

Wright State University

CORE Scholar

[Browse all Theses and Dissertations](#)

[Theses and Dissertations](#)

2015

Temporally Biased Search Result Snippets

J. Abhiram Tatineni

Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all



Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Repository Citation

Tatineni, J. Abhiram, "Temporally Biased Search Result Snippets" (2015). *Browse all Theses and Dissertations*. 1588.

https://corescholar.libraries.wright.edu/etd_all/1588

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.

Temporally Biased Search Result Snippets

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science

By
J Abhiram Tatineni
B.Tech., KL University, 2009

2015
Wright State University

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

February 17, 2015

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY
SUPERVISION BY ABHIRAM J. TATINENI ENTITLED Temporally Biased Search
Result Snippets BE ACCEPTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF Master of Science.

Krishnaprasad Thirunarayan, Ph.D.
Thesis Director

Mateen Rizki
Chair, Department of Computer Science and
Engineering

Committee on
Final Examination

Krishnaprasad Thirunarayan, Ph.D.

Keke Chen, Ph.D.

Tanvi Banerjee, Ph.D.

Robert E. Fyffe
Vice President for Research and
Dean of the Graduate School

ABSTRACT

Abhiram, Tatineni. M.S., Department of Computer Science and Engineering, Wright State University, 2015. *Temporally Biased Search Result Snippets*.

The search engine result snippets are an important source of information for the user to obtain quick insights into the corresponding result documents. When the search terms are too general, like a person's name or a company's name, creating an appropriate snippet that effectively summarizes the document's content can be challenging owing to multiple occurrences of the search term in the top ranked documents, without a simple means to select a subset of sentences containing them to form result snippet.

In web pages classified as narratives and news articles, multiple references to explicit, implicit and relative temporal expressions can be found. Based on these expressions, the sentences can be ordered on a timeline.

In this thesis, we propose the idea of generation of an alternate search results snippet, by exploiting these temporal expressions embedded within the pages, using a timeline map. Our method of snippets generation is mainly targeted at general search terms. At present, when the search terms are too general, the existing systems generate static snippets for resultant pages like displaying the first line. In our approach, we introduce an alternate method of extracting and selecting temporal data from these pages to adapt a snippet to be a more effective summary. Specifically, it selects and blends "temporally interesting" sentences. Using weighted kappa measure, we evaluate our approach by comparing snippets generated for multiple search terms based on existing systems and snippets generated by using our approach.

Contents

Introduction	1
1.1 Search results snippets	2
1.2 Alternatives for general search terms	4
1.3 Outline	8
Related Work	9
2.1 Temporal data in search results	9
2.2 Temporal data extraction from Wikipedia.....	14
Temporally Biased Snippets	16
3.1 Temporal tagging	18
3.2 Contextually Irrelevant Expressions Elimination	29
3.3 Elimination of Sentences Irrelevant to Search Terms	30
3.4 Sentence ranking and selection criteria.....	36
Implementation and Evaluation.....	40
4.1 Implementation	40
4.1.1 Temporal tagging algorithm	40
4.1.2 Algorithm for Contextually Irrelevant Expressions Elimination.....	43
4.1.3 Algorithm for Elimination of Sentences Irrelevant to Search Terms.....	44
4.1.4 Algorithm for Sentence ranking and summarizing	46
4.2 Evaluation.....	49
Conclusion and Future Work.....	52
Bibliography	53
Appendix A	55

List of Figures

1.1	Google Searcher per Year	2
1.2	Search results for “android studio installing sdk 22”	3
1.3	Search result for “Christopher Columbus”	4
1.4	Knowledge graph result for “Tesla Motors Inc”	5
1.5	Answer Box result for “Sapphire”	6
1.6	Search result page for “fifa world cup”	7
2.1	Baseline and Proposed temporal result snippet for ‘Tom Bosley’ [10].....	10
3.1	Temporal expressions in a simple paragraph identified by SUTime temporal tagger	24
3.2	Temporal expressions in a simple paragraph identified by HeidelTime temporal tagger	27
3.3	Graph showing the years appearing in the below excerpt	31
3.4	Graph showing the years appearing in the below excerpt.....	32
3.5	Graph showing the years appearing in an excerpt from the Wikipedia document ‘Martin Luther King, Jr.’	33
3.6	Graph showing the years appearing in an excerpt from the Wikipedia document ‘Normandy’	34
3.7	Normalized value of years in a window.....	36
3.8	Calculated $f(t,d) + \lambda$ values for the wiki document Apple Inc.....	39
4.1	Pseudo code for Heidel Time integration.....	41
4.2	Pseudo code for creating temporal tagged documents using Heidel Time.....	41
4.3	Pseudo code for extracting sentences with temporal expressions and their temporal value	43
4.4	Pseudo code for identifying contextually irrelevant temporal expressions	44
4.5	Pseudo code for rescaling temporal values of sentences.....	45
4.6	Pseudo code for identifying the lowest temporal value	45
4.7	Pseudo code for identifying the highest temporal value.....	45
4.8	Pseudo code to identifying a spike in temporal values.....	46
4.9	Pseudo code to calculate number sentences for each year appearing in a document	47
4.10	Pseudo code to generate temporal biased snippet	48

List of Tables

Table 2.1 Users comments received for survey on situations that did not present adequate information in search snippet [4].....	12
Table 2.2 Users comments received for size of the snippet [4]	13
Table 3.1 TIMEX3 values for “Last winter, they met every Thursday afternoon, from 10:00 am to 2:00 pm.”	21
Table 3.2 Comparing features of the three temporal tagger tools	28
Table 4.1 Weights for Kappa Measure	49
Table 4.2 Observed probabilities	50
Table 4.3 Expected probabilities.....	50
Table 4.4 Observed Values	51

Acknowledgment

I would like to express my warmest gratitude to everyone who helped me in the completion of this thesis and those who became a part of my life at Wright State University.

