

Persons of Interest

Nigel Grech

Supervisor: Dr. Joel Azzopardi

Co-supervisor: Dr. Charlie Abela



Department of Intelligent Computer Systems

Faculty of ICT

University of Malta

May 2016

*Submitted in partial fulfillment of the requirements for the
Degree of B.Sc. ICT Artificial Intelligence (Hons.)*

Abstract:

In our culture there is a deep interest in the lives of public figures and noteworthy individuals. There are genres of books, websites even television and YouTube channels dedicated to examining their lives. Timelines are a visual medium through which the moments in these individuals' lives can be expressed. However for the researcher, journalist or educator who has the task of compiling these timelines a great deal of time and effort is required. To our knowledge there exists no automated system that can analyse text documents to generate a human readable timeline.

In this research, we propose a solution to this problem by providing a system which analyses a source document and generates a timeline of all the relevant moments in a targeted person's life. The method proposed consists of three main components: firstly obtaining and managing relevant structured information. Secondly obtaining unstructured text and preprocessing it by means of tokenization, part of speech tagging and named entity recognition. After collecting and preparing the required data, all relevant information needs to be extracted from it. This final step will involve extracting dates and related phrases. The relevance of these phrases to the timeline will then be determined, this process will be aided by comparing phrases with statements obtained from structured information. In cases where structured information provides no assistance the system will then have to determine if a phrase is relevant to the individual in question and the context of a timeline.

Finally the events that have been extracted need to be compiled into a timeline and visualised for a human reader. We do this through the use of a JavaScript application that allows the user to interact with the timeline through a browser.

In this prototype we have mainly focused on politicians and public figures. Working with data obtained from Wikipedia and WikiData. However such a system can be extended to work in different text sources and different types of people.

Contents

1	Introduction	1
1.1	Problem Definition and Motivation	1
1.2	Proposed Solution	3
1.3	Aims and Objectives	3
1.4	Document Overview	4
2	Background and Literature Review	4
2.1	Background	4
2.1.1	Wikipedia	4
2.1.2	WikiData	4
2.2	Literature Review	4
2.2.1	Temporal Information Retrieval Systems and Techniques	5
2.2.2	Visualisation of Temporal and Structured Information	9
3	Design	11
3.1	High Level System Architecture	11
3.2	Structured Data Processing Module	12
3.3	Text Processing Module	13
3.4	Event Extraction Module	15
4	Methodology	19
4.1	Structured Data Processing Module	19
4.2	Text Processing Module	19
4.3	Event Extraction Module	21
4.3.1	Text Processing	22
4.3.2	Date Extraction	23
4.3.3	Single Date Processing	24
4.3.4	Multi-Date Processing	26
4.3.5	Timeline Generation	27
5	Evaluation and Testing	29
5.1	Similar Systems Evaluation	29
5.2	Ideal Evaluation Plan	29
5.3	Experiments	30

5.3.1	Date Extraction Evaluation	31
5.3.2	Multi-Date Phrase Extraction Evaluation	32
5.3.3	Evaluation of Timeline Events	33
5.4	Testing	35
6	Conclusion and Future Work	36
6.1	Accomplishments and Limitations	36
6.2	Future Work	37
6.3	Final Remarks	39

List of Figures

1	An example of the structured data available from WikiData	2
2	Temporal document annotation model [1].	7
3	High level architecture of PoI system	11
4	Design for the Structured Data Processing Module	13
5	Design of the Text Processing Module	14
6	Steps in the Event Extraction Module	16
7	Steps in the Text Processing Module	20
8	Three phases of Event Extraction	22
9	Overview of Text Processing	23
10	Overview of Single Date Processing	25
11	Overview of Multi-Date Processing	26
12	A Generated Timeline	28

List of Tables

1	Date extraction evaluation results.	32
2	Multi-Date phrase extraction evaluation results.	33
3	Timeline Events evaluation results.	34

Acknowledgements:

My sincere appreciation and gratitude is extended to Dr. Joel Azzopardi and Dr. Charlie Abela for their unwavering help throughout this research. Both going above and beyond in the help, expertise and encouragement they provided.

Abstract: In this research we propose a solution to the problem of the automated generation of timelines for the domain of the lives of prominent people. We present our methods of extracting dates with the use of a rule based system. Also presetting our strategy of leveraging structured data obtained from WikiData to aid in the information retrieval process. In this research we also present the promising results obtained from evaluating the prototype developed. Finally discussing how this system can be improved upon.

1 Introduction

The volume of information on the World Wide Web is expanding at an ever increasing rate, this is true for both structured information such as RDF and unstructured sources such as websites and text documents. Within this huge collection there is a great wealth of information regarding public figures. This can easily be seen by performing an on-line search for some public figure, say Nelson Mandela which will result in over 50 million results, of course this number will be substantially lower for a lesser known person.

1.1 Problem Definition and Motivation

When considering the task of compiling a list of the relevant events in a person's life, such as a timeline, it is not feasible to read through all available documents. Normally a researcher would start from some already summarised document or official source and then research individual events as necessary using their judgement on which events are relevant. This problem can be faced by many different types of people such as journalists, researchers and educators. All of whom simply need an easy to use and understand resource that will aid them in a greater task (writing an article or teaching about a specific person's life for example).

Providing a slightly more formal description of the problem. The researcher is searching through the text for information that relate to some person of interest. Ensuring that each item of information has some temporal component or qualifier that allows it to be plotted. After collecting the information the researcher would then have to compile the information correctly into a timeline.

This task of extracting information may be very time consuming and mundane for a person to perform, which makes it ideal to be automated by an information extraction system. However this is a nontrivial task, since the system must be able to identify the relations and temporal expressions present in a document. Once it has performed this

initial task it will then need to filter out all relations which cannot be associated with a temporal expression and all the relations which are not directly related to the current person of interest. For example the phrase ‘Nelson Mandela was elected as President of South Africa in 1994’ should be resolved to the relation:

$$((NelsonMandela) \xrightarrow{elected} (PresidentofSouthAfrica)) \xrightarrow{in} (1994)$$

On the contrary if the subject of the relation is not Nelson Mandela (assuming that he is the person of interest in this case) the relation should be disregarded. And the relation should also be disregarded if there is no reference to the date or some temporal expression which can be distinguished in time.






position held		President of South Africa	 edit
		start time	10 May 1994
		replaces	F. W. de Klerk
		replaced by	Thabo Mbeki
		end time	14 June 1999
		▼ 0 references	+ add reference
		member of the National Assembly of South Africa	 edit
		▼ 0 references	+ add reference
			+ add

Figure 1: An example of the structured data available from WikiData

Valuable information pertaining to persons of interest may not only be found within text documents but may be also found within structured repositories of information- examples include knowledge bases like WikiData¹ and DBpedia² which have been compiled form through information extraction techniques and crowd sourcing. These repositories contain valuable information regarding noteworthy people and, as can be seen in the Figure 1 below, this information may also contain temporal qualifiers. This type of information may not be very useful to a researcher trying to compile a timeline. When considering an automated system the utilisation of structured information should be considered as a possible means of reducing the complexity of the information extraction process.

¹<https://www.wikidata.org>

²<http://wiki.dbpedia.org/>

1.2 Proposed Solution

To tackle these problems we propose the design and development of a prototype system which will be able to extract temporal events and plot them on a timeline. We shall refer to this prototype as the Persons of Interest (PoI) system. PoI will utilise both structured data and unstructured data. Structured information will be obtained from WikiData and the unstructured information will be scraped from Wikipedia articles.

The PoI system will allow a user to provide a targeted individual and will then proceed to generate a timeline which can be viewed through a web browser. PoI will accomplish this by performing the following phases of operation. The first is the collection, resolution and indexing of structured data from WikiData. The second is the collection and preprocessing of unstructured text from Wikipedia. Next PoI will analyse the text by means of a rule based system to extract the relevant phrases and related dates. Finally it will compile all these into a format which can then be displayed as a timeline to the user.

1.3 Aims and Objectives

The main aim of this work is to be able to autonomously produce an interactive and human readable timeline about a targeted individual. The process of completing this aim can be described through the objectives detailed below:

1. Obtaining, resolving and managing structured data from WikiData.
2. Obtaining and processing the unstructured text from Wikipedia.
 - Extracting dates from phrases.
 - Determining if a phrase is relevant to the targeted person.
 - Using structured data to aid in identifying relevant phrases.
3. Generation of the timeline.

In addition to implementing the system we shall also be evaluating the overall result and the intermediary results produced by individual subsystems.

1.4 Document Overview

The remainder of this document is organised as follows. Section 2 contains a review and analysis of technologies and systems similar to our proposed solution. Sections 3, 4 and 5 describe how the system was designed, implemented and evaluated respectively. Finally Section 6 describes future work which can be carried out and conclusions made during the implementation and evaluation of this project.

