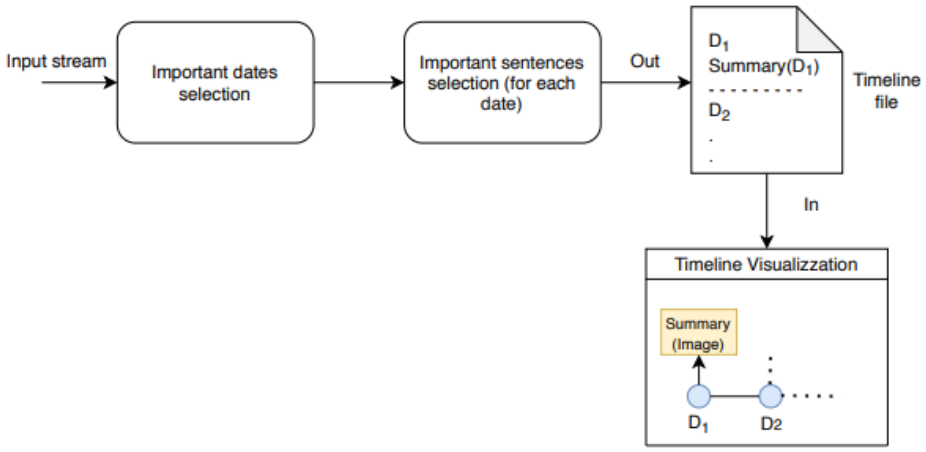

Figure 4.1: Timeline summarizaton pipeline of the proposed system

In this chapter we will discuss about the first two blocks, providing a brief description of the first and describing more in detail the second block, while in the next chapter we will present the web application, TimelineVisualization, that constitutes the third block of the pipeline. To get more in detail, our framework performs the TLS task using the previously mentioned **Date-wise Approach** in which a predefined number of dates are selected as important dates, that will be the "timepoints" of the final timeline, and then for each "important" date a fixed number of sentences are selected to form the final summary for that date. In particular our framework aims to generate a textual file that will be read, as we will see, from a minimal web application that will show to the user the final timeline. The framework receives as input a stream of news about a specific topic (i.e. a set of document associated with their publication date), the number of dates to select, the minimum length (in terms of number of words) that a sentence should have to be considered as a candidate sentence for the final date summary, the maximum number of sentences to select for each date summary; in the first step date selection is performed: news are annotated with a date using the publication date of the article in which

they are reported and using heideltime, which is a multilingual system that performs temporal tagging by finding temporal reference in the text; at this point sentences are assigned to a date, obtaining for each date a set of associated sentences: several methods can be found in SOTA for date selection, most of them are based on counting how often a date is mentioned in sentences: authors of [37] present three main methods, proposed by Tran et al.[21]: in PUBCOUNT the score of a date is computed as the number of articles published on that date,in MENTIONCOUNT the score of a date is the number of sentences that mention that date, in SUPERVISED date features (based on publication count and different variants of date mentions) are used together with classification or regression to predict if a date is likely to appear in a ground-truth timeline; in our framework methods that are not object of this thesis are applied to detect the most L important dates that will reach the following step; the second step is the "Date Summarization" step that performs,for each date, text summarization to extract N sentences for each important date as in Figure 4.2: the illustrated diagram presents the Date summarization process which is repeated **for each important date**: the date summarization block receive as input a set of date-related sentences, the minimum length TH of the candidate sentences, the maximum number of sentences to select: in the first step sentence preprocessing is performed to remove duplicates (in this case with "duplicate sentence" we refer to sentences that are identical to each other and not similar in terms of content, we bring just one of them to the next step) and to discard sentences whose length is below the given, user driven, input threshold TH; the next step is the date summarization step: this step is performed in our framework by a Summarizer: we introduce in this step several Summarizers that have the same structure but exploit several python libraries that implement several summarization algorithms:

- TextRank [3]: graph-based algorithm that assign score to sentences according to their neighbours' votes: sentences are vertices of the graph while edges represent the weighted vote that two sentences cast to each other: the weight of the edge is the similarity score computed between the sentences linked by that specific edge; the

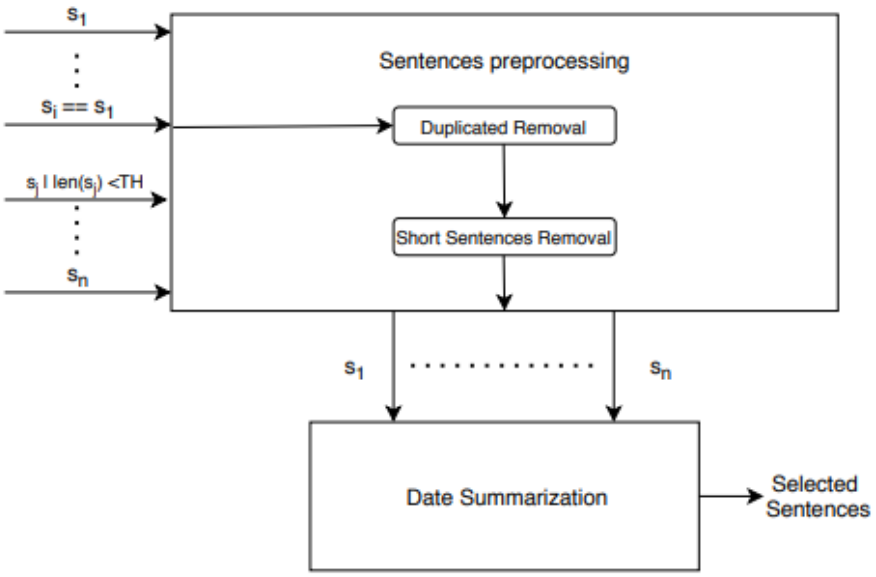

Figure 4.2: For each important date,the framework follow the illustrated steps to select, among the input sentences associated to the current date, the sentences that will form the summary for that date,using for each date the same summarization algorithm.

score of a sentence is the summation of all the votes cast for it by its "neighbour" sentences; in our framework we use implementations of Text Rank that use tf-idf to compute sentences representation and cosine similarity to compute similarity score between two sentences. We exploit two different implementations contained in Sumy [38] and Sumpy[39], which are python libraries which contains several baseline text summarization methods implementation. We also employed the implementation of a variant of TextRank described in [40] that exploit BM25, which is a variation of TF-IDF usually used for Information Retrival, to score sentences, instead of the cosine similarity.

- LexRank [5]: another graph-based algorithm that computes sentence importance as sentence centrality in the graph, computed

using a similarity matrix between sentences, using a centrality propagation function to take into account into LexRank score where a vote cast to a sentence comes from (in order to avoid selecting sentence that are central locally to the belonging document which instead is not relevant to the input topic). Also for LexRank we propose two different Summarizers that exploit Sumy and Sumpy implementation respectively.