I would specially like to thank my thesis advisor Dr. T.K. Prasad for this opportunity. I am thankful for his incredible support and always taking time out from a busy schedule to help me with my thesis. I would also like to thank Dr. Keke Chen and Dr. Tanvi Banerjee, for readily agreeing to be on the thesis committee and their valuable feedback.

Last but never the least, I would like to thank my parents Anjali and Prasad and all my friends for their encouragement and support.

“Dream is not that which you see while sleeping it is something that does not let you sleep.”

-APJ Abdul Kalam, Wings of Fire.

Introduction

The World Wide Web is perhaps the greatest invention man has made. It has become an integral part of everyday life. The usage of Internet has changed the way people think and work. In the 1990's and before, studies have shown that people preferred to ask other people for information [1]. Today people are no longer dependent on others for things like latest news or even driving directions.

"But do you know that, although I have kept the diary [on a phonograph] for months past, it never once struck me how I was going to find any particular part of it in case I wanted to look it up?"

—Dr Seward, Bram Stoker's Dracula, 1897

Fig 1.1 shows the trend in number of Google searches made each year spanning over a decade and half. The graph shows an exponential growth in the number of search queries being made by Internet users. This can be attributed to the increase in number of Internet users and number of Internet website pages. As of 2013, the number of pages indexed by Google is about 50 billion¹.

¹ <http://expandedramblings.com/index.php/by-the-numbers-a-gigantic-list-of-google-stats-and-facts/>

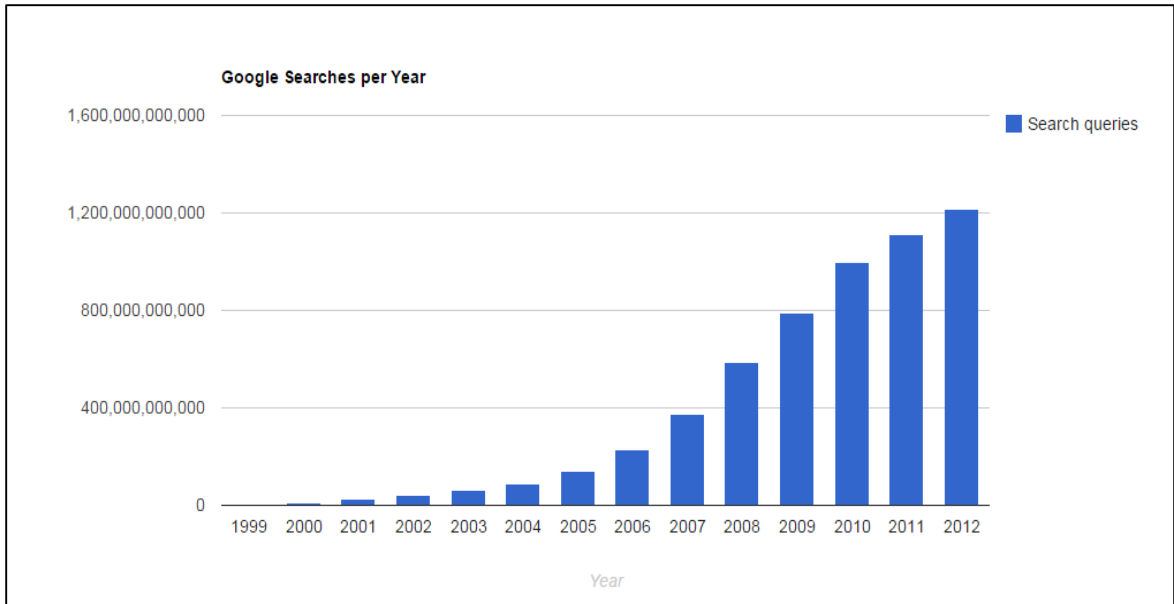


Figure 1.1 Google Searcher per Year

1.1 Search results snippets

Normally, outcome of a search is a collection of URL links to retrieved pages, and an accompanying snippet which is an excerpt from the result page. In early search engines, snippets were generated based on a fixed number of bytes from lines that appear first in the page. These snippets were generated independent of the search terms and were referred to as query independent snippets. Google was one of the first to introduce query biased snippets [19, 17]. The snippet's sentences might be from header, body, within a tag or from the meta-data of the page [14, 16]. These sentences also contain the search query terms referred to as KWIC (Key Words In Context)². The keyword appear bold within these sentences selected for the snippets. The sentences displayed are also partial, only a few words closer to the key words within a sentence are displayed instead of the entire sentence, and these are determined by content features [15]. There are several methods available for scoring of sentences in the page for generating the snippet, based

² <http://searchengineland.com/anatomy-of-a-google-snippet-38357>

on various features [17, 20, 21, 33, 34]. Fig 1.2 shows the top two search results displayed when we search for “android studio installing sdk 22”. The snippet gives the user a better understanding of the content within the pages.

However there are cases in which the search terms are too general, like when search is done for a specific person using his/her name or for an organization using its name. Fig 1.3, displays the search results from Wikipedia for ‘Christopher Columbus’. In these cases, the snippets retrieved contains the first line from the web page. This doesn’t really help the users to decide if the result page contains enough information to satisfy the user’s need. Alternatively, a better summary with more information from the page might help users to decide the richness of information in each result page.