2 Background and Literature Review

2.1 Background

2.1.1 Wikipedia

Wikipedia is a free collaborative encyclopedia which is written collaboratively by its users. It boasts a huge collection of articles on almost every subject. Wikipedia is also made accessible through the MediaWiki API. The API allows users to search and retrieve articles.[2]

2.1.2 WikiData

Wikidata is a free collaborative knowledge base, launched in October of 2012. WikiData implements a simple data model to represent structured data. Within WikiData a statement is described as ‘property-value’ pairs. For example the person ‘Nelson Mandela’ is an item within the system and will have numerous property-value pairs. One such pair could ‘occupation’ ‘politician’, where occupation is considered a property and politician is its value. In this case politician is also a WikiData item and will have its own sets of property-value pairs. It should also be noted that a property-value pair may have its own subordinate property-value pairs which are referred to as qualifiers. Another important feature of WikiData is that it attempts to remain language neutral, to accomplish this each item in the knowledge base is given an id and associated with the appropriate labels in the different languages. [3]

2.2 Literature Review

The following is a review and analysis of similar works and technologies. The section is divided as follows: section 2.2.1 will discuss different types of techniques used in temporal

information extraction and systems which use them; then, Section 2.2.2 will provide an overview of the system, techniques and technologies which can use and visualise structured and temporal information. Before delving deeper into the next two sections we shall briefly discuss the nature of time and how it is expressed in text, allowing for a better appreciation of the systems and techniques described below.

In the past there have been authors such as Bruce [4] and Allen [5] who have created formal definitions or models of time as to be able to reference it. The former described time as an ordered pair of a set of time points and a relation which orders the set. The latter states that there may be up to 13 different types of relations which can distinguish two points in time. However for the work we have carried out we will consider time a bit more informally as described in [1], where time is seen as a point-in-time value. These point-in-time values may have different precision or granularity such as day, month, and year and so on. This allows time values to be physically represented in calendars or timelines.

Now let us consider time denoting expressions. According to [6] there are three way in which temporal references can be denoted:

1. Explicit reference
2. Indexical reference
3. Vague reference

The first, explicit references, refer to entries in calendar systems and time expressions such as ‘30/01/2016’. Expressions such as these are easy to plot on a timeline. Indexical references are those time expressions which can only be evaluated given some other time point, such as ‘tomorrow week’. Finally Vague references or those phrases which are difficult to accurately plot such as ‘sometime next month’. When considering the task at hand, the first class of expressions are those which are most important - this is an assumption which has been made based on the reasoning that seminal events in a person’s life will be denoted with an explicit time reference. Note that Pustejovsky et al. [7] make note of another form denoting temporal expressions that of the Implicit reference such as ‘New Year’s day’. These forms of expressions may cause issues since the actual day may vary from culture to culture. We can see this from the fact that the Chinese new year and the Western one do not fall on the same day.

2.2.1 Temporal Information Retrieval Systems and Techniques

Before examining any specific information extraction systems we will first discuss the basic steps required in Temporal Information Retrieval which are as follows as described in [1]:

1. Tokenization
2. Sentence Extraction
3. Part of Speech (POS) Tagging
4. Named Entity Extraction

This format was the traditional method of performing information extraction, with the named entity extractor also detecting temporal phrases. However the process of extracting temporal phrases has been separated in most state of the art system. With the Temporal Information Extraction process involving three steps:

1. Recognition of Temporal expressions
2. Normalisation
3. Temporal Annotation

The first step is responsible for identifying the temporal expressions discussed above. Normalisation is the process of creating a common representation for all the different types of temporal expressions. Finally the document is annotated with some form of mark-up usually TimeML³. The entire process from tokenization to annotation can be depicted in the Figure 2.

A driving force behind advancements in the field of temporal information extraction are the TempEval sub-tasks in the International Workshop on Semantic Evaluation⁴. This section shall compare some of the systems produced for TempEval-3⁵.

TempEval-3 was the first time that the workshop focused on end-to-end temporal relation classification [8]. Nine participants submitted in the workshop however here only three will be discussed, the two top performing systems HeidelTime [9] and NavyTime [10] both of which are rule based systems, in contrast we will also be discussing ClearTK-TimeML [11] a minimalist approach which made use of machine learning compared to the more traditional rule based systems.

³<http://www.timeml.org/>

⁴<http://alt.qcri.org/semeval2015/>

⁵<https://www.cs.york.ac.uk/semeval-2013/task1/>

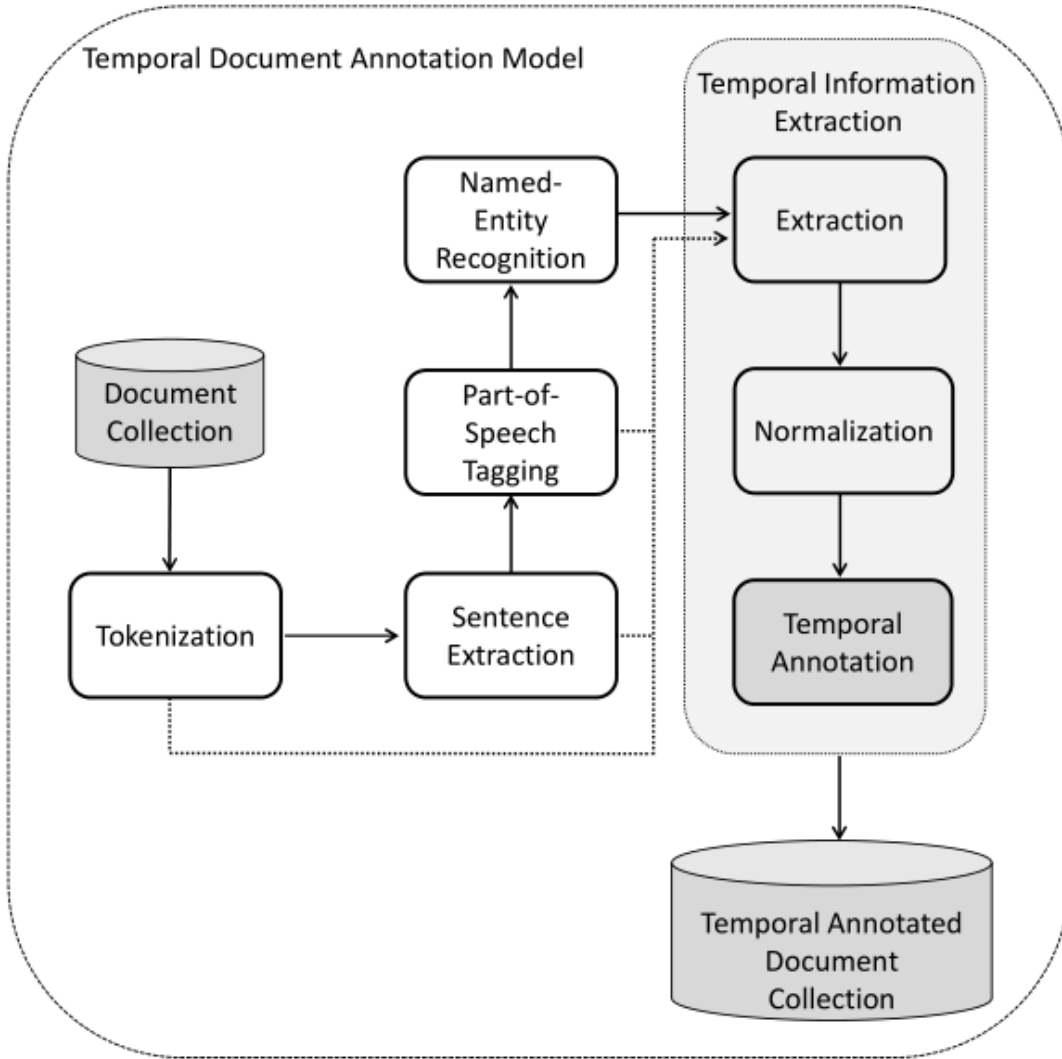


Figure 2: Temporal document annotation model [1].

All three systems performed preprocessing on the raw text before classification of temporal events, with all three performing tokenization and part of speech tagging, however NavyTime and CLeaRTK required other extra inputs. NavyTime also made use of 2 and 3 n-gram tokens as well as the WordNet lemma for those tokens and a binary value which was true if the lemma of the token existed in the WordNet event synset amongst others. The preprocessing for CLeaRTK was slightly different than the other two systems - it created two feature sets, one was passed to a time model and the other was passed to a event model, the first being used to extract temporal expressions and the latter being used to extract the events form the text [9, 10, 11].

Once preprocessing has been carried out the systems proceed to extract temporal relations. ClearTK uses a set of time models and event models to extract time expressions, which are normalised by a third party system. These times and events are then passed to a set of three temporal relation models which then classify the relation followed by some post-processing and annotation of the text. This strategy resulted in statistical improvements in both precision and recall. However the addition of a larger set of relations during the training process resulted in a trade-off between recall and precision [8, 11]. The use of machine learning systems greatly simplifies the problem of generating a temporal extractor however it requires training sets of annotated text for the specific domain.

In contrast to ClearTK and HeidelTime implements rule based systems however they do so in different ways. HeidelTime makes use of a resource interpreter and resources which contain patterns, normalisations and rules the interpreter then applies the resources to the text to annotate it. This has the added benefit of making it somewhat easier to develop for different languages. Compared to the other two, HeidelTime was the only system to also be evaluated against Spanish text during TempEval. During evaluation the system seem to suffer from many false negative results, lowering its recall. This was partly caused by the fact that the system ignored vague time expressions since they are rather difficult to normalise. However the system did perform best overall [8, 9]. Possibly one of the more attractive features of this system is that of its modularity. The ability to swap out resources means that the system will be able to cope with different languages and different domains, as long as the resource is available. Perhaps the next step for a system such as this would be the introduction of learning algorithms to reduce the time required to generate specific resources.