- KLSum [25]: add sentences greedyly to the summary, as long as it decrease the Kullback-Lieber divergence, in order to minimize the KL-divergence between the document set unigram distribution and the final summary unigram distribution; even in this case we exploit Sumy implementation of KLSum.

- tf-idf based summarizer [24]: based on the concept that two sentences don't influence each other in the same way: "nuclei", self-contained sentences that are meaningful, and "satellites", sentences that are meaningful only because they refer to another sentence, influence another sentence's score differently: "nuclei" (usually appearing before "satellites" in the text) influence referenced "satellites" with a negative score, on the contrary "satellites" influence referenced "nuclei" with a positive score: the score of each sentence is the linear combination of all the votes expressed by other sentences; the vote that a sentence express to another one is the similarity between two sentences: in our summarizer we exploit the "PacSum" [41] python implementation of [24], based on tf-idf and cosine similarity for sentences representation and scoring. PacSum contains also the implementation based on BERT that we don't exploit because we focused on unsupervised summarization methods.

- Centroid based summarizer [7]: sentences are gathered into event-based cluster, i.e. set of sentences that talk about the same sub-event: the method uses CBSU(cluster-based sentence utility) to compute sentence score according to the number of contained "centroid words" (words that have an high IDF score), while it

uses CSIS(Cross-sentence informational subsumption) between the sentence and the already selected ones, to compute the "redundancy penalty" to remove from the sentence in analysis. We exploit the Sumpy implementation of this algorithm.

- MMR summarizer[10]: the aim is to maximize marginal relevance of the input by choosing sentences whose similarity with the input query is high and whose redundancy, similarity with the already selected sentences, is low: the score of each sentence at each passage is so computed as the linear combination of similarity score with the query(positive contribution) and similarity score with already selected sentences (negative contribution): at each iteration the sentence with the maximum score is selected and added to the summary. We exploited the "Vishnu" [42] implementation of the described algorithm.

- CoreRank summarizer [4]: graph-based algorithm in which each node of the graph is a unique unigram from the original text; nodes are connected from a weighted edge representing the number of co-occurrences in a fixed dimension window of the represented unigrams; k-core decomposition is performed and the score of each node is computed as the summation of the **core numbers** of neighbors, where the core number is the highest order of a core that the node belongs to. Top p percent of nodes are selected as keywords and then candidate summaries (combination of N sentences from the input text) are evaluated in terms of total number of keyword multiplied by each one score and in terms of Diversity or "Keyword coverage", i.e. the number of unique keywords contained with respect to the total number of unique keywords. Also in this case we propose a python implementation of this algorithm.

- Clusterization based summarizers [6]: algorithm that receives as input lecture transcriptions and uses BERT, using the default pre-trained model, to generate sentence embeddings and gathers them into clusters using K-Means clustering, to extract at the end

from each cluster the sentence that is the nearest to the centroid of the cluster: the sentences from the k clusters will generate a summary of k sentences. We propose two different python implementation of this algorithm: they both use RoBERTa [43], which is an extension of BERT that train the model longer,over more data and on longer sequences, removing with respect to BERT the next sentence prediction objective training on longer sequences [44], in combination with SBERT [45] in order to obtain semantically meaningful sentence embeddings that can be compared,thanks to this,using cosine similarity[46]; both Summarizers exploit K-Means to perform clusterization as in the cited paper [6] and then they use respectively cosine distance (ClusterCosine summarizer) and Euclidean distance (ClusterEuclidean summarizer) to compute,for each cluster, distance between the centroid and all the other sentences of the cluster, in order to extract, as previously described, sentences that will form the final summary.

Our framework performs date summarization using a "Summarizer" that internally uses an actual summarizer that is the implementation of one of the previously mentioned algorithms; we developed several "Summarizers": all of them have the same structure. i.e. they receive as input the set of sentences for the current date, the minimum sentence lenght with a default value of ten words, optionally a model (but we only use unsupervised approach so we never pass a model to the Summarizer) and the maximum number of sentences to extract, they perform "Short Sentence Removal" and then text summarization on the filtered sentences using a summarizer (that implements a summarization algorithm) to give N sentences (or all the sentences if their total is minus than N) as output for the current date: the difference between the "Summarizers" is that each one of them exploit a different summarizer so a different algorithm; the final output is a textual file that contains each important date (received as input of this step) followed by the selected sentences for that date; note that all the date contained in the output file will have been summarized using **the same "Summmarizer"** (Figure 4.3): we can obtain several output files by repeating the date summarization step using different "Summarizers":

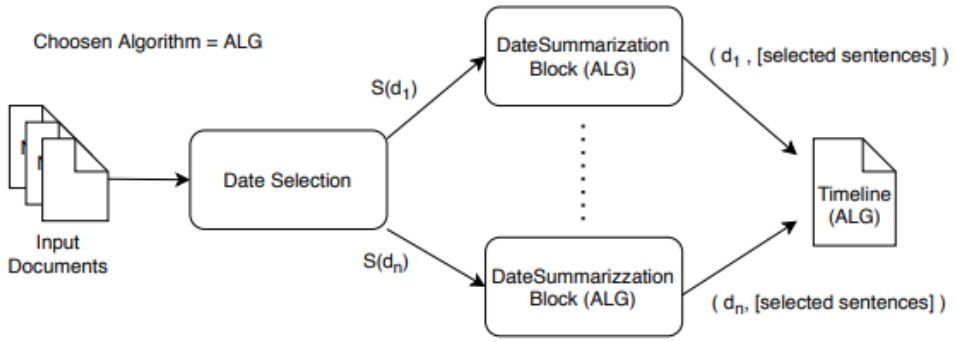

Figure 4.3: Graphical representation of the framework: for each important date,extracted in the date selection step, the same algorithm ALG is applied to create the final timeline by extracting sentences from the input ones using the date summarization block of Figure 4.2

each of the textual output file will represent a timeline obtained with a specific text summarization algorithm. The output file is given as input to a web application that performs, as we will see in the next chapter, timeline visualization.

Chapter 5

Visual Summary Exploration

In this chapter we will present the web application that we developed to show the output file's content in a timeline. TimelineVisualizaton is a minimal web application that receive as input the file containing the timeline obtained as output from our framework: in the output file each timeline entry is characterized by a date, a list of sentences (extracted to be the summary for that date) and a separator string to identify the beginning of a new timeline entry:

```
2011-03-10
Saleh says by the end of this year a new constitution will transfer power from the president to a parliamentary system .
--------------------------------
2011-03-18
After a deadly attack on protesters that kills dozens , Yemeni authorities declare a nationwide state of emergency .
--------------------------------
---- -- --
```

Figure 5.1: textual file containing the list of sentences extracted for each date.

the aim of this application is to provide users with a comfortable way of exploring the content of the generated timeline, providing a visual representation of the timeline that highlights the dates, the time distance between two sub-events and that enrich the timeline with representative photos, of the less user-oriented textual representation of the timeline contained in the output file.