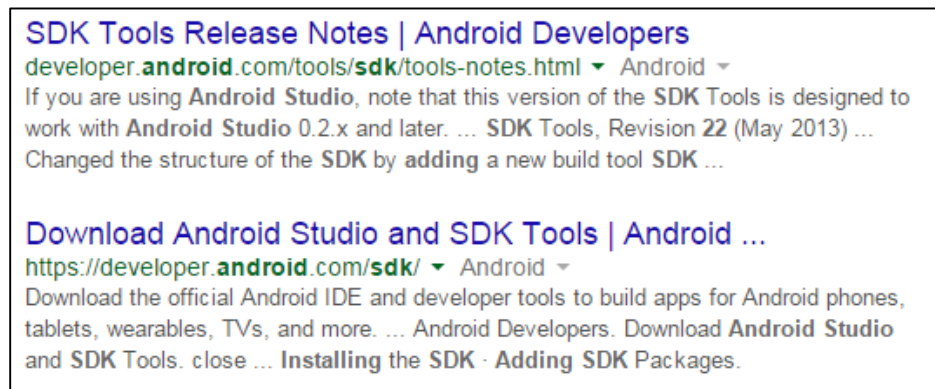


Figure 1.2 Search results for “android studio installing sdk 22”



Figure 1.3 Search result for “Christopher Columbus”

1.2 Alternatives for general search terms

Here we listed a few alternatives to displaying search results when the search terms are too general and also the current limitations.

More recently, knowledge graphs have been implemented that extract new knowledge from various sources. This extracted knowledge is represented in the form of facts. For example, facts for a person can be his date of birth, occupation, noted works and more.

Knowledge Graph: The knowledge graph when introduced by Google boasted of 3.5 billion facts about 500 million objects³ and it is increasing every day. One of its goal is to summarize content relevant to a topic. For example, if a person is searching for ‘Tesla Motors Inc’, they would also find additional information regarding the current stock price and latest models available. Fig 1.4 shows the knowledge graph result for ‘Tesla Motors Inc’. What kind of facts to retrieve for each entity being searched is decided based on user’s search logs collected over time. For example, it was found that people were more interested in books written by author Charles Dickens and less interested in books written by Frank L. Wright but more interested in Frank L. Wright’s design work [18].

³ <http://searchengineland.com/google-launches-knowledge-graph-121585>

Tesla Motors

Automotive company

Tesla Motors, Inc. is an American automotive and energy storage company that designs, manufactures, and sells electric cars, electric vehicle powertrain components, and battery products. [Wikipedia](#)

Founded: 2003, [San Carlos, CA](#)

Stock price: [TSLA](#) (NASDAQ) \$274.66 0.00 (0.00%)
Jul 17, 4:01 PM EDT - Disclaimer


Customer service: 1 (877) 798-3752

Sales: 1 (888) 518-3752






CEO: [Elon Musk](#)

Latest models: 2015 Tesla Model S, 2011 Tesla Roadster

Founders: [JB Straubel](#), [Martin Eberhard](#), [Ian Wright](#), [Elon Musk](#), [Marc Tarpenning](#)








Profiles

 Facebook
 LinkedIn
 Twitter
 YouTube
 Google+

People also search for

[View 15+ more](#)

 SpaceX
 SolarCity
 Maserati
 Fisker Automotive
 BMW

[Feedback](#)

Figure 1.4 Knowledge graph result for “Tesla Motors Inc”

Apart from Google, there are also other information extraction systems available – OpenIE [22] and the Never Ending Language Learning project [23] which use various fact extraction algorithms from knowledge bases.

Answer Box: Another new addition to the way search results are displayed is the answer box. Though this feature is mainly targeted at questions users' type in as a search query, it is found to display results for general search terms. Initially the number of resources used for finding information was limited to Wikipedia, Google+, Freebase [26] and YAGO [27] [25]. Fig 1.4 shows the Answer Box result for "Sapphire". But though these knowledge bases containing huge world knowledge are available, the knowledge bases were considered incomplete due to missing data. For example, in Freebase the place of birth of more than 70% of the people are missing [24]. More recently, in addition to the information available in knowledge bases, the Answer Box has been extended to pull information from third party sites directly.

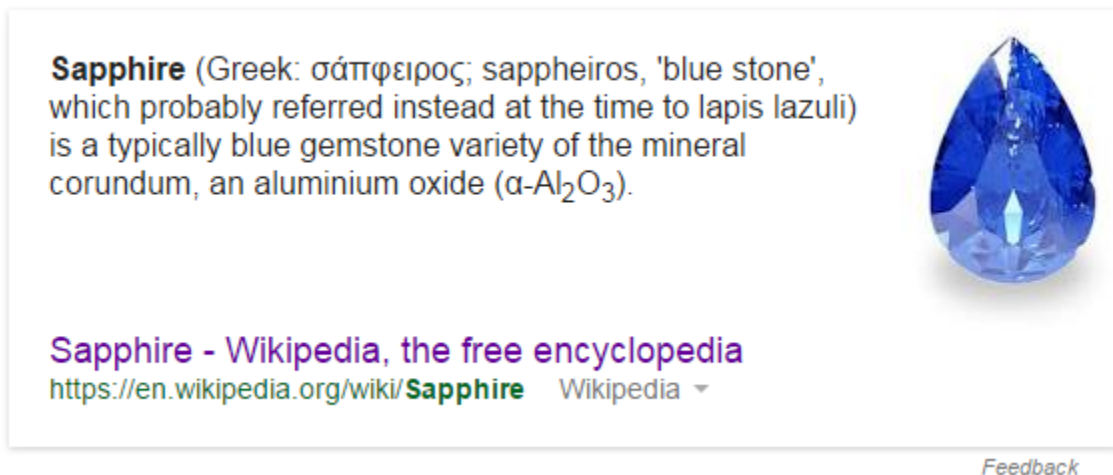


Figure 1.5 Answer Box result for "Sapphire"

Though the Knowledge Graph and Answer Box give a good solution to display more data on the results page for general search terms, one main drawback found is that it is not available for every generalized search query. For example, the search term “FIFA World Cup” shows plain results with snippets without the Answer Box or the Knowledge Graph.