NavyTime’s novel aspects are the use of a rich set of contextual features, dividing the problem of event and temporal extraction and the use of a larger number of classifiers which they have shown yields better results [10]. This approach of many smaller classifiers which are more specialised and together cover a wider range of possibilities seems ideal for a long term project in which over time specific classifiers can be built to cover specific subsets of possible temporal relations.

Yet another system to come out of SemEval, in this case SemEval 2015 task 4⁶, is closely related to the work at hand. The task was to perform temporal relation extraction on a set of documents and extract the relations which relate to a particular entity finally sorting those discovered relations in a timeline [12]. The submission to this problem by

⁶<http://alt.qcri.org/semeval2015/task4/>

Laparra et al. [13] tackles this problem by mainly targeting time-anchors in the text. The system worked by performing temporal extraction and reasoning and entity extraction on the texts. It did this through the use of the Stanford Core NLP⁷ and a rule based system. It should also be noted that it made use of entities found in the DBpedia knowledge base to help disambiguate named entities. The system targeted mainly explicit time expressions which could easily be plotted on a timeline without ambiguity, while the system scored very highly overall the work found that within the domain of news articles many implicit temporal expressions were missed. While this is a genuine concern within the domain of news we will have to see if the same holds true for biographical texts which will be used within the work presented in this document.

Another system which worked in the domain of news, EXPOSÉ [14], is an exploratory search system for news articles and makes use of two sources, a collection of news articles and Wikipedia as orthogonal sources. It makes use of a combination of techniques within a Temporal Retrieval Engine. For each search it considered the text, the publication date temporal expressions extracted from the text and combinations of the publication date and temporal expressions to resolve implicit expression. The final results of the system were plotted in a timeline which depicted references to relevant entities in news articles over time.

The final system we will discuss in this section is rather different to the ones which have previously been discussed, since it does not follow the standard approaches as described above, but rather uses social media, in this case Twitter⁸, and ‘personal important events’ of individuals to generate a timeline of events in the user’s life [15]. Rather than focusing on extracting temporal expressions within the tweets the system uses a Dirichlet Process model to evaluate the tweet. The time the tweet was published is taken as the temporal information which is related to the text. While this work does not hold a lot of relevance to solving the problems at hand it is interesting to see a similar end result using a completely different approach.

2.2.2 Visualisation of Temporal and Structured Information

This section will describe a system which generated timelines through the use of structured data only however first we will begin by discussing about some different tools and methods for visualisation.

⁷<http://stanfordnlp.github.io/CoreNLP/>

⁸<https://twitter.com/>

Most research describe and implement timelines as 2-dimensional textual or graphical representations of some set of linear events in a given order [16, 17]. A note worth system is the Lifelines system which displays medical and criminal information about a particular person while Lifelines provides a rich visualisation of the data it is very domain specific [18]. Another tool which was evaluated during the research phase of this project is SIMILE (Semantic Interoperability of Metadata and Information in unLike Environments) Widgets⁹ which is a tool which helps with access, management and visualisation of structured data including temporal data and even has a timeline module [19]. One of the more aesthetically pleasing and easy to use visualisations is a contribution by Northwestern University called Timeline JS¹⁰, which is an open source java script application that visualises JSON data. Due to the ease of use of this software and the fact that it is rather aesthetically pleasing this visualisation software will also be used to present the end product produced by the PoI system.

Finally in this section we shall discuss TimeMachine, a system presented by Althoff et al. [20] this system generates timelines for events and relations that relate to a particular entity which exists in some knowledge base. The system makes use of the Freebase knowledge base, which has since been closed and the data migrated to WikiData¹¹. This work has a promising implication for the PoI system, since it generated timelines entirely with the use of events and relations from a knowledge base it supports the assumption that knowledge bases will contain valuable information which can be used to help in the temporal information extraction process.

⁹<http://simile-widgets.org/>

¹⁰<http://timeline.knightlab.com/>

¹¹https://www.wikidata.org/wiki/Wikidata:WikiProject_Freebase

3 Design

In this section we will be breaking down the requirements and functionality of the PoI system. Starting with a high level perspective of the system in Section 3.1, then going into further detail about all the components in Sections 3.2, 3.3 and 3.4.

Before discussing the design of the system we shall briefly discuss the technologies used in the building of the system. The PoI system was built using the latest version of Python (Python 3) and makes use of a number of packages available. Most notably it makes use of the Natural Language Toolkit (NLTK) which is used both in the Text Processing Module and the Event Extraction Module. The Text Processing Module also makes use of the Stanford Named Entity Recogniser, through a wrapper made available in NLTK. Other packages used are discussed later when relevant. Visualisation of the timeline produced is carried out through the Timeline JS JavaScript application.

3.1 High Level System Architecture

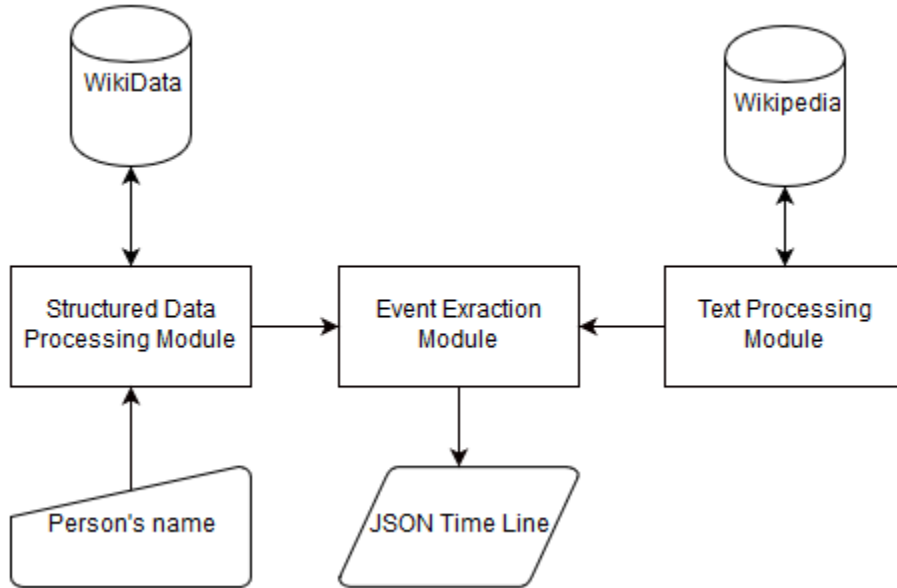


Figure 3: High level architecture of PoI system

The only input that the system will require is the name of a well known person. It

is important that the person in question be well known otherwise there may not be any information available both structured and unstructured.

Once the system has finished processing it will output a JSON and HTML file. The JSON file will contain the data required to populate the timeline and the HTML file will contain the code necessary to call the Timeline JS application to display the timeline. The overall system can be seen depicted in Figure 3.

The Structured Data Module (discussed further in Section 3.2) is responsible for obtaining the structured data from WikiData. It will also filter out irrelevant statements and index entities, properties and other information obtained from WikiData.

The Text Processing module (discussed further in Section 3.3) also only requires the name of the person of interest. On completion of its task, this module will output the results of processing text obtained from Wikipedia. It accomplishes this by first downloading the relevant Wikipedia page, extract the text and processing it.

The Event Extraction Module (discussed further in Section 3.4) will require the outputs of the two aforementioned modules. This module will analyse the text, identifying those phrases/sentences which contain date references. It will determine which of those phrases relate to the person of interest and if so it will index the phrases. Once it has completed analysing the text it will then compile the indexed information into a JSON file and create an HTML document from which the timeline can be viewed.

3.2 Structured Data Processing Module

This module solves the first of the objectives listed in Section 1.3. It is the first subsystem which is triggered and is responsible for collecting, resolving and indexing data obtained from WikiData.

The data collected is indexed in two ways: by property-value pair and by time. Those statements collected from WikiData with temporal qualifiers are indexed according to their date allowing them to be quickly compared with dates found in the text.

Inspiration for this portion of the system was drawn from the works of Althoff et al. [20] and Laparra et al. [13]. The former acted as evidence that structured information can be used to generate good timelines. It also aided in the decision of which knowledge base to use. Since it found good results with Freebase, which no longer exists but whose data has been moved to WikiData, we chose to use WikiData as the source for structured information. The work of Laparra et al. provided evidence that structured information can be used to facilitate the information retrieval process.