TimelineVisualization consists of an initial homepage that, as we can see in figure 5.2, contains one input button used to insert the file containing the timeline, and a button to start files processing: the timeline file is mandatory (an error popup is shown if we try to proceed without inserting a file) and it must have the structure showed in Figure 5.1, i.e. it must contain the timeline to visualize.

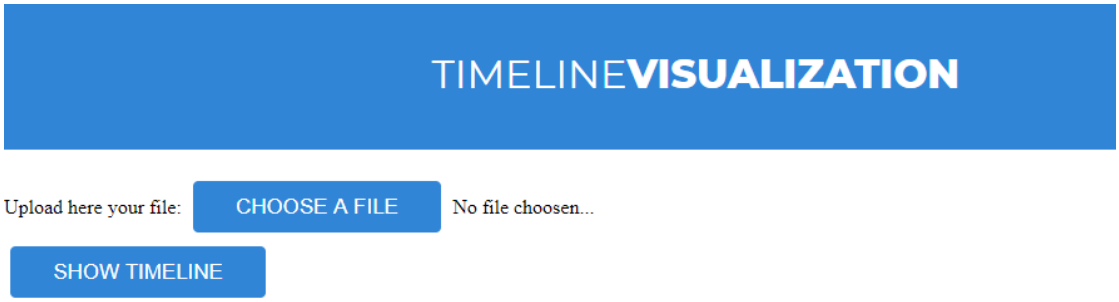

Figure 5.2: homepage of TimelineVisualization

Once we select the timeline input file, it is processed and for each date in the file a vis.js (library that we exploited to construct the timeline) compatible object is created to be inserted in the final list of "Timeline entries" to show: the final result is a size scalable (to show timeline in a more compact way or to higlight the date corresponding to each entry) timeline in which for each date an entry,containing the selected sentences for that date, is created and put in the corresponding timepoint in the timeline; the result is illustrated in Figure 5.3.

As we can see in figure 5.3 each timeline entry is a block that contains the date, a preview of the summary for that date and a button: clicking on that button appears a popup, realized exploiting sweetalert library, that contains the entire final summary extracted for that date (Figure 5.4). Moreover, the popup will show a representative image obtained using "PHP Simple HTML DOM Parser" [47] to get from Google the most correlated image associated to the input query which, in our case, is the full summary associated to the timeline entry whose "Show Content" button has just been clicked.

As previusly mentioned,in order to realize the timeline we exploited

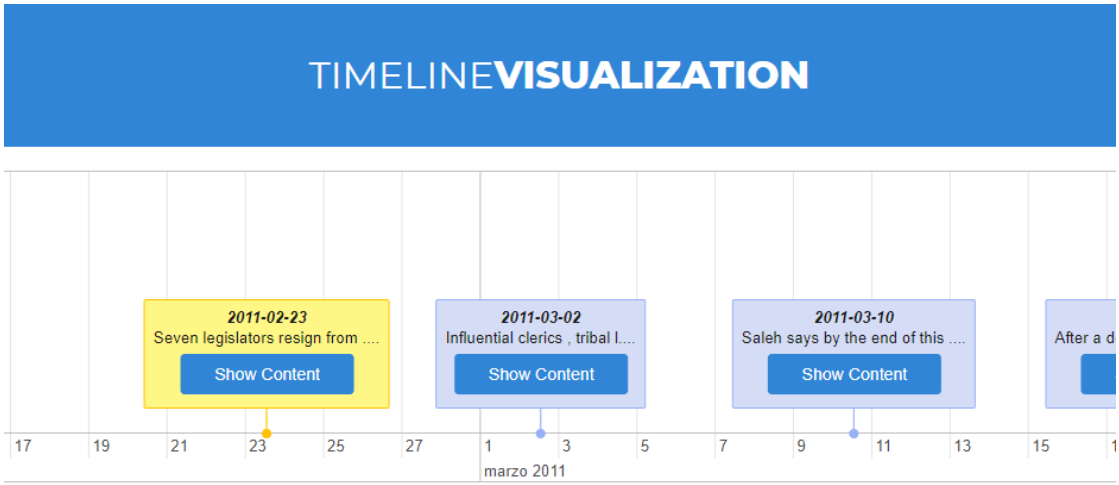

Figure 5.3: final timeline visualization

Figure 5.4: popup showing the finanl summmary extracted for the selected date

vis.js [48] which is an open source dynamic, browser based visualization library that contains several component: in particular we exploited the "Timeline" component [49] which is an interactive visualization chart to visualize data in time, choosing one of the proposed templates [50]

as starting point for the visualizzation: we adapted the template to our purpose in order to build the timeline entry as in figure 5.3. We exploited SweetAlert [51] to show the final summary for each date: SweetAlert is a library that provides to developers several template to customize popups in web applications.

5.1 Possible extensions of the visual interface

The presented application could be improved in future works: as previously said it receive a single file containing the summary for each date: the summary is used as searching query for a Google research that is performed runtime while graphically constructing the timeline: this could be more ore less computationally expensive depending on the number of timeline entries: so the first future extension that we identified is a modification that involve also the previous blocks of the pipeline: it consists in a system that extracts from the input documents also urls of the contained images and associates them to a date, obtaining a second file structured as a list of couples (date, image url) that can be read as second file from the application: while processing the first file to build the timeline entry, the application will search in the second file the image url that is eventually associated to the date in analysis and it will associate the found url to the timeline entry of the same date: so an image will be showed in the pop-up as in figure 5.4 only if it has been previously extracted and associated to the date reported in the popup (the date in analysis); moreover, in the current implementation of the application we take the first image obtained from google for the input query: a future extension could perform search optimization to obtain an image that respects some constraints such as dimension or publication date constraints.

Chapter 6

Experimental Results

As previously said we propose several Summarizer for date summarization task, each one of them exploiting a specific implementation of a specific summarization algorithm: the goal of this chapter is to compare the performance of the summarization algorithms used in the date summarization step: for this purpose we will provide the results obtained evaluating the timeline generated by each algorithm, tested on two different dataset, T17 and CRISIS, using some variant of the ROUGE metric adapted to timeline evaluation.