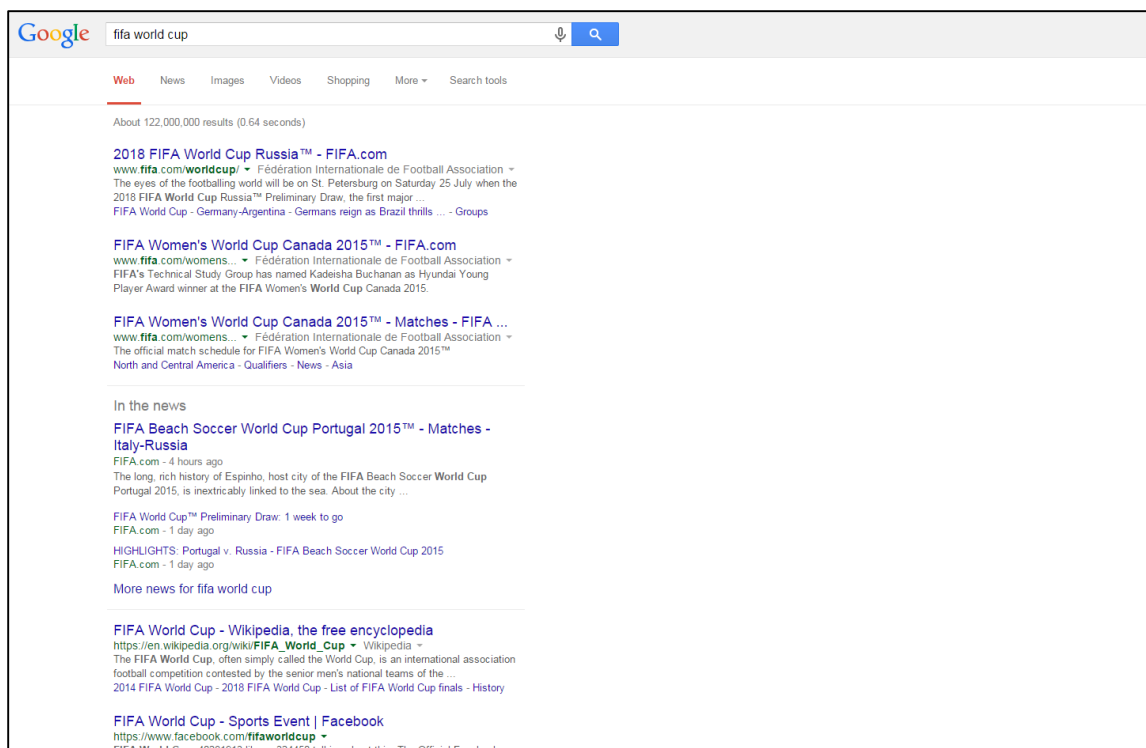


Figure 1.6 Search result page for “fifa world cup”

In this thesis, we propose an alternative method of snippet generation for general search terms. We base our research on the finding that most of the information available about entities can be expressed in terms of time. Many sentences within a page or a document can have a temporal value associated with it based on the type of document

(narratives and news), it can be represented on the basis of when the event in the sentence occurred or when that particular sentence was created.

Relations between two events can also be identified based on their temporal values [3]. The order of occurrence of these events can be determined. Exploiting this temporal value of the content can be used to represent the entire content on a time map.

Multiple values of temporal data can be present in each document. These temporal values can be represented in different forms in the documents. It is important to identify the temporal value associated with the sentence and mark it for evaluation purpose.

In our approach, we exploit this temporal element to create alternate summaries and evaluate its effectiveness.

1.3 Outline

Here is the organization of the rest of this document. In Section 2, we discuss previous related works in the area of temporal data in search results. In Section 3, we will mention the different types of temporal data that can appear in a document and how to identify temporal references. We will also discuss various techniques to filter out noisy data and our approach to select sentences from each document for its search result snippet. In Section 4, algorithms implemented for each step in Section 3 will be shown. Evaluation of our approach is also given here. Thesis conclusion and future prospects will be discussed in Section 5.

Related Work

In this section we discuss the works already published related to exploiting temporal values in documents. We also include works which use temporal values in search results and also outline their limitation.

2.1 Temporal data in search results

Temporally Dynamic Search Snippets: Svore et al. introduced the idea of creating a dynamic result page snippet biasing new page content which may help users make a better decision on the relevance of the page [10]. Their approach is based on the fact that content is always changing on the internet and pages are constantly updated and this drives people to visit these pages to stay updated [32]. An example for this approach is a search made for 'Tom Bosley'. Fig 2.1 shows the search result for 'Tom Bosley' and a proposed result generated by including new content from the page.



Figure 2.1 Baseline and Proposed temporal result snippet for 'Tom Bosley' [10]

The importance of the temporal snippet when the news about Tom Bosley's death is still new might help the user to make a better decision about page relevance. In this approach two lengths of snippet are considered - 2 lines referred to as short snippet and 4 lines referred as long snippet. Additionally, variation of the snippet generation is also introduced – one that contains a blend of original and temporal snippet and second that contains only the temporal snippet. A crawl is made on the new version of the page and also on a cached version. A cached version is considered here to determine the changes made to the page. The short baseline snippet is created based on the earliest appearance of search terms in the new version. The same procedure is followed for long baseline snippet, but four sentences are chosen.

For the temporal snippet, sentences that have changed the most in the new version from the cached copy is determined using Dice coefficient. All the sentences are then ranked based on their dice scores. All ties are handled based on the position of the

sentence and the appearance of search terms. For the short blend snippet, top ranked sentences from each of temporal ranking and baseline ranking is considered. For the long blend snippet, two top ranked sentences from each of temporal ranking and baseline ranking are considered.

This approach is evaluated using crowd sourcing approach and the evaluators are provided with the different snippets generated as discussed above. The study showed an inclination towards including new content or the latest events in the snippets especially in cases when the queries are trending.