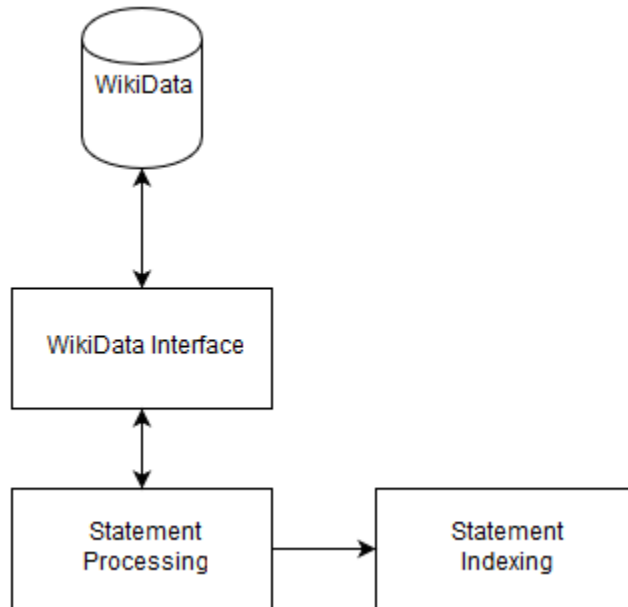


Figure 4: Design for the Structured Data Processing Module

An outline of the basic system can be seen in Figure 4. The WikiData interface is responsible for communication between the system and the WikiData servers. The statement processing component will process all the statements obtained from WikiData and will resolve the WikiData items into their labels. Requesting additional information through the WikiData Interface when needed. Statements that have been resolved will then be indexed as described above.

3.3 Text Processing Module

This module meets the requirements of second objective detailed in Section 1.3. The goal of this module is to collect data from Wikipedia and complete the tasks required for preprocessing. The techniques used in this process have been inspired by the preprocessing carried out in the works presented for the TempEval tasks [8], described in Section 2.2.1. For preprocessing to be successful the following tasks need to be carried out (The entire process is depicted in Figure 5):

1. Downloading the correct Wikipedia page
2. Scraping the text from the page

3. Paragraph and Sentence Extraction
4. Word Tokenization
5. Part of Speech Tagging
6. Named Entity Recognition

Using the target's name the system will first download the corresponding Wikipedia page as an HTML file. It will then scrape the summary and all relevant sections from the page, excluding section like 'References', 'See also' and 'Other' which are unlikely to contain pertinent information. The scraping process will also maintain the structure of the sections holding each section as a pair of section heading and section text. Each sentence in text and each heading will then be tokenized again turning them into a list of words.

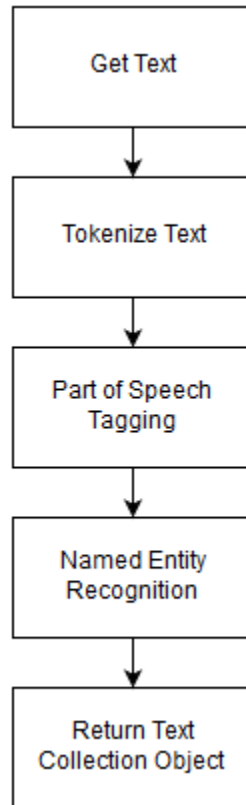


Figure 5: Design of the Text Processing Module

Once tokenization is complete Named Entity Recognition is then performed on the text.

To perform the Named Entity Recognition, the Stanford Named Entity Recogniser¹² (NER) will be used. It is important to note that Stanford NER can tag temporal expressions. While it tags explicit temporal relations well, it struggles with tagging vague and referential temporal expressions. This design choice is in contrast to most of the examples of systems submitted for TempEval. The reason why this choice was made is due to the fact that the end result needs to be a human readable timeline rather than a TimeML annotated text as required in TempEval. Hence we do not need to perform level of temporal tagging required in TempEval and the date tag provided by the Stanford NER is sufficient.

The final step of preprocessing is to compile the results from the previous steps into a Text Collection object. This is a simple collection object which will be used by the next phase to iterate through the text.

3.4 Event Extraction Module

The Event Extraction Module is responsible for tackling the remaining four objectives listed in Section 1.3 which are as follows:

1. Extracting dates from phrases.
2. Determining if a phrase is relevant to the targeted person.
3. Using structured data to aid in identifying relevant phrases.
4. Generation of the timeline.

To achieve these goals the module processes each sentence in the text by first recoding entities (people and organisation) that appear in the sentence. These will then be used to resolve references in subsequent sentences. It then proceeds to extract and normalise any dates from the sentence. Once date extraction is complete the sentence will fall into one of three categories:

1. No date phrase - ‘Mandela visited France’.
2. Single date phrase - ‘Mandela visited France in 1999’.
3. Multi-date phrase - ‘Mandela visited France in 1999, and again in 2005’ .

¹²<http://nlp.stanford.edu/software/CRF-NER.shtml>

The first case is discarded since the event cannot be plotted in a timeline. Single date phrases will be compared to known events from structured information and if a match is found the phrase will be indexed. If no match is found the system will then try to determine if the phrase contains an event which relates to the person of interest. For the multi-date case the system attempts to chunk the phrase into several date phrase. Each of which are bound to one of the extracted dates and treated as single date phrase. Once all the text has been processed the indexed events are compiled into a timeline. This is done by generating a JSON file which represents all the events and by generating an HTML document. When the HTML document is viewed by through a web browser it will call the Timeline JS code to display the information stored in the JSON file. The complete process is depicted in Figure 6.

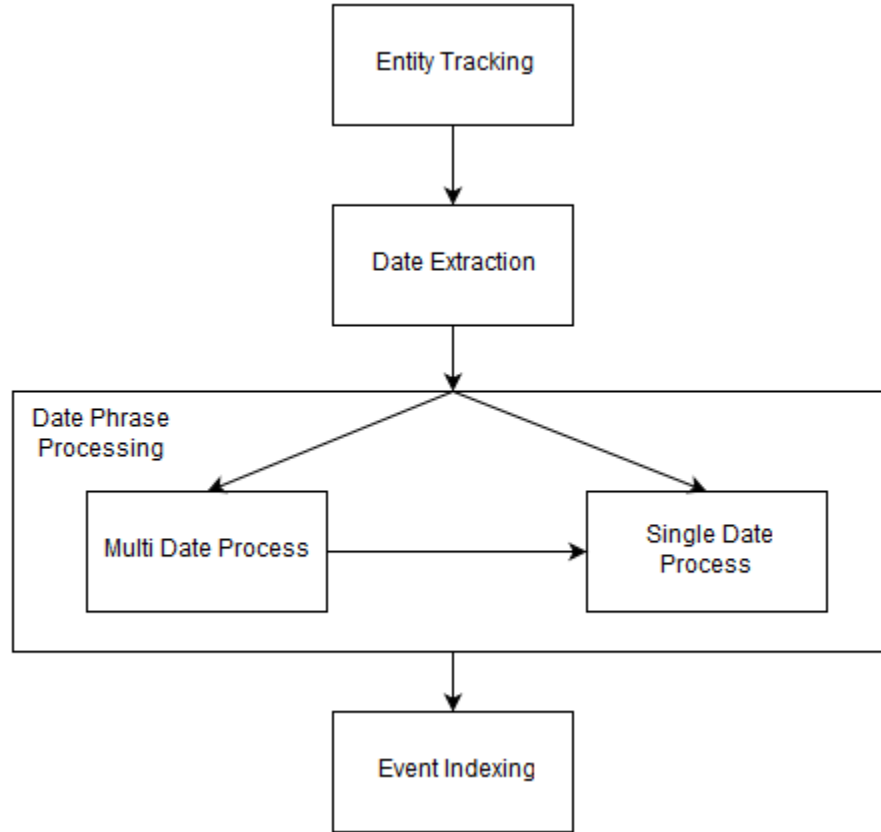


Figure 6: Steps in the Event Extraction Module

When designing and building this system to solve these objectives the works of Mishra and Berberich [14] and Minard et al. [12] were very influential. From the former we have

the ideas of multiple indexes and multiple methods of retrieving temporal information. While the work of Minard et al. shows a method of how entities may be used to help temporal relation. One should also note that in our case there is no need to extract the events completely from the text. It is more important to ensure that they are present in a phrase since the extracted phrases are intended to be human readable.

Entity Tracking

This component of the system acts as a memory for the entities such as people and organisations which are mentioned in sentences. It stores them in order of appearance and uses them determine if a pronoun refers to the person of interest.

Date Extraction

The date extraction process is responsible for meeting the objective of date extraction. This is a very important step in our system since if dates cannot be extracted correctly the related events will be disregarded. The system uses the tags provided by the NER process in combination with a collection of rules to process the phrase and extract a normalised representation of the date. This module also needs to take care of the extraction of different types of dates which we categorise into two cases: points and ranges. A point being a date that refers to a single event that happens once such as ‘3 June 1994’, whereas a range defines a pair of dates that define an event that continues to happen for some period of time such as ‘between 2003 and 2004’. The date extraction process will also ensure to correctly classify lists of dates such as ‘in 1999, 2000 and 2001’ as individual date points.

The Date extraction process also keeps a memory of last referenced date to be able to resolve dates that appear in the text without a year such as ‘in April’. Here it is assumed that the year was mentioned in the previous sentence and that the month that appeared in the sentence after refers to that same year.

Another notable feature is that it uses information obtained from the structured data. If available it will utilise the date of birth and death of the person of interest to prevent the extraction of dates outside the person’s life.

Date Phrase Extraction

Date phrase extraction is the process of extracting a phrase which is related to a particular date that has already been extracted from the text.

There are two cases for date phrase extraction. The first is the case that in a phrase there is only one date. The second is that of multiple dates occurring in a phrase. It should be noted that date ranges are considered to be a single date.