6.1 Evaluation metrics

Martschat and Markert [52] present an innovative method to evaluate timelines obtained by the TLS process: it takes origin from the most diffused evaluation metrics, i.e. ROUGE, in particular ROUGE-N which uses N-gram (the most used are ROUGE-1 and ROUGE-2 that work with unigram and bigram respectively) overlapping between the system generated summary and a set of reference summaries to evaluate the summary, assigning to it a recall, precision and f-measure (harmonic mean of recall and precision) score. ROUGE [53] is a metric used to evaluate a single summary extracted from a text summarization algorithm against a set of ideally summaries, manually created by humans: several metrics have been introduced by authors in [53]:

- as previously anticipated, the most used Rouge metric is ROUGE-N that is computed as the maximum number of N-grams that co-occur in the generated summary and in reference summary, divided by the total number of N-grams that occurs in the reference summary (if the number of reference summary is greater than one, the ROUGE-N score is computed against each reference summary and the highest is kept as ROUGE-N score of the generated summary: this criterion is used also for other rouge metrics)

- ROUGE-L (longest common subsequence): summary sentences are seen as a sequence of words: the longer is the LCS score between two sentences the higher two sentences will be similar: the total score of the summary is the summation of all the LCS score between each sentence of the generated summary and each sentence of the reference summary

- ROUGE-W, is weighted LCS: in this case the LCS score will be higher if the n-gram co-occur near to each other, i.e. if we have a reference sentence [A B C D E F G] and we consider unigrams and two generated sentences $S_1 = [ABZKLCD]$ and $S_2 = [ABCDKLM]$ then even if the longest unigram common subsequenc is equal to 4, S_2 is more similar to the reference and so it will receive an higher score.

As we just described, ROUGE compare the generated summary to one (or more) reference summary: in TLS we will have a summary for each date, so several adaptation of ROUGE to TLS have been proposed to evaluate the overall timeline:

- **Concatenation-based ROUGE**: this metric computes ROUGE score between two summaries obtained concatenating respectively the summaries computed by the system for each date of the generated timeline (summary to evaluate) and the summaries of the reference timeline (reference summary): in this way the overall summary is evaluated but all time-related information are lost.

- **Date-agreement ROUGE**: this method evaluates the quality of each date summary by computing recall, precision and f-measure

between the generated summary for date D_{gen} and a set of reference summaries for date D_{ref}: the advantage is that temporal dimension is taken into account, while the disadvantage is that D_{ref} and D_{gen} must match exactly (otherwise 0 is assigned as score for D_{gen} generated summary)

Authors propose a third variant of ROUGE called **Alignment-based ROUGE**: the aim of this variant of ROUGE is to take into account the temporal similarity and semantic similarity of daily summary between the generated timeline and a set of reference timelines: the idea is that each generated daily summary should be compared to a set of reference daily summaries that are close in time to the generated daily summary in analysis: so Date Alignment is performed: each date of the reference timelines is assigned to one date of the generated timeline according to time distance(in terms of number of days that separate the two dates)- **Date Alignment**- and also content similarity (similarity between the daily summaries) can be considered (**Date-content Alignment**), assigning a reference date to a computed date (through an injective function, so there will be a 1:1 relationship between aligned dates) with the aim of minimizing the cost of Date Alignment (the more will be the time distance between the dates the higher will be the alignment cost, the more the date-content will be different the higher will be the date-content alignment cost). Then the ROUGE precision, recall and f-measure score will be computed between each date summary and the reference summary, i.e. the summary of the date aligned to the date in analysis: the score will be influenced by a weighting factor that is inversely proportional to the time distance of the two dates involved in the computation. A variant of this method is Many-to-one Date-content (align+ m:1) that remove the injectivity from the alignment function.

We used these metrics to evaluate timelines extracted by our Summarizers from two different datasets: 17 Timelines (T17) and CRISIS. In particular the most appropriate metrics to evaluate the generated timelines are concatenation-based Rouge, optimal to evaluate the content of the summaries (even if the time information is, in this case, lost), date-agreement, that take, instead, into account temporal information

and date-content alignment many-to-one that compares the summary of a date to a set of reference daily summary "aligned" to the date in analysis based both on date nearness and content similarity: in particular we used these metrics using Rouge-1 and Rouge-2.

6.2 Datasets description

Dataset "17 Timelines" (T17) is the corpus of documents introduced by Tran et al. [21]: authors collected several timelines published by popular news agency about famous topics: among these timelines only the ones in which timestamps was explicit dates (i.e. dates including at least day,month,year) was kept as starting point for dataset construction: 17 timelines have been obtained in this first step: then for each timeline authors used Google to look for articles published in the temporal range of the timeline in analysis by the news agency that published the timeline in analysis: the top 400 returned articles was kept as part of the corpus, obtaining, after the duplication removal step, 4650 news articles that have been split, in the last step, according to their topic, into training and testing sets: we used the testing sets of T17 to evaluate the score of timelines generated by our Summarizers using the previously mentioned metrics.

Dataset CRISIS, proposed by Tran et al.[18] is a corpus composed of news article, published by some important news agency and used as input, and of several expert timeline summaries used as reference in evaluation: the articles talk about several crisis: wars in Lybia and Syria, crisis in Yemen and revolution in Egypt. Authors started from the timelines and constructed some queries, specifying location, long-running event such as crisis or war, to use for a Google time-filtered research obtaining in this way 15.534 articles about the four previously mentioned events in the temporal range of the events; 25 timelines have been extracted from 24 popular news agency (the same of the previous step) to be considered as ground-truth timelines, while the retrieved document are used as input.

6.3 T17 results

Each of the tables reported in this section represent the scores obtained by all the Summarizers used to obtain a timeline from the T17 corpus: as we can see in the tables below the Summarizer that exploit the implementation of TextRank in combination with BM25 [40] obtained the best result in term of Precision according to all the metrics except for concat-Rouge-1 (Table 6.1) in which SummarizerSumpyLede obtained a Precision score of 0.403 which is the best according to this metric; CoreRank obtained the second best result according to all metrics except for concat-Rouge-1, in which TextRankBM25 was the second best result; the "Sumpy" [39] implementation of the Centroid Based Summarizer [7] obtained the best result in terms of Recall according to all of the metrics except for agreement-Rouge-2 (Table 6.4) and in date alignment Rouge-2 (Table 6.6) in which TextRankBM25 was the best performing in terms of Precision. TextRankBM25 was the best performing Summarizer (F-Measure) according to all the adopted metrics. The second best performing Summarizer is SummarizerCoreRank that defeated all the other Summarizers (except for TextRankBM25) in terms of F-Measure according to each one of the adopted evaluation metrics, except for concat-Rouge-1 (Table 6.1), according to which the Sumpy [39] implementation of LexRank [5] obtained the second best result (0.379 against 0.367 of CoreRank that is the fourth best result according to this metric). Good results have been obtained by SumpyTextRank [3] that obtained,according to concat-Rouge metrics (Table 6.1 and Table 6.2) a score of 0.370 against 0.379 of SumpyLexRank using concat-Rouge-1 (the best, TextRankBM25, obtained 0.397) and 0.077 against 0.082 of SumpyLexRank and 0.083 of CoreRank using concat-Rouge-2 (the best,TextRankBM25, obtained 0.091). As we can see from Tables from 6.3 to 6.6 SumpyLexRank obtained the third best result using all these metrics (following TextRankBM25 and CoreRank); good results have been obtained also by clusterization based summarizers (ClusterCosine, ClusterEuclidean and Centroid) and by SumpyTextRank.