TSnippet: Alonso et al. introduced the idea of creating search results snippets based on temporal information in the pages [4]. In their study, initially a set of questions were posted as a Human intelligence task to users. One question asked users to point to scenarios when adequate information was not present in the search snippets. Fig 2.2 shows the user comments received for this question.

Category	User comment
Poor quality	If search terms are too generic, impossible to find information through snippet. Usually have to search through several sites to find the information - A common problem is having a snippet that is out of context. - There are some sites that basically give an introduction as their snippet, which is sometimes annoying. - I do not click on a link when the snippet just contains random keywords, especially if some of them are not at all related to my search. - When snippets are too general or when they don't support the title in any logical way.
Depth, sentence length	Not showing when a famous person was born/died. - When looking for profound information usually it doesn't all show up in the snippet - If it does not fully answer my question, I will click on the page. - If the full sentence didn't complete with the information I needed. I would then click into the site to read the rest of the sentence.
Metadata	If I'm looking for hotels close by the University I don't think I would be able to find any hotels by clicking on the links. - I remember seeing some results in a foreign language, but it does not specify which language it most likely is, so I don't know which language to translate from on my translator extension.

Table 2.1 Users comments received for survey on situations that did not present adequate information in search snippet [4].

One more interesting comment observed in this survey was regarding the size of the snippet. Fig 2.3 shows the comment received about the size of the snippets. This survey shows various concerns about lack of temporal information and on the size of the snippets.

Sentence length	It might be good if they were a line or two longer because sometimes the info in the snippet is cut off and hard to understand. - Maybe scrolling in snippets - Full sentences. - A way to enlarge the text field and read more of the snippet - It might be neat if people could customize the number of lines for each snippet.
-----------------	---

Table 2.2 Users comments received for size of the snippet [4]

The TSnippet was generated based on a two-step approach- (i) Sentences with temporal values are selected, (ii) The selected sentences are ranked using a ranking function. Initially, for the sentence selection, the first line containing a temporal value was considered, but this approach was disregarded due to the following scenarios – (i) Appearance of photo captions as the first sentences, (ii) Very long sentences were observed, mainly in cases where there was a table and the detectors failed to identify it, and (iii) There is a relative temporal expression in the first line. Based on these observations, the sentence selection criteria is modified to – (i) The sentences with at least one temporal value are considered, (ii) Only explicit temporal values are considered, and (iii) the sentence length is within a predefined threshold.

The following were considered for ranking the temporal sentences –

- p - the position of identified temporal value
- s - the sequence number of sentence .
- sl - the length of the sentence.
- co – the order of appearance of chronon (units of time).
- cf - the frequency of chronon in the document.

- cfs - the frequency of chronon in the sentence.

Based on the above mentioned values, in addition to the cosine similarity of each sentence with the search query, the sentence rank is determined. The top sentences after ranking are selected for the snippet. These sentences are displayed in chronological order.

This approach was evaluated using crowd sourcing against search result snippets from two existing search engines. Alonso et al. approach showed a favorable inclination of users towards finding time sensitive information in the search results snippet.

Campos et al. showed that often queries have an underlying temporal intent, though they are not explicitly given [21]. For example, a user searching for 'Presidential Election' might actually be looking for information regarding the next Presidential election rather than history or general information about 'Presidential Election'

2.2 Temporal data extraction from Wikipedia

Kuzey et al. introduced an information extraction framework for semi structured data and free text [28]. This extracts temporal facts and events from Wikipedia and creates a temporal ontology. Initially they created a temporal data representation model. This model is made aware of the events. They also created a tool which extracts data from info

boxes available in the page. Additionally, temporal values that appear in categories, titles and lists are also extracted. Finally, the free text is also considered for extraction of temporal values. All this data is used to create a temporal knowledge base.

Kuzey et al. highlight the fact that many of the existing knowledge bases like DBpedia[29], KnowItAll[30], Intelligence In Wikipedia[31] and YAGO[27] containing millions of entities and their relations, focus mainly on static facts. The implementation is evaluated using English Wikipedia XML dump. Kuzey et al. highlights the areas within Wikipedia with rich temporal data and which can be harvested to obtain facts.

Temporally Biased Snippets

In this Chapter, we discuss and analyze the identification, extraction, filtering and selection of sentences for the snippet.

The first step in processing a document to generate a temporally biased snippet is to identify all the temporal expressions within the document. TimeML is a markup language for annotating temporal and event expressions that occur within a document⁴. Identifying temporal expressions is discussed in detail in Section 3.1. We make use of this markup to tag various events and expressions. In a document there can exist three types of temporal expressions – explicit temporal expressions, implicit temporal expressions and relative temporal expressions [4].

Step1: Identifying temporal expressions: *Explicit temporal expressions* are values which correspond to a specific date. For example, “12th Jun 1998” and “07/21/05”. The day, month and year values are explicitly available. *Implicit temporal expressions* correspond to an event that occurred at a point in time, but the time information can be deduced by knowing the type of event. For example, “New Year day 2001” and “Thanksgiving of 2014”. These values state an event. *Relative temporal expressions* are references in point of time that are dependent on another temporal expression to get its actual value. For example, “the next day”, “one month later”. These cannot be mapped to

⁴ <http://www.timeml.org/site/index.html>

a time point without knowing the context. These type of temporal expressions are the hardest to identify. Additionally, there are various formats of date that can appear in documents such as – ‘mm/dd/yyyy’, ‘mm-dd-yy’ and ‘dd-mm-yyyy’. This requires complex date recognizer for completeness.

Therefore, for generating an effective temporally biased snippet, it is very important to identify and consider all three types of expressions along with various date formats. A technique that is powerful enough to identify all the three types of expressions should be considered. A few techniques have been evaluated for this purpose in Section 3.2.