Multi-date phrases are first converted to single date phrase, each containing a single date. In order to extract the single date phrases from a sentence, it is chunked by using NLTK Regex Parser and a custom hand built grammar. The goal of the parser is to build trees which divide the sentence up into 'date clauses' which represent some possible event appearing next to a date. Each of these date clauses are compared and bound to the dates which have been previously extracted. Each of the date clauses which were successfully bound to a date are then treated as single date clauses.

During single date phrase processing there are two attempts made to resolve the date phrase. The first uses structured information while the second attempts to see if the phrase references the person of interest and any other entities.

When trying to resolve the phrase by structured information the system will retrieve all statements which are indexed to the same date. It will also retrieve those statements which partially match the date. For example if the date in the text is normalised to 2003 it will retrieve all the statements which occurred in 2003 irrelevant of their month or day. The system does this to deal with the problem of precision. The date mentioned in the text may be less precise than the date obtained from the structured data. If a match is found the entities in the statement are compared with phrase. If they are found to be similar the phrase is then considered to match to the statement and it is indexed as an event.

If no match was found, the system will attempt to determine if the phrase is related to the person of interest. If the phrase does reference the person of interest and the phrase contains at least one other entity the phrase is considered to be an event relating to the targeted person and indexed.

Timeline Generation

The final step that the system takes is to generate and compile the information into a JSON file that can be rendered by the Timeline JS software. The timeline will consist of slides with each slide representing either a point in time or a period in time. On each slide the system will include all the events it extracted for that specific moment in time.

4 Methodology

In the previous Section 3 we looked at the proposed design for PoI system. In this section we discuss how the system was implemented. We start by describing how the structured and unstructured data was collected and prepared in Sections 4.1 and 4.2. We will then proceed to describe how the dates and events are extracted as date phrases and subsequently indexed, in Section 4.3.

4.1 Structured Data Processing Module

In order to access WikiData information a wrapper was built which wraps the MediaWiki API¹³. This wrapper handled sending and receiving requests to the API. A python script was then developed which used the wrapper to get all statements for an individual. However since all statements are returned as a JSON object containing WikiData item ids, the script then resolves each of those ids into a English language label. A problem was encountered at this stage since in some cases a WikiData item would not have an English label, in these cases the item was ignored. It should be noted that to resolve each item into a label required sending a request to the WikiData server.

Along with this wrapper and script two indexing classes were also implemented. The first handled indexing all the WikiData statements and their qualifiers along with the labels for the WikiData items in the statements and qualifiers. The second was used to index those statements which had time qualifiers.

Once all the data has been indexed it is stored for use by the Event Extraction Module.

4.2 Text Processing Module

This module is responsible for obtaining the HTML page related to a targeted person, scraping the text, cleaning the text and performing all necessary preprocessing on the text as is shown in Figure 7.

Unlike in the previous case a wrapper already existed which wrapped the WikiMedia functionality responsible for accessing Wikipedia pages. This Wikipedia¹⁴ wrapper was used to find and download the HTML of a targeted individual.

¹³https://www.mediawiki.org/wiki/API:Main_page

¹⁴<https://wikipedia.readthedocs.io/en/latest/>

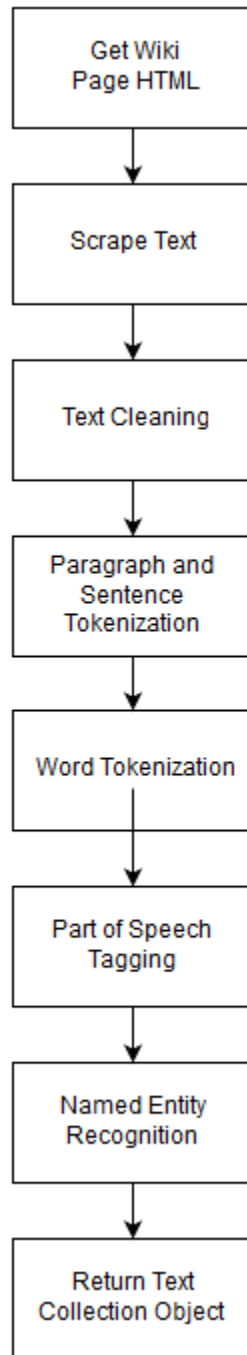


Figure 7: Steps in the Text Processing Module

The text was then scraped from the HTML. The scraping is performed by a Python

script that makes use of the BeautifulSoup¹⁵ package to generate a DOM tree. The scraper then navigates to relevant sections and retrieves the text ignoring sections such as ‘external links’, ‘See also’, ‘Notes’ and ‘Further reading’ to mention a few. This is because on examining sections it was determined that relevant dates are unlikely to occur in them and they are unlikely to be valuable to the timeline building process.

After scraping the text a few problems were detected. One of the main problems was the presence of non utf-8 characters such as ‘ä’. These cause problems for the NER process as discussed below. Another problem was caused by the square bracket citations such as ‘[99]’ throughout the text. The first problem was solved by using a Unicode decoder¹⁶ to ensure that all the text is in the correct encoding for the NER. The problem of citations was resolved with the help of a regular expression that was used to find and remove all citations. These solutions along with some others which resolved other minor problems were then packaged up into a cleaner that was run on all the text.

Another problem which was tackled by the text cleaner is that of normalising some dates, specifically the case of years which represent a range such as ‘1994-99’ and ‘1994/99’. We chose to tackle this problem here since these cases can easily be detected with regular expression and converted to a standard form such as ‘1994 to 1999’.

Once the text has been extracted and cleaned it is then tokenized first by paragraph, then sentence and finally by word. Tokenization is carried out using the NLTK tokenize package¹⁷.

The final steps of this module are to perform POS tagging and NER. This is accomplished by using the NLTK POS tagger¹⁸ and the Stanford NER through a NLTK wrapper¹⁹. Once the text has been tagged it is packaged up in a text collection object and stored for use by the Event Extraction Module. The Stanford NER is initialised before use with a 7 class model which tags: Location, Person, Organisation, Money, Percent, Date and Time. This model was chosen since it can tag dates.

4.3 Event Extraction Module

Once the structured and unstructured data has been prepared and stored, the Event Extraction Module can start to analyse the text and extract date phrases. This system

¹⁵<https://www.crummy.com/software/BeautifulSoup/>

¹⁶<https://pypi.python.org/pypi/Unidecode/0.04.1>

¹⁷<http://www.nltk.org/api/nltk.tokenize.html>

¹⁸http://www.nltk.org/_modules/nltk/tag.html

¹⁹http://www.nltk.org/_modules/nltk/tag/stanford.html

has three separate phases: of Initialization, Text Processing and Timeline Generation as depicted in Figure 8.

The Initialisation process is responsible for loading the data prepared by the previous two modules and initialising all the objects which will be used in text processing. The text processing component analyses the text and extracts and indexes date phrases (this component is discussed in further detail below). The final component of Timeline generation builds all the necessary files required to display the timeline performing some post-processing to improve the quality of the data.

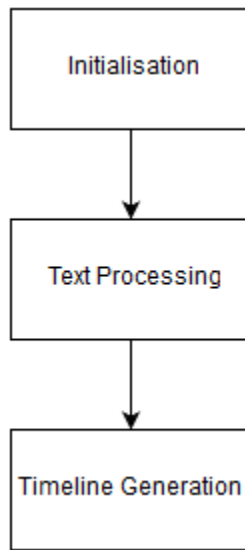


Figure 8: Three phases of Event Extraction

4.3.1 Text Processing

This component handles the extraction of dates and date phrases from the text by sequentially analysing the sentences in the text. An overview of how it works can be seen in Figure 9. Each sentence in the text is passed through this component.

The first step is entity tracking where the system extracts entities that have been tagged by the NER. It has two separate memories one for people referenced in the text and one for other entities such as organisation or locations. These memories are then used later during the single date processing. Since this system relies on the NER to correctly tag the entities it can fail to assign entities to the correct memory if they are incorrectly tagged or not assign them at all.

The date extraction process then strips out and normalises the dates found in the sentence (we go into further detail about how this works in Section 4.3.2). The result from the date extraction process is a list of dates. If this list contains only one date, the date and the phrase are passed to the single date process (refer to Section 4.3.3). If there are multiple dates they are passed thorough the multi-date process (refer to Section 4.3.4). If no dates are found the system will then proceed to the next sentence.