concat-Rouge-1			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.354*	0.329*	0.341*
SumpyTextRank	0.307*	0.467*	0.370*
SumpyLede	**0.403**	0.242*	0.302*
SumpyCentroid	0.282*	**0.505**	0.362*
VishnuMMR	0.394	0.330*	0.359*
SumpyLexRank	0.320*	0.464*	0.379*
SummarizerCorerank	0.384*	0.350*	0.367*
SummarizerClusterCosine	0.361*	0.353*	0.357*
SummarizerClusterEuclidean	0.364*	0.361*	0.363*
TextrankBM25	0.390*	0.404*	**0.397**

Table 6.1: Results obtained testing algorithm on T17 and evaluating results using concat-Rouge-1. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result.

concat-Rouge-2			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.066*	0.062*	0.064*
SumpyTextRank	0.064*	0.097*	0.077*
SumpyLede	0.073*	0.042*	0.053*
SumpyCentroid	0.058*	**0.104**	0.074*
VishnuMMR	0.083	0.069*	0.075*
SumpyLexRank	0.069*	0.101*	0.082*
SummarizerCorerank	0.087	0.079*	0.083
SummarizerClusterCosine	0.075*	0.073*	0.074*
SummarizerClusterEuclidean	0.075*	0.076*	0.076*
TextrankBM25	**0.089**	0.092*	**0.091**

Table 6.2: Results obtained testing algorithm on T17 and evaluating results using concat-Rouge-2. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result.

agreement-Rouge-1			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.067*	0.063*	0.065*
SumpyTextRank	0.066*	0.104*	0.080*
SumpyLede	0.076*	0.042*	0.054*
SumpyCentroid	0.063*	**0.117**	0.082*
VishnuMMR	0.086*	0.072*	0.078*
SumpyLexRank	0.072*	0.108*	0.086*
SummarizerCorerank	0.093*	0.087*	0.090*
SummarizerClusterCosine	0.086*	0.083*	0.084*
SummarizerClusterEuclidean	0.087*	0.085*	0.086*
TextrankBM25	**0.104**	0.107	**0.106**

Table 6.3: Results obtained testing algorithm on T17 and evaluating results using agreement-Rouge-1.* indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result.

agreement-Rouge-2			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.012*	0.012*	0.012*
SumpyTextRank	0.014*	0.024*	0.018*
SumpyLede	0.016*	0.010*	0.012*
SumpyCentroid	0.014*	0.028	0.019*
VishnuMMR	0.020*	0.017*	0.018*
SumpyLexRank	0.018*	0.028	0.022*
SummarizerCorerank	0.024	0.024*	0.024*
SummarizerClusterCosine	0.020*	0.020*	0.020*
SummarizerClusterEuclidean	0.021*	0.021*	0.021*
TextrankBM25	**0.030**	**0.031**	**0.031**

Table 6.4: Results obtained testing algorithm on T17 and evaluating results using agreement-Rouge-2. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result.

align+ m:1 ROUGE-1			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.071*	0.064*	0.067*
SumpyTextRank	0.072*	0.109*	0.086*
SumpyLede	0.082*	0.046*	0.059*
SumpyCentroid	0.068*	**0.124**	0.088*
VishnuMMR	0.096*	0.076*	0.085*
SumpyLexRank	0.078*	0.112*	0.092*
SummarizerCorerank	0.103*	0.090*	0.096*
SummarizerClusterCosine	0.090*	0.084*	0.087*
SummarizerClusterEuclidean	0.092*	0.087*	0.089*
TextrankBM25	**0.112**	0.113	**0.112**

Table 6.5: Results obtained testing algorithm on T17 and evaluating results using Date-content alignment many to one (align+ m:1) ROUGE 1. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result.

align+ m:1 ROUGE-2			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.010*	0.008*	0.009*
SumpyTextRank	0.014*	0.020*	0.016*
SumpyLede	0.015*	0.009*	0.011*
SumpyCentroid	0.014*	0.025	0.018*
VishnuMMR	0.021*	0.016*	0.018*
SumpyLexRank	0.018*	0.025	0.021*
SummarizeCorerank	0.024	0.020*	0.022*
SummarizerClusterCosine	0.019*	0.017*	0.018*
SummarizerClusterEuclidean	0.019*	0.017*	0.018*
TextrankBM25	**0.029**	**0.029**	**0.029**

Table 6.6: Results obtained testing algorithm on T17 and evaluating results using Date-content alignment many to one (align+ m:1) ROUGE 2. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result.

6.4 CRISIS Results

In this section we report, as we did with T17 in the previous section, tables illustrating the results in terms of Precision, Recall and F-Measure obtained by the developed Summarizers: VishnuMMR, the Summarizer that exploits the implementation of a MMR summarizer, was the best performing in terms of Precision according to concat-Rouge-1 (Table 6.7), obtaining a Precision score of 0.259 against 0.245 of Sumpylede and 0.240 of CoreRank, which obtained the best Precision result according to all the other metrics: as we can see in Table 6.10 the ClusterCosine summarizer [6] obtained the same result of CoreRank and TextRankBM25 (0.009); SumpyCentroid obtained the best result in terms of Recall score according to concat-Rouge-1 (Table 6.7) and concat-Rouge-2 (Table 6.8), defeating TextRankBM25 Summarizer that instead obtained the best Recall score according to all the other metrics; the best performing

algorithm is VishnuMMR according to concat-Rouge-1, in which it obtained the best score in terms of F-Measure(0.331) followed by CoreRank (0.322) which is the best performing algorithm according to all the other metrics, obtaining the same results of TextRankBM25: only according to agreement-Rouge-1 TextRankBM25 is the only best performing. ClusterCosine and ClusterEuclidean summarizers, obtained good results, obtaining the second and third best result (F-Measure) in all the metrics except for concat-Rouge-1 (6.7) and concat-Rouge-2 (6.8) in which the second best result was obtained by TextRankBM25; good results was obtained also by the Sumpy implementation of TextRank.

concat-Rouge-1			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.224*	0.437*	0.296*
SumpyTextRank	0.167*	0.616*	0.263*
SumpyLede	0.245	0.295*	0.268*
SumpyCentroid	0.149*	**0.653**	0.243*
VishnuMMR	**0.259**	0.458*	**0.331**
SumpyLexRank	0.178*	0.591*	0.273*
SummarizerCorerank	0.240*	0.491*	0.322
SummarizerClusterCosine	0.232*	0.498*	0.317*
SummarizerClusterEuclidean	0.232*	0.502*	0.318*
TextrankBM25	0.230*	0.546*	0.324

Table 6.7: Results obtained testing algorithm on CRISIS and evaluating results using concat-Rouge-1. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result.

concat-Rouge-2			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.033*	0.064*	0.043*
SumpyTextRank	0.032*	0.117*	0.050*
SumpyLede	0.028*	0.032*	0.030*
SumpyCentroid	0.029*	**0.128**	0.047*
VishnuMMR	0.046*	0.082*	0.059*
SumpyLexRank	0.035*	0.118*	0.054*
SummarizerCorerank	**0.053**	0.109*	**0.071**
SummarizerClusterCosine	0.042*	0.088*	0.057*
SummarizerClusterEuclidean	0.042*	0.090*	0.057*
TextrankBM25	0.046*	0.109*	0.065*