Step2: Contextually Irrelevant Expressions: The second step deals with eliminating temporal expressions which are already known not to relate to a point in time, disassociated with the context of a document, but still has temporal keywords. For example ‘copyrights 2015’ and ‘Act of 2009’. It is important to eliminate these types of references as they lead to false positives in the selection criteria for the snippet generation. Elimination process for these type of values will be discussed in Section 3.2.

Step3: Sentences Irrelevant to Search Terms: The third step deals with eliminating sentences that are considered to be out of context to the subject being searched. It can be observed in several documents that there are references to points in time which are not related to the timeline of the subject being searched. For example, consider sentences such as – “His work was inspired by Shakespeare’s Hamlet written in 1599” or “Nathan became Governor for a consecutive time after the state amended its rule for allowing same Governor in succession, in the year 1995”. In the first example, the year 1599 will be identified as temporal data, but it cannot be precisely considered as relating to the subject of the document. Same is the case with the second example, where 1995 will be identified as temporal data, but the year 1995 is about the rule being amended and the subject in question here is the Governor Nathan. So 1995 should not be considered as relevant for

the Governor. Identifying such sentences and the elimination criteria will be discussed in Section 3.3.

The main purpose of the above two steps is to eliminate all sentences that do not relate to the subject of the document, to select temporally relevant sentences to the subject matter.

Step4: Temporally Relevant Sentences: The final step deals with identifying the best sentences to show in the snippet, from the remaining subset. Various ranking methods will be considered to identify these sentences. Blending 3-4 interesting sentences will be considered. These methods will be discussed in Section 3.4.

3.1 Temporal tagging

TimeML was proposed during the 2002 TERQAS (Time and Event Recognition for Question Answering Systems) workshop which mainly focused on understanding and enhancing question and answer systems based on temporal data within news articles⁵.

TimeML is a case sensitive markup language which uses multiple tags to distinguish various time related events in text. These tags are used to annotate parts of text that correspond to a temporal value. It consists of various tags – **EVENT**, **MAKEINSTANCE**, **TIMEX3**, **SIGNAL**, **TLINK**, **SLINK**, **ALINK**, and **CONFIDENCE**⁶.

EVENT – This tag annotates elements in text that mark semantic events described by it

⁵ <https://en.wikipedia.org/wiki/TimeML>

⁶ http://www.timeml.org/site/publications/timeMLdocs/timeml_1.2.1.html

MAKEINSTANCE – This tag indicates different instances of a given event. The tense and aspect of the event are also described in this tag. Aspect here relates to the flow of time.

TIMEX3 – This tag is specifically used to markup explicit temporal expressions such as dates, times and duration.

SIGNAL – This tag annotates sections of text that indicate how temporal objects are to be related to each other.

TLINK – This tag is used to represent relations between two temporal elements.

SLINK – This is a subordination link that is used for contexts involving modality, evidential, and factives. It is used in cases where an event instance subordinates another event instance type. This is used in cases where a verb takes a match and subordinates the event instance. This instance is referred to in its complement.

ALINK – This is an aspectual link; it indicates an aspectual connection between two events. In some ways, it is like a cross between TLINK and SLINK in that it indicates both a relation between two temporal elements, as well as aspectual subordination

CONFIDENCE – This tag is used to mark the confidence values for various aspects of annotation.

Similar to XML, TimeML also has a root node that is TimeML. For our purposes, we are mainly interested in the TIMEX3 tag. This tag is used to mark temporal data with an explicit time value. The following are TIMEX3 attributes -

```
attributes ::= tid type [functionInDocument] [beginPoint] [endPoint]
               [quant]   [freq]   [temporalFunction]   (value      |
valueFromFunction)
               [mod] [anchorTimeID] [comment]

tid ::= ID
{tid ::= TimeID
TimeID ::= t<integer>}
type ::= 'DATE' | 'TIME' | 'DURATION' | 'SET'
```

```

beginPoint ::= IDREF
{beginPoint ::= TimeID}
endPoint ::= IDREF
{endPoint ::= TimeID}
quant ::= CDATA
freq ::= Duration
functionInDocument ::= 'CREATION_TIME' | 'EXPIRATION_TIME' |
'MODIFICATION_TIME' | 'PUBLICATION_TIME' | 'RELEASE_TIME' |
'RECEPTION_TIME' | 'NONE' {default, if absent, is 'NONE'}
temporalFunction ::= 'true' | 'false' {default, if absent, is 'false'}
{temporalFunction ::= boolean}
value ::= Duration | Date | Time | WeekDate | WeekTime | Season |
PartOfYear | PaPrFu
valueFromFunction ::= IDREF
{valueFromFunction ::= TemporalFunctionID
TemporalFunctionID ::= tf<integer>}
mod ::= 'BEFORE' | 'AFTER' | 'ON_OR_BEFORE' | 'ON_OR_AFTER' | 'LESS_THAN'
| 'MORE_THAN' | 'EQUAL_OR_LESS' | 'EQUAL_OR_MORE' | 'START' | 'MID' | 'END' |
'APPROX'
anchorTimeID ::= IDREF
{anchorTimeID ::= TimeID}
comment ::= CDATA

```

tid – Unique identifier.

type – Type of temporal expression, can be 'DATE', 'TIME', 'DURATION' or 'SET'.

functionInDocument – It is an optional attribute which provides a temporal anchor for other temporal expressions in the document.

beginPoint – It anchors durations to other time expressions in the document.

endPoint - It anchors durations to other time expressions in the document.

quant – Used to specify a quantified time in TIMEX3 in literal form.

freq - Used to specify a quantified time in TIMEX3 in integer form.

temporalFunction – It specifies if the temporal expression is used as a temporal function. It can have an actual value or a pointer represented by valueFromFucntion.

mod - It is used for temporal modifiers that cannot be expressed either within value proper, or via links or temporal functions.

anchorTimeID – It is used to point to another TIMEX3 in cases which have functional interpretation.