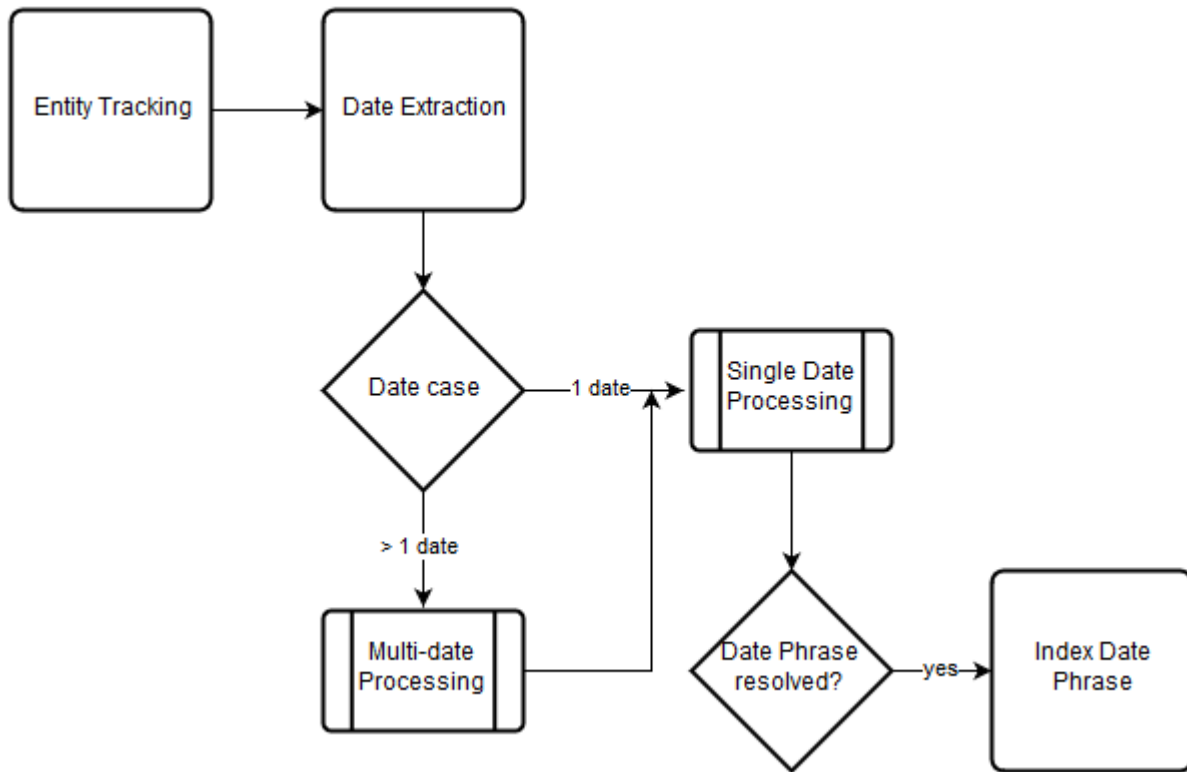


Figure 9: Overview of Text Processing

4.3.2 Date Extraction

The date extraction process is given a phrase which has been tokenized and tagged. If there is at least one token in the phrase which has been tagged as a date it will attempt to extract all dates in the phrase.

The process starts by collecting the indexes of the tokens tagged as dates and assigning them into groups of neighbours, which are in turn processed individually. A date compo-

ment type (day month or year) is assigned to each member of a group. This is done by using a rule system which compares and tests the value of the item to determine its type.

Once all the components of a group have been classified then they are added to a date holder object - a custom object used to store and manage dates and their properties. In implementing this system we encountered the problem of dates missing the year component. While examining this problem we noted that in most cases the year would appear in the previous phrase. To fix this we keep the previously extracted date in memory and assigned its year value when a date was found with a missing year.

After a date has been found the following word is tested to determine if it indicates a date range. This determination is performed by comparing the expression against a list of known words which indicate date ranges. If the date is followed by a range word and another date then two dates are extracted together as a date range. A special case had to be developed for dates separated by the word ‘and’, since it indicates one of two possibilities either a range or the end of a date list. To determine which case is true we examine what precedes the first date. If it is preceded by a word which indicates range such as ‘between’ it is treated as a range, otherwise they are treated as separate dates.

Once all the date groups have been processed, the dates which have been extracted filtered. This process removes any dates which are out of range, before the person’s birth or after their death. It also attempts to flag any irrelevant dates such as those which appear in quotations. This final list is then returned to the event extraction process. Note that at this point the list may be empty.

4.3.3 Single Date Processing

This component evaluates a date which has been extracted from a phrase and will determine if that phrase should be included in the timeline. An overview of how it does this can be seen in Figure 10.

The first attempt that the system makes to determine if a phrase should be indexed is to compare the date that has been extracted with those that are present in the structured data. If no exact match is found then it will decrease the precision of the date and try again. Assuming a match is found the WikiData statements associated with that match are retrieved.

In the case that a match is found the system checks to see if the entities described in the statements against the phrase. This comparison is done with the use of a FuzzyWuzzy python package, which implements fuzzy string matching. Using this package we assign a

score to each statement. If the highest scoring statement has a score over a set threshold it is considered a match with the phrase. A threshold of 85 out of a possible 100 was selected after a process of trial and error.

If a statement is matched to the phrase, the date associated with the statement and the date extracted from the phrase are compared. The one with the highest degree of precision is accepted and used to index the phrase.

In the case that no match could be found between the WikiData statements and the phrase the system attempts to determine if the phrase corresponds to some event related to the person of interest. If the phrase does reference the person of interest the system will then attempt to determine if the phrase refers to an event

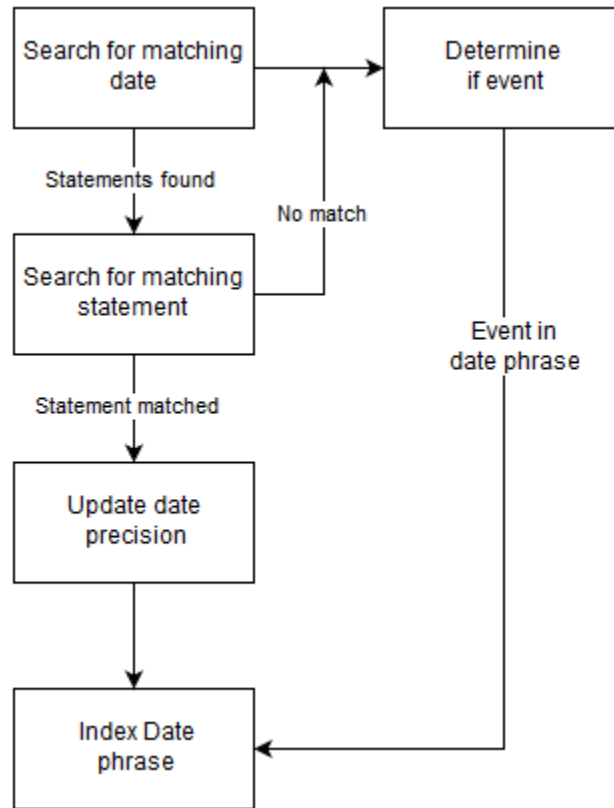


Figure 10: Overview of Single Date Processing

This reference to the person can be either explicit (person referred to by name) or implicit (referred to with a pronoun). To resolve the first case we simply see if the person's name or a portion of it appears in the phrase. For the second case we first resolve the

pronoun by referencing to the entity memory and checking if the last person mentioned was the person of interest. A limitation of this system is that it can confuse people with the same name such as in the case of these two names: ‘Barack Obama Sr.’ and ‘Barack Obama’. Assuming the targeted person is referenced in the phrase the system will then check for references to other entities related to the person of interest. The phrase will then be indexed if both of these conditions have been met.

4.3.4 Multi-Date Processing

The aim of the multi-date process is to divide the phrase into several phrases each of which represents a single date event. This process is broken down into five main steps as shown in Figure 11.

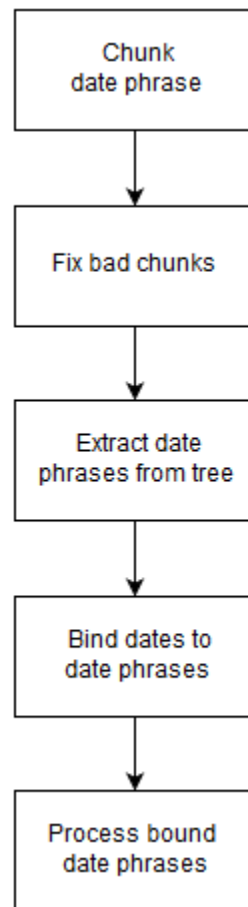


Figure 11: Overview of Multi-Date Processing

The first step generates a tree that represents the phrase. This is accomplished through the use of an NLTK Regex Parser and a custom made grammar. The grammar was developed through a process of mainly trial and error. The grammar will attempt to chunk the text into clauses and date clauses. Once the tree has been generated it is checked for any bad chunks and system will attempt to fix them.

The next step is to then identify all those clauses containing dates and extract them as separate phrases. Problems were encountered here with clauses that have no subject like in the case of ‘helped found the National League for Democracy on 27 September 1988’. For these cases the system will attempt to look at the previous clause chunk and extract the correct subject from there and apply it to the beginning of the phrase. However given the complexity of language this method is not very reliable.

The final steps in this process are that of associating the dates which were extracted to the extracted date phrases and process each of the date phrases which are bound to a date like the single date cases as discussed above.

4.3.5 Timeline Generation

Once all the text has been processed and the events indexed the timeline is compiled. This is done by iterating through the indexed events and generating the appropriate JSON object for each of them. Once the JSON objects are created they are serialised and saved as a file.

The final step is the generation of an HTML document to display the time line. To achieve this a template file is edited so that it will load the appropriate JSON file and pass it to the Timeline JS app which is called in the HTML document. Figure 12 shows a screen shot of a timeline being viewed through a browser.