Table 6.8: Results obtained testing algorithm on CRISIS and evaluating results using concat-Rouge-2. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result.

agreement-Rouge-1			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.025*	0.049*	0.033*
SumpyTextRank	0.024*	0.090*	0.038*
SumpyLede	0.019*	0.022*	0.020*
SumpyCentroid	0.021*	0.094	0.034*
VishnuMMR	0.037	0.068*	0.048*
SumpyLexRank	0.025*	0.088*	0.040*
SummarizerCorerank	**0.043**	0.088	0.057
SummarizerClusterCosine	0.038	0.080*	0.051*
SummarizerClusterEuclidean	0.038	0.081*	0.051*
TextrankBM25	0.041	**0.098**	**0.058**

Table 6.9: Results obtained testing algorithm on CRISIS and evaluating results using agreement-Rouge-1.* indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result.

agreement-Rouge-2			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.003*	0.005*	0.004*
SumpyTextRank	0.004*	0.018*	0.007*
SumpyLede	0.001*	0.001*	0.001*
SumpyCentroid	0.004*	0.019	0.006*
VishnuMMR	0.008	0.015*	0.011
SumpyLexRank	0.006*	0.021	0.009*
SummarizerCorerank	**0.009**	0.020	**0.013**
SummarizerClusterCosine	**0.009**	0.017	0.011
SummarizerClusterEuclidean	0.008	0.016	0.011
TextrankBM25	**0.009**	**0.022**	**0.013**

Table 6.10: Results obtained testing algorithm on CRISIS and evaluating results using agreement-Rouge-2. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result (We considered TextRankBM25 as the best performing in terms of Precision and F-Measure).

align+ m:1 ROUGE-1			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.037*	0.062*	0.047*
SumpyTextRank	0.036*	0.114*	0.055*
SumpyLede	0.030*	0.031*	0.030*
SumpyCentroid	0.032*	0.119	0.051*
VishnuMMR	0.055*	0.085*	0.067*
SumpyLexRank	0.038*	0.110*	0.057*
SummarizerCorerank	**0.062**	0.106*	**0.078**
SummarizerClusterCosine	0.053*	0.097*	0.069*
SummarizerClusterEuclidean	0.054*	0.098*	0.069*
TextrankBM25	0.058	**0.120**	**0.078**

Table 6.11: Results obtained testing algorithm on CRISIS and evaluating results using Date-content alignment many to one (align+ m:1) ROUGE 1. * indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result (We considered TextRankBM25 as the best performing in terms of F-Measure).

align+ m:1 ROUGE-2			
Summarizer	**Precision**	**Recall**	**F-Measure**
SumyLexrank	0.004*	0.007*	0.005*
SumpyTextRank	0.006*	0.021*	0.009*
SumpyLede	0.002*	0.003*	0.002*
SumpyCentroid	0.006*	0.022	0.009*
VishnuMMR	0.010	0.017*	0.013
SumpyLexRank	0.008*	0.024	0.011*
SummarizerCorerank	**0.013**	0.023	**0.017**
SummarizerClusterCosine	0.011	0.020	0.014
SummarizerClusterEuclidean	0.011	0.018*	0.014
TextrankBM25	0.012	**0.025**	**0.017**

Table 6.12: Results obtained testing algorithm on CRISIS and evaluating results using Date-content alignment many to one (align+ m:1) ROUGE 2.* indicates that result obtained by the best performing algorithm is statistically relevant against the current algorithm's result (We considered TextRankBM25 as the best performing in terms of F-Measure).

Chapter 7

Conclusion and Future Work

The increasing amount of news article about several topics/events force the reader, that is interested in a specific event, to look for information from several sources, published in several website: it would be useful for the reader to be provided with a complete and concise summary, that highlights the main information about an event, or to be kept updated about the developing of an event with novel updates, or to have a panoramic visualization of the developing of a concluded event during time: these are the reasons that made the research in NLP field move to text summarization, temporal summarization and timeline summarization. In this thesis we focused our attention on timeline summarization and text summarization; text summarization is the branch of NLP that aims to extract a summary from an input text (that can be a single document or a set of document in case of Multi-document summarization) and it comprises extractive techniques, aimed at extracting the most important sentences from the input documents, and abstractive ones, whose goal is to generate new content according to end-user needs. Text summarization does not take into account temporal information, timeline summarization has instead the aim of solving this text summarization issue, providing to the user a cross-temporal representation of the developing of an event along a timeline that highlights the most significant dates in a range that goes from the

beginning to the end of the considered event.

In first place we examined the state of the art in text, temporal and timeline summarization, describing several algorithm that have been presented in past years but the contribution brought with this thesis is a new pipeline for timeline summarization, that use a date-wise approach so from the extracted dates it then selects the important sentences: the proposed pipeline is composed by three blocks: the first is the date selection block, but in this thesis we focused on the second block, which is the date summarization block, and the third block, which is the timeline visualization block: the first block extract, from the input sentences, the most significant dates and then the second block perform text summarization to extract, for each date, the most important sentences: we implemented several Summarizers each one of them exploiting one of the state-of-the-art methods that we previously described: in our work we used, as in most of the SOTA works, the extractive summarization approach and all ours summarizers were unsupervised.

One of the goal of this thesis is to compare the performance of the several state-of-the-art algorithm that we re-implemented: for this purpose we tested our framework on two different datasets using all the proposed Summarizers: we discarded some of them and we summarized in some tables the results obtained by the good-performing algorithms evaluated using several variants of rouge metrics: as conclusion we can say, as highlighted in the previously mentioned tables, that graph-based algorithms were the best performing but also clusterization based summarizers obtained good results: in particular TextRankBM25, i.e. the summarizer that exploit the graph-based TextRank algorithm integrating BM25 as similarity measure, was the algorithm that performed significantly better than all the other algorithms according to the most part of exploited evaluation metrics.

Another contribution of the thesis is the visualization of the produced timeline: the output of the date summarization step could be given as input to the web application that we propose, which show the content of the file given as output by the date summarization block in a graphical timeline.

We can conclude saying that we presented a timeline summarization framework that obtained fairly good results depending on the used text summarization algorithm for the date summarization step and that provide an user-friendly way of visualizing and exploring content of the generated timeline.

7.1 Future works

In this section we propose several possible future extensions, aimed at improving the performance of the overall pipeline.

The first possible future variant of the system that we propose could move to the Direct Summarization approach, in which dates are consequences of the selected sentences: the intuition behind the proposed framework consists in selecting dates and then selecting important sentences for each date: the proposed extension would perform text summarization directly on dated sentences to select the most important sentences and then extract from each sentence the corresponding date that will constitute one entry of the timeline,along with all the sentences associated to that date.