For example, consider the line “Last winter, they met every Thursday afternoon, from 10:00 am to 2:00 pm.” The TIMEX3 tag values are shown in table 3.1

Text	Value	TIMEX3
Last winter	2014-WI	<TIMEX3 tid="t1" type="DATE" value="2014-WI">Last winter</TIMEX3>
every Thursday afternoon	XXXX-WXX-4TAF	<TIMEX3 periodicity="P1W" quant="every" tid="t2" type="SET" value="XXXX-WXX-4TAF">every Thursday afternoon</TIMEX3>
10:00 am	2015-07-21T10:00	<TIMEX3 tid="t3" type="TIME" value="2015-07-21T10:00">10:00 am</TIMEX3>
2:00 pm	2015-07-21T14:00	<TIMEX3 tid="t4" type="TIME" value="2015-07-21T14:00">2:00 pm</TIMEX3>

Table 3.1 TIMEX3 values for “Last winter, they met every Thursday afternoon, from 10:00 am to 2:00 pm.”

Extracting temporal data is key to many applications like question answering and summarization [6]. Therefore a powerful tagger which can precisely identify all the temporal values should be considered. We consider three temporal taggers which use TimeML markup – SuTime, HeidelTime and TRIPS/TROIS System, and evaluate their expressiveness using various examples.

SUTime : SUTime is a temporal tagger used for identifying and normalizing temporal values in text [5]. It is an integral part of Stanford CoreNLP. This rule based temporal tagger is built on pattern recognition implemented using regular expressions. The main features offered by SUTime are: **(i)** Extraction of temporal data from text, **(ii)** Representation of temporal objects as Java classes and **(iii)** Resolution of temporal expressions. SUTime identifies and annotates the temporal expressions within a document using TIMEX3 tags. It also extends ISO 8601⁷ standard. SUTime has tools to map temporal expressions to data structures for easy handling. It also uses the document creation time as reference for all the relative expressions that appear within a document. The relative temporal expressions are marked as the next instance that occurs after the creation of the document. For example, if a document was created on '14-Jun-2015', a Sunday, the appearance of the relative temporal expression 'next Sunday' would be marked as '21-Jun-2015'.

SUtime supports four basic types of temporal objects – Time, Duration, Interval and Set. Time is a specific point, which is an exact date. For example, 24th Feb 1999 is an

⁷ <http://www.iso.org/iso/iso8601>

instance of Time. Duration is the interval between two points of time. SUTime supports three types of duration – Exact duration, inexact duration and duration range. Exact durations are the specific durations, such as ‘25 days’. Inexact durations are values with no explicit mention but the unit of time is known, such as ‘a few months’. Duration ranges are the values that have a minimum and maximum specified, such as ‘7 to 8 years’. Interval is a range with two specific time values which are the interval’s boundary, such as ‘from March to September’. In TIMEX3, interval is not a separate type. This range is represented using the duration tag. Additionally, SUTime also supports time intervals between two date and time values. Set are the temporal values that occur with a frequency. For example, ‘every second saturday’.

Temporal expressions are specified in three different ways – text regex rules, compositional rules and filtering rules. Text regex rules are used to specify temporal values based on regex patterns over characters. The compositional rules deal with mapping regex over temporal data to their representations. Filtering is an important phase in which expressions which are false positives are eliminated. Parts of speech tags are used to identify the type, based on which the expression is determined as either a temporal value or not a temporal value, such as the word ‘fall’. If the part of speech tag is not a noun then it will be considered as the act of falling and not considered as the season.

A simple paragraph is given as input to SUTime temporal tagger, which is shown in Fig 3.1. It also shows the annotated text and all the identified temporal expressions identified using the rules mentioned above.

Stanford Temporal Tagger: SUTime

Please enter a reference date (format must be YYYY-MM-DD):

Date:

Please enter your text here (sample sentence):

On 7th Jan 1999, Mr. X won the presidential elections.
On 20th Jan he took the presidential oath. Four years
later, he won the presidential elections again.

Options:

☐ Mark time ranges
☐ Include nested
☐ Include range

Rules: English ▼

Annotated Text
(tagged using sutime)

On **7th Jan 1999**, Mr. X won the presidential elections. On **20th Jan** he took the presidential oath. **Four years later**, he won the presidential elections again.

Temporal Expressions

Text	Value	Timex3 Tag
7th Jan 1999	1999-01-07	<TIME3 tid="t1" type="DATE" value="1999-01-07">7th Jan 1999</TIME3>
20th Jan	2015-01-20	<TIME3 tid="t2" type="DATE" value="2015-01-20">20th Jan</TIME3>
Four years later	2019-06-14	<TIME3 tid="t3" type="DATE" value="2019-06-14">Four years later</TIME3>

Figure 3.1 Temporal expressions in a simple paragraph identified by SUTime temporal tagger

Heidel Time: Heidel Time is a context sensitive, rule based system for identifying and extracting temporal expressions based on regular expressions, knowledge resources and linguistics [7]. Its implementation is based on UIMA – Unstructured Information Management Architecture, which is extensively used for unstructured content. Hiedel Time maps every temporal expression to a three tuple set –

$$te_i = \langle e_i, t_i, v_i \rangle$$

e_i – Expression

t_i – Type of expression

v_i – Normalized value

The expression e_i is the original temporal expression occurring in the document text such as months 'May' and 'June' or seasons such as 'summer' and 'fall'. The expression t_i represents the type of the temporal expression, such as 'Date', 'Duration' and 'Set'. The expression v_i represents the normalized value of the temporal expression, such as '08' for 'August' and 'FA' for 'Fall'.