Figure 12: A Generated Timeline

5 Evaluation and Testing

In this Section we will discuss the testing and evaluation carried out on our system. We attempted to evaluate all of the individual components involved in meeting the objectives described in Section 1.3. We first start by discussing the evaluation of similar systems, followed by a descriptions of what could be considered the ideal strategy for evaluating the system. Then we will propose a series of experiments to evaluate the system, discussing how these were carried out and analysing the results. Finally we present a Section which describes how the system was tested during implementation as well as its performance.

5.1 Similar Systems Evaluation

In this Section we will be looking at the general methods of evaluation used in the TempEval. [8] gives an overview of all the evaluation carried out during TempEval-3. The systems which took part in TempEval-3 were evaluated against several annotated corpora. Existing corpora of annotated text included examples such as TimeBank²⁰, which were also improved upon before use. In addition to already existing corpora they produced new corpora such as the TempEval-3 platinum evaluation corpus, which was manually annotated by the organisers of the workshop.

The first method of evaluation we shall discuss is the evaluation of temporal expression extraction. Precision, recall and f-measure were used as metrics for evaluation. In addition to this there were two modes of evaluation which were carried out, strict and lenient. In the case of strict temporal expression were considered to be extracted correctly if there was an exact match between the extracted text and the annotated example. In the lenient case partial matches were also accepted. The system's ability to normalise dates was also evaluated. In this case the accuracy of the normalised date when compared to the annotated example was used as a metric.

5.2 Ideal Evaluation Plan

The ideal evaluation plan should consist of a rigorous evaluation of the system in its entirety including the final output. This plan should evaluate the following:

²⁰<https://catalog ldc.upenn.edu/LDC2006T08>

1. The dates extracted.
2. The extraction of multi-date phrases.
3. The indexing of relevant events.

As far as date extraction is concerned, the ideal evaluation case would require a corpora of annotated Wikipedia pages about people where all the dates have been annotated. This annotation would need to be done by experts. The dates extracted from the system could then be compared with the annotated texts and precision recall and f-measure would be calculated as evaluation metrics.

To best evaluate the extraction of multi-date phrases a corpus would need to be compiled with all examples of multi-date phrases and the correctly extracted sub phrases. From this we would then be able to compare the systems results and calculate precision, recall and f-measure.

The final evaluation process would best be performed with the use of gold standard timeline that have been compiled by experts using the same data source (Wikipedia). This timeline would then be used to compare the one generated by the system and we would be able to calculate precision, recall and f-measure metrics.

5.3 Experiments

In this Section we present the experiments carried out on the system as well as the results obtained from those experiments. This Section also includes an analysis of the results, starting with a description of the data used to run the experiments and then going into detail about each of the three experiments carried out.

Due to time restriction and resource limitation the ideal data sources for evaluation (described in the previous Section) could not be built. Instead we chose the following six individuals:

1. Nelson Mandela
2. Barack Obama
3. Aung San Suu Kyi
4. Enoch Powell
5. Eddie Fenech Adami

6. Dom Mintoff

Eddie Fenech Adami and Dom Mintof were chosen as local candidates whose international recognition is far less than the other and also have less information available on their wiki page. The remainder of the individuals were chosen since they have a large volume of information available on their wiki pages. Note that we also selected these individuals from different cultures to try and provide as much diversity as possible while ensuring that all the individuals are politicians and public figures.

5.3.1 Date Extraction Evaluation

The goal of this experiment was to evaluate the extraction of dates from the text. In this case we are not concerned whether a date is relevant to the timeline or not since that decision comes after the date has been extracted.

We started by manually creating a list of all the dates for each sentence of the texts to represent the ideal results. Once this had been done we then passed the texts through the system logging every date extracted. We then proceeded to calculate precision recall and f-measure. In this case precision is the fraction of correctly extracted dates. Recall is the the fraction of correct dates which were extracted. Both precision and recall were calculated with the equations below.

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{no. of correctly extracted dates}}{\text{no. of all dates extracted}}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{no. of correctly extracted dates}}{\text{no. of all correct dates}}$$

F-measure is the harmonic mean of precision and recall calculated by the equation below.

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Table 1 shows the results of this experiment. We note that the date extractor scored very highly. We believe that the high result are due to the limited variety in the way dates are described in documents of this nature, adding weight to the assumption made in Section 2 about the importance of Explicit Dates. However when examining the data we also noted several reoccurring errors.

One such error is due to an oversight where the system is unable to correctly extract dates where the month appears before the day, such as ‘May 2 1950’. A by-product of this

Person	Precision	Recall	F-measure
Nelson Mandela	0.94	0.96	0.95
Barack Obama	0.8	0.96	0.87
Aung San Suu Kyi	0.99	0.99	0.99
Enoch Powell	0.97	0.89	0.93
Eddie Fenech Adami	0.98	0.94	0.96
Dom Mintoff	1	0.92	0.96
Average	0.93	0.94	0.943

Table 1: Date extraction evaluation results.

error is that the system mistakenly extracted multiple dates. Extracting the month as one date and the day and year as another. For example ‘May 2 1995’ would be extracted as February 1995 and May, an incorrect year would also be resolve for May by looking back at the previous year in memory.

The second common type of error observed was when dates appeared in the forms such as ‘mid-nineties’, in these cases the system was unable to resolve them. A final category of errors were those of implicit dates and vague dates. Most of these cases were found not to be tagged by the NER and hence would not have attempted to resolve them.

5.3.2 Multi-Date Phrase Extraction Evaluation

The goal of this experiment was to evaluate the extraction sub-date phrased from multi-date phrases. This test will evaluate the chunking and date binding process.

We started by manually creating a document with all the multi date phrases found in the targeted individuals text. For each of those phrases we then created ideal sub-phrases. Once this answer set had been made we ran the system for the target individuals logging all the output of the multi-date processing component. The output was then compared with the answer set and used to calculate precision, recall and f-measure. In this case f-measure is calculated as in the first experiment. However precision is the fraction of correctly extracted phrases and recall is the fraction of correct phrases which were extracted.

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{no. of correctly extracted phrases}}{\text{no. of all phrases extracted}}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{no. of correctly extracted phrases}}{\text{no. of all correct phrases}}$$

Person	Precision	Recall	F-measure
Nelson Mandela	0.88	0.86	0.87
Barack Obama	0.48	0.45	0.46
Aung San Suu Kyi	0.68	0.5	0.58
Enoch Powell	0.69	0.61	0.65
Eddie Fenech Adami	1	1	1
Dom Mintoff	0.5	0.75	0.6
Average	0.71	0.7	0.74

Table 2: Multi-Date phrase extraction evaluation results.

From the results, shown in Table 2 we can see that the performance of this varied from greatly. With the lowest f-measure being that of 0.46 for the text regarding Obama and the highest score of 1 for the text regarding Fenech Adami. With regards to Fenech Adami, we believe that the low number of multi-date phrase examples in that text greatly contributed to the perfect score. However that said the system also scored well on the text of Mandela which had many examples of multi-date phrases.

After analysing the results we also noted some contributing errors. One of the more prominent errors was a result of the badly extracted dates which were in the format ‘month, day, year’. In this case multiple dates were erroneously extracted for a single date and in most cases resulting in a single date phrase being passed to the chunker. The chunker was not designed with these kind of phrases in mind and as a result chunks them incorrectly. The error continues to propagate when the incorrect date is bound to a broken phrase.

Another common error found was missing components of the phrase such as ‘Obama announced on November 13, 2008’.

5.3.3 Evaluation of Timeline Events

The goal of this experiment was to evaluate how the system extracts relevant events to compile the timeline. This experiment will evaluate the event extraction module.

We started by manually creating a standard timeline for each targeted person. The timelines consisted all events related to the target person that could be extracted from

the text. Once timelines were completed we ran the system for the target individuals logging all the events which were included in the final timeline. The output was then compared with the answer set and used to calculate precision, recall and f-measure. In this case f-measure is calculated as in the first experiment. However precision is the fraction of correctly extracted timeline events and recall is the fraction of correct timeline events which were extracted.

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{no. of correctly extracted timeline events}}{\text{no. of all timeline events extracted}}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{no. of correctly extracted timeline events}}{\text{no. of all correct timeline events}}$$

Person	Precision	Recall	F-measure
Nelson Mandela	0.94	0.96	0.95
Barack Obama	0.77	0.88	0.82
Aung San Suu Kyi	0.9	0.78	0.84
Enoch Powell	0.8	0.96	0.87
Eddie Fenech Adami	0.95	0.8	0.87
Dom Mintoff	0.94	0.58	0.85
Average	0.87	0.82	0.84

Table 3: Timeline Events evaluation results.

Table 3 shows the results from the evaluation of the phrases extracted for inclusion in the timelines. Overall these results are rather good. However it should be noted that if an event appeared multiple times in the text the system would have also extracted it multiple times. This is a factor which contributed to the timelines appearing crowded. It should also be noted that when comparing the extracted phrases to the generated timelines partial matches were accepted. However each of the partial matches were then inspected manually to determine if enough of the phrase was extracted for it to make sense to a human reader. In the cases that it did not the extracted phrase was then counted as a false positive.