Remaining instead in the date-wise approach field, the first possible future extension consist in the the employment of other unsupervised text summarization algorithms or in the development of a new summarization algorithm from scratch or combining techniques of the state of the art, trying to improve the performance of the already implemented ones.

Another possible future extension could instead integrate into the date summarization step the usage of supervised technique, that could exploit default pre-trained models or that could train model on the training sets provided by the datasets that we employed, maybe using deep learning techniques based on Neural Network; all our summarizers are already predisposed to accept a model as input, for text summarization algorithm that would use a model to guess if a sentence would be or not an important sentence.

The last proposed extension involves also the developed web application: in current version our application receive as input a single

file containing the summary for each date: the summary is used as searching query for a Google research that is performed at runtime while graphically constructing the timeline: this could be more ore less computationally expensive depending on the number of timeline entries: the future extension that we propose consists in integrating into the sysetm a component that extracts from the input documents also urls of the contained images and associates them to a date, obtaining a second file structured as a list of couples (date, image url) that can be read as second file from the application: while processing the first file to build each timeline entry, the application would search in the second file the image url that is eventually associated to the date in analysis, in order to show the image in the popup representing the relative date, if it is an important date.

Bibliography

[1] Luìs Gonçalves. *Automatic Text Summarization with Machine Learning — An overview*. Apr. 2020. URL: https://medium.com/luisfredgs/automatic-text-summarization-with-machine-learning-an-overview-68ded5717a25# (cit. on pp. 6, 8).

[2] URL: https://www-nlpir.nist.gov/projects/duc/duc2007/tasks.html (cit. on p. 8).

[3] Rada Mihalcea and Paul Tarau. «TextRank: Bringing Order into Text». In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 404–411. URL: https://www.aclweb.org/anthology/W04-3252 (cit. on pp. 9, 14, 15, 45, 59).

[4] Antoine Tixier, Polykarpos Meladianos, and Michalis Vazirgiannis. «Combining Graph Degeneracy and Submodularity for Unsupervised Extractive Summarization». In: *Proceedings of the Workshop on New Frontiers in Summarization*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 48–58. DOI: 10.18653/v1/W17-4507. URL: https://www.aclweb.org/anthology/W17-4507 (cit. on pp. 9, 17, 48).

[5] G. Erkan and D. R. Radev. «LexRank: Graph-based Lexical Centrality as Salience in Text Summarization». In: *Journal of Artificial Intelligence Research* 22 (Dec. 2004), pp. 457–479. ISSN: 1076-9757. DOI: 10.1613/jair.1523. URL: http://dx.doi.org/10.1613/jair.1523 (cit. on pp. 9, 15, 46, 59).

[6] Derek Miller. *Leveraging BERT for Extractive Text Summarization on Lectures*. 2019. arXiv: 1906.04165 [cs.CL] (cit. on pp. 9, 21, 48, 49, 65).

[7] Dragomir R. Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. «Centroid-based summarization of multiple documents». In: *Information Processing Management* 40.6 (2004), pp. 919–938. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2003.10.006. URL: http://www.sciencedirect.com/science/article/pii/S0306457303000955 (cit. on pp. 9, 20, 47, 59).

[8] Dingding Wang, Shenghuo Zhu, Tao Li, Yun Chi, and Yihong Gong. «Integrating Document Clustering and Multidocument Summarization». In: *ACM Trans. Knowl. Discov. Data* 5.3 (Aug. 2011). ISSN: 1556-4681. DOI: 10.1145/1993077.1993078. URL: https://doi.org/10.1145/1993077.1993078 (cit. on pp. 9, 23, 24).

[9] Xiaoyan Cai and Wenjie Li. «A spectral analysis approach to document summarization: Clustering and ranking sentences simultaneously». In: *Information Sciences* 181.18 (2011), pp. 3816–3827. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2011.04.052. URL: http://www.sciencedirect.com/science/article/pii/S0020025511002386 (cit. on pp. 9, 23, 24).

[10] Jaime Carbonell and Jade Goldstein. «The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries». In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '98. Melbourne, Australia: Association for Computing Machinery, 1998, pp. 335–336. ISBN: 1581130155. DOI: 10.1145/290941.291025. URL: https://doi.org/10.1145/290941.291025 (cit. on pp. 9, 19, 48).

[11] Josef Steinberger. «LSA-Based Multi-Document Summarization». In: (Jan. 2007) (cit. on pp. 9, 19).

[12] Josef Steinberger and Karel Ježek. «Update Summarization Based on Latent Semantic Analysis». In: *Text, Speech and Dialogue*. Ed. by Václav Matoušek and Pavel Mautner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 77–84. ISBN: 978-3-642-04208-9 (cit. on pp. 9, 20).

[13] Josef Steinberger and Karel Jezek. «Using Latent Semantic Analysis in Text Summarization and Summary Evaluation». In: Jan. 2004 (cit. on pp. 9, 20).

[14] Elena Baralis, Luca Cagliero, Saima Jabeen, and Alessandro Fiori. «Multi-Document Summarization Exploiting Frequent Itemsets». In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. SAC '12. Trento, Italy: Association for Computing Machinery, 2012, pp. 782–786. ISBN: 9781450308571. DOI: 10.1145/2245276.2245427. URL: https://doi.org/10.1145/2245276.2245427 (cit. on pp. 9, 25–27).

[15] Elena Baralis, Luca Cagliero, Alessandro Fiori, and Paolo Garza. «MWI-Sum: A Multilingual Summarizer Based on Frequent Weighted Itemsets». In: *ACM Transactions on Information Systems* 34 (Sept. 2015), 5:1–. DOI: 10.1145/2809786 (cit. on pp. 9, 26).

[16] Luca Cagliero, Paolo Garza, and Elena Baralis. «ELSA: A Multilingual Document Summarization Algorithm Based on Frequent Itemsets and Latent Semantic Analysis». In: *ACM Trans. Inf. Syst.* 37.2 (Jan. 2019). ISSN: 1046-8188. DOI: 10.1145/3298987. URL: https://doi.org/10.1145/3298987 (cit. on pp. 9, 28, 29).

[17] Hai Leong Chieu and Yoong Keok Lee. «Query Based Event Extraction along a Timeline». In: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '04. Sheffield, United Kingdom: Association for Computing Machinery, 2004, pp. 425–432. ISBN: 1581138814. DOI: 10.1145/1008992.1009065. URL: https://doi.org/10.1145/1008992.1009065 (cit. on pp. 11, 37, 42).

[18] Giang Tran, Mohammad Alrifai, and Eelco Herder. «Timeline Summarization from Relevant Headlines». In: *Advances in Information Retrieval*. Ed. by Allan Hanbury, Gabriella Kazai, Andreas Rauber, and Norbert Fuhr. Cham: Springer International Publishing, 2015, pp. 245–256. ISBN: 978-3-319-16354-3 (cit. on pp. 11, 13, 39, 40, 42, 58).

[19] Sebastian Martschat and Katja Markert. «A Temporally Sensitive Submodularity Framework for Timeline Summarization». In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 230–240. DOI: 10.18653/v1/K18-1023. URL: https://www.aclweb.org/anthology/K18-1023 (cit. on pp. 11, 41, 42).

[20] Jiwei Li and Sujian Li. «Evolutionary Hierarchical Dirichlet Process for Timeline Summarization». In: Aug. 2013, pp. 556–560 (cit. on pp. 11, 40, 42).

[21] Giang Binh Tran, Mohammad Alrifai, and Dat Quoc Nguyen. «Predicting Relevant News Events for Timeline Summaries». In: *Proceedings of the 22nd International Conference on World Wide Web*. WWW '13 Companion. Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, pp. 91–92. ISBN: 9781450320382. DOI: 10.1145/2487788.2487829. URL: https://doi.org/10.1145/2487788.2487829 (cit. on pp. 11, 38, 42, 45, 58).

[22] Julius Steen and Katja Markert. «Abstractive Timeline Summarization». In: *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 21–31. DOI: 10.18653/v1/D19-5403. URL: https://www.aclweb.org/anthology/D19-5403 (cit. on pp. 11, 38, 42).

[23] Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. «TREC 2015 Temporal Summarization Track Overview». In: (). URL: https://trec.nist.gov/pubs/trec24/papers/Overview-TS.pdf (cit. on pp. 12, 30).

[24] Hao Zheng and Mirella Lapata. «Sentence Centrality Revisited for Unsupervised Summarization». In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 6236–6247. DOI: 10.18653/v1/P19-1628. URL: https://www.aclweb.org/anthology/P19-1628 (cit. on pp. 16, 47).

[25] Aria Haghighi and Lucy Vanderwende. «Exploring Content Models for Multi-Document Summarization». In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Boulder, Colorado: Association for Computational Linguistics, June 2009, pp. 362–370. URL: https://www.aclweb.org/anthology/N09-1041 (cit. on pp. 18, 47).

[26] Xiaojun Wan and Jianwu Yang. «Multi-Document Summarization Using Cluster-Based Link Analysis». In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08. Singapore, Singapore: Association for Computing Machinery, 2008, pp. 299–306. ISBN: 9781605581644. DOI: 10.1145/1390334.1390386. URL: https://doi.org/10.1145/1390334.1390386 (cit. on p. 22).

[27] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. *SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents*. 2016. arXiv: 1611.04230 [cs.CL] (cit. on p. 24).

[28] Xiaojun Wan. «TimedTextRank: Adding the Temporal Dimension to Multi-Document Summarization». In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '07. Amsterdam, The Netherlands: Association for Computing Machinery, 2007, pp. 867–868. ISBN: 9781595935977. DOI: 10.1145/1277741.1277949. URL: https://doi.org/10.1145/1277741.1277949 (cit. on p. 30).

[29] T. Schubotz and R. Krestel. «Online Temporal Summarization of News Events». In: *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. Vol. 1. 2015, pp. 409–412 (cit. on p. 31).

[30] Yun Zhao, Fei Yao, Huayang Sun, and Zhen Yang. «BJUT at TREC2014 Temporal Summarization Track». In: (2014) (cit. on p. 31).

[31] Richard McCreadie, Romain Deveaud, M-Dyaa Albakour, Stuart Mackie, Nut Limsopatham, Craig Macdonald, Iadh Ounis, and Thibaut Thonet. «University of Glasgow at TREC 2014: Experiments with Terrier in Contextual Suggestion, Temporal Summarisation and Web Tracks». In: (2014) (cit. on p. 32).

[32] Ahmed Tazibt and Farida Aoughlis. «Latent Dirichlet allocation-based temporal summarization». In: *International Journal of Web Information Systems* 15 (Nov. 2018). DOI: 10.1108/IJWIS-04-2018-0023 (cit. on p. 32).

[33] Rafik Abbes, Nathalie Hernandez, Karen Pinel-Sauvagnat, and Mohand Boughanem. «Détection d'informations vitales pour la mise à jour de bases de connaissances». In: June 2015 (cit. on p. 33).

[34] Chris Kedzie, Kathleen McKeown, and Fernando Diaz. «Predicting Salient Updates for Disaster Summarization». In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 1608–1617. DOI: 10.3115/v1/P15-1155. URL: https://www.aclweb.org/anthology/P15-1155 (cit. on p. 33).

[35] James Allan, Rahul Gupta, and Vikas Khandelwal. «Temporal Summaries of New Topics». In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '01. New Orleans, Louisiana, USA: Association for Computing Machinery, 2001, pp. 10–18.

ISBN: 1581133316. DOI: 10.1145/383952.383954. URL: https://doi.org/10.1145/383952.383954 (cit. on p. 34).

[36] Chunyun Zhang, Zhanyu Ma, Weiran Xu, Jiayue Zhang, and Jun Guo. «A Multi-level System for Sequential Update Summarization». In: Jan. 2015. DOI: 10.4108/eai.19-8-2015.2260848 (cit. on p. 35).

[37] Demian Gholipour Ghalandari and Georgiana Ifrim. *Examining the State-of-the-Art in News Timeline Summarization*. 2020. arXiv: 2005.10107 [cs.CL] (cit. on pp. 36, 45).

[38] URL: https://github.com/miso-belica/sumy (cit. on p. 46).

[39] URL: https://github.com/kedz/sumpy (cit. on pp. 46, 59).

[40] Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. *Variations of the Similarity Function of TextRank for Automated Summarization*. 2016. arXiv: 1602.03606 [cs.CL] (cit. on pp. 46, 59).

[41] URL: https://github.com/mswellhao/PacSum (cit. on p. 47).

[42] URL: https://github.com/vishnu45/NLP-Extractive-NEWS-summarization-using-MMR (cit. on p. 48).

[43] URL: https://ai.facebook.com/blog/roberta-an-optimized-method-for-pretraining-self-supervised-nlp-systems/ (cit. on p. 49).

[44] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL] (cit. on p. 49).

[45] URL: https://www.sbert.net/ (cit. on p. 49).

[46] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: 1908.10084 [cs.CL] (cit. on p. 49).

[47] URL: https://simplehtmldom.sourceforge.io/ (cit. on p. 52).

[48] URL: https://visjs.org/ (cit. on p. 53).

[49] URL: https://visjs.github.io/vis-timeline/docs/timeline/ (cit. on p. 53).

[50] URL: https://visjs.github.io/vis-timeline/examples/timeline/ (cit. on p. 53).

[51] URL: https://sweetalert2.github.io/ (cit. on p. 54).

[52] Sebastian Martschat and Katja Markert. «Improving ROUGE for Timeline Summarization». In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 285–290. URL: https://www.aclweb.org/anthology/E17-2046 (cit. on p. 55).

[53] Chin-Yew Lin. «ROUGE: A Package for Automatic Evaluation of Summaries». In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013 (cit. on p. 55).