One of the main reoccurring errors noticed when evaluating these results was that the system was at time unable to distinguish between two individuals with similar names. An example of this is Obama and his father who share the same name and is a contributing factor to the lower than average precision score in his case. Other common errors were errors which occurred due to errors in previous components.

5.4 Testing

In this section we will discuss some testing which was performed on the system during and after implementation.

Each of the modules were built and then unit tested using dummy data. This was an incremental process of logging the outputs of the intermediary stages. After each test the logs were examined and errors and their causes identified and rectified. This was particularly true for the case of developing the grammar used in the chunking process.

After the implementation was completed we also tested the systems performance. We found that in all cases the system was able to generate a timeline in under two minutes. We also found that when comparing individuals with sparse data against ones with a lot of data, such as Minto vs Mandela, there is, as expected, an increase in the amount of time required to process the data. This increase is very small with an approximate difference in time of about a second. The component of the system which was found to take the most time was the retrieval of the structured data due to the amount of requests being sent to external servers.

6 Conclusion and Future Work

In this section we bring this research to a close by first discussing how we met the aims and objectives and also discuss future works and possible extensions to the system.

6.1 Accomplishments and Limitations

In this research we have developed a prototype system which is capable of obtaining structured and unstructured information, processing it by extracting dates and phrases which relate to a targeted individual. We have also visualised the extracted information in a timeline that can be viewed through a browser. In order to review this work we shall refer back to Section 1.3.

- **Obtaining, resolving and managing structured data from WikiData.** Data can be successfully collected from WikiData through the use of a custom wrapper that was developed to interface with the MediaWiki API. The data which is collected is then resolved by requesting the server for additional information. Finally we manage and store the information in bespoke indexing objects.
- **Obtaining unstructured text from Wikipedia.** We collect unstructured information from Wikipedia through the use of a wrapper for the MediaWiki API. The text is then scraped and preprocessing carried out.
- **Extracting dates from phrases.** This has been accomplished with very promising results as shown in Section 5.3.1. In spite of the promising results there are still issues which can be improved upon. Better handling for a wider variety of date formats one obvious case.
- **Extracting relevant phrases from the text.** We have been successful in extracting dates which are relevant to the target individual. From the results ordained in Section 5.3.3 we demonstrated good precision and recall but both still can be improved. We also recognise that the multi-date extraction process achieved reasonable results, however it is still rather limited we believe that improving this module would result in an improved recall score.

- **Generation of the timeline.** Timeline have been generated and visualised through the use of Timeline JS. We however feel there is room for improvement. The timeline suffers from overcrowding since a lot of events are plotted onto it. This is especially true for more notable individuals. Another limitation of the system is that it is vulnerable to repeating the same information. We believe that these two issues are two of the main limitations of the system.

6.2 Future Work

In this section we present a series of lists of possible future work which if carried out will reduce the limitations of the system and increase its usability.

Date extraction:

- Adding functionality to recognise a wider range of date formats. One method of tackling this problem may be to extend the current methods by first running the system on a large number of appropriate texts, identifying the mistakes and making the appropriate adjustments. Another approach would be to use the mistakes to build a larger corpora of examples, then these examples would need to be manually cleaned and improved. Once the set is complete then it could be used to train a classifier to identify dates types. The classifier would need to be used in conjunction with a system that can then extract the date in an appropriate manner based on the type.
- Extending the system to be able to catch implicit dates. Assuming that implicit dates can be recognised accurately by the previously suggested improvement. Information obtained from structured data and information obtained while processing the text can be used to build a memory of temporal facts. These fact can then be used to resolve the implicit dates.

Multi-date phrase processing:

- Improvements to the grammar. This could be performed as a trial an error process to incrementally improve the chunker's performance. Another possible solution to this problem would be that of creating a very large collection of multi date phrases. Then using unsupervised machine learning techniques to identify common patterns and create specific grammars for each pattern. In a production system a phrase would be categorised and then parsed with an appropriate grammar.

- A limitation of the system was discovered in the parsing of the trees generated by the chunking process. Improvements to the algorithm parse the tree and build the new phrases can be implemented by trial and error.

Date phrase processing:

- Improved resolution of sentences including intra-sentence resolution will go a long way to improving the overall performance of the system. This is especially true with regards to resolving the current subject of a phrase. Improvements in this area will help resolving errors such as the system not being able to distinguish between two people with almost identical names such as ‘Obama Jr’ and ‘Obama Sr’.
- Specific method to identify events in sentences that do not contain dates, but nonetheless still describe some event. This may be accomplished with the use of an NLP module that scans for specific keywords or phrases that indicate such phrases. Another possible method of detecting these phrases would be through the use of trained classifiers.

General improvements to the system to improve the timeline generation:

- The addition of an information aggregation or symmetrisation system. The timelines presented can become very crowded. One of the contributing factors is the same information being extracted multiple times from within the same source. This problem is made further complicated by the fact that the same events may not be labelled with the same date. This may be caused either due to an error or a difference in precision.
- Another possible solution to the over crowding problem would be to obtain unstructured information from multiple sources. Then the relevance of each fact could be determined on the number of times it is found in the separate sources. A system such as this may also be improved by the development of an algorithm which uses an ontology or knowledge base to identify keywords associated with relevant events.
- A final improvement would be to also gather multi-media items such as images, videos or audio that can be displayed alongside the extracted information.

6.3 Final Remarks

In this research we presented a prototype system that has been designed to tackle the aims and objectives set in Section 1.3. The system achieved respectable results in evaluation and on the whole achieved the aim of the research. While there remains room for improvement we are satisfied with the systems overall performance.

References

- [1] R. Campos, G. Dias, A. M. Jorge, and A. Jatowt, “Survey of temporal information retrieval and related applications,” *ACM Comput. Surv.*, vol. 47, pp. 15:1–15:41, 2014.
- [2] Wikipedia, “Wikipedia:about.” <https://en.wikipedia.org/wiki/Wikipedia:About>, 2016. Accessed: 2016-5-10.
- [3] D. Vrandečić and M. Krötzsch, “Wikidata: A free collaborative knowledgebase,” *Commun. ACM*, vol. 57, pp. 78–85, Sept. 2014.
- [4] B. C. Bruce, “A model for temporal reference and its application in a q/a program,” *Artificial Intelligence*, vol. 3, pp. 1–25, 1972.
- [5] J. F. Allen, “Maintaining knowledge about temporal intervals,” *Commun. ACM*, vol. 26, pp. 832–843, 1983.
- [6] F. Schilder and C. Habel, “From temporal expressions to temporal information: Semantic tagging of news messages,” in *Proceedings of the Workshop on Temporal and Spatial Information Processing - Volume 13*, TASIP ’01, (Stroudsburg, PA, USA), pp. 9:1–9:8, Association for Computational Linguistics, 2001.
- [7] J. Pustejovsky, J. M. Castaño, R. Ingria, R. S. Colomer, R. J. Gaizauskas, A. Setzer, G. Katz, and D. R. Radev, “Timeml: Robust specification of event and temporal expressions in text,” in *NDQA*, 2003.
- [8] N. UzZaman, H. Llorens, L. Derczynski, J. Allen, M. Verhagen, and J. Pustejovsky, “Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations,” in *In Proceedings of the SemVal’13 Workshop*, 2013.
- [9] J. Strötgen, J. Zell, and M. Gertz, “Heideltime: Tuning english and developing spanish resources for tempeval-3,” in *In Proceedings of the SemVal’13 Workshop*, 2013.
- [10] N. Chambers, “Navytime: Event and time ordering from raw text,” in *In Proceedings of the SemVal’13 Workshop*, 2013.
- [11] S. Bethard, “Cleartk-timeml: A minimalist approach to tempeval 2013,” in *In Proceedings of the SemVal’13 Workshop*, 2013.

- [12] A.-L. Minard, M. Speranza, E. Agirre, I. Aldabe, M. V. Erp, B. Magnini, G. Rigau, R. Urizar, F. B. Kessler, and I. Trento, “Semeval-2015 task 4: Timeline: Cross-document event ordering,” in *In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval2015)*, pp. 776–786, 2015.
- [13] E. Laparra, I. Aldabe, and G. Rigau, “Document level time-anchoring for timeline extraction,” in *ACL*, 2015.
- [14] A. Mishra and K. Berberich, “Exposé: Exploring past news for seminal events,” in *WWW*, 2015.
- [15] J. Li and C. Cardie, “Timeline generation: Tracking individuals on twitter,” *CoRR*, vol. abs/1309.7313, 2014.
- [16] S. F. Silva and T. Catarci, “Visualization of linear time-oriented data: A survey,” in *WISE*, 2000.
- [17] V. Kumar, R. Furuta, and R. B. Allen, “Metadata visualization for digital libraries: Interactive timeline editing and review,” in *DL*, 1998.
- [18] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman, “Lifelines: Visualizing personal histories,” in *CHI*, 1996.
- [19] MIT, “Simile widgets free, open-source data visualization web widgets, and more.” <http://simile-widgets.org/>, 2009. Accessed: 2016-5-10.
- [20] T. Althoff, X. L. Dong, K. Murphy, S. Alai, V. Dang, and W. Zhang, “Timemachine: Timeline generation for knowledge-base entities,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’15, (New York, NY, USA), pp. 19–28, ACM, 2015.