Heidel Time supports four types of temporal objects – Date, Time, Duration and Set. It also supports tags based on TimeML markup. It extracts all the temporal expressions and marks their type and specifies the normalized value. It uses hand crafted rules, which are also a triple of functions – expression function, type function and normalization function, and each generates the above mentioned expressions - e_i , t_i and v_i for every temporal expression. Heidel Time recognizes

all the three possible types of temporal expressions – explicit, implicit and relative types. It marks the temporal expressions as UNDEF when the year part of the expression is unknown. However, when there is a relative expression referencing time, it marks the expression with UNDEF-REF. For example, 'May' is marked as UNDEF, but expressions like, 'last May' or 'in May' or 'next May' are marked UNDEF-REF. Relative expressions marked as UNDEF-REF, which appear after a sentence with an explicit mention of year, will consider that year as the base for the relative expression and mark the relative expressions accordingly. This would be the case with documents classified as narrative. However, documents classified as news will use the document creation date as the base value for these types of relative expressions. Consider the two sentences 'In 1999, the world awaits the Y2K bug, with more drastic millennial theorists warning of Armageddon. The next year, life continued as usual. '. If this is from a document classified as *narrative*, the term 'next year' is normalized as year 2000 considering 1999 from the previous

sentence as base. If this document was classified as news, 'next year' would be normalized based on the creation date of the document.

To determine the exact value for expressions marked as UNDEF, the tense of the sentence is used, which is determined by parts of speech tagging. The tense is identified as either past, present or future, based on which the UNDEF value is post fixed with values 'last' for past tense and 'next' for future tense. All other invalid expressions are ignored after this post processing step. Invalid expressions are all the expressions that are already included in other expressions, such as the phrase 'November 23'. The whole phrase 'November 23' is found by a rule as well as just 'November'. Since 'November' is in 'November 23', 'November' is ignored.

Input

Choose between manually entering text and inserting a text file (up to 2 MB, ensure it's encoded in UTF-8).

☒ Text

☐ File

On 7th Jan 1999, Mr. X won the presidential elections. On 20th Jan he took the presidential oath. Four years later, he won the presidential elections again.

Output

Extracted temporal expressions are marked in blue. To see their normalization value, click them.

You may also receive a TimeML-annotated file (ensure your browser isn't blocking popups).

☒ Yes, I want to receive a TimeML-annotated file.

Compute

Resulting document:

On 7th Jan 1999, Mr. X won the presidential elections. On 20th Jan he took the presidential oath. Four years later, he won the presidential elections again.

Type: DATE

Value: 2003

Figure 3.2 Temporal expressions in a simple paragraph identified by HeidelTime temporal tagger

TRIPS and TROIS System: The TRIPS and TROIS system is a hybrid between linguistically motivated solutions and machine classifiers [9]. It makes use of the TRIPS parser to parse text and is based on a set of hand coded rules for extracting patterns to obtain temporal expressions. To further complement the process of TRIPS, machine

27

learning classifiers based on token-by-token classification are used with conditional random field (CRF) classifier. TRIPS parser extracts temporal relation between temporal expressions and events with CRF based extractor also identifying temporal expressions. Both are used in TRIPS and TROIS system implementation and differentiated by adding a TRIPS parser id for expressions identified by TRIPS.

After temporal values are extracted using regular expressions, they are normalized and TimeML scheme is used to tag the expressions. All the relative temporal expressions are normalized based on the creation date of the document.

	SUTime	Heidel Time	TRIPS and TROIS
Resolving relative expressions	Poor	Good	Poor
Support for ranges of dates	Poor	Good	Good
Multi language	Not supported	Supported	-
Holiday dates	Doesn't recognize	Recognizes	-
Handling ambiguous phrases	Poor	Poor	Poor
Support for non-whole number('a year and half')	Not supported	Not supported	Not supported

Table 3.2 Comparing features of the three temporal tagger tools

SuTime, TRIPS and TROIS have limitations with relative temporal expressions. In SuTime, support for ranges of dates is poor. However, Heidel Time scores better in the areas of identifying and normalizing relative temporal expressions and additionally supports multiple languages. The precision optimized rule set also adds for achieving

better results in interpreting semantics of temporal expressions. Based on all these points, we will be using Heidel Time for temporal data extraction.

3.2 Contextually Irrelevant Expressions Elimination

The next step after tagging all the temporal expressions in the document is applying a set of post processing rules to eliminate sentences with contextually irrelevant temporal expressions. The initial step would identify all the sentences containing temporal expressions but which do not relate to a point in time such as sentences containing 'Copyrights 2009'. All such sentences are eliminated. These sentences are identified based on a set of hand crafted rules containing keywords, and these keywords are searched in all the sentences. Eliminating such false positives would increase the quality of the results.

The following are a few examples of such fragments –

- All works are protected under Copyrights 2015
- The American Recovery and Reinvestment Act of 2009
- Mc Donald's Annual Report 2010
- The New York Times, retrieved 7th May 2005

These expressions appearing in sentences have a temporal value present, but they are not to be considered as temporal expressions on the time map. Considering these would give undue importance to the year appearing in the expression, thus interfering with sentence selection based on temporal importance for the search snippet.

These rules are maintained as an open list, which has the capability to accommodate additional rules to minimize false positive clauses. These rules are

implemented using regular expressions to identify occurrences within the sentences being parsed.

The purpose of this step is to get only the set of sentences that are relevant to the subject of the document.

3.3 Elimination of Sentences Irrelevant to Search Terms

The next step in post processing rules consists of identifying sentences that are out of context for the subject of the document. It is important to identify and filter such sentences to prevent unnecessary weightage to the years mentioned in these sentences, thereby improving the quality of summary snippet. For example, sentences in a paragraph which consists of temporal expressions with years in close range but one sentence among them contains a year with a far off range. Narrative documents are usually written with events and information in chronological order. Based on our observation in several narrative documents, it has been found that though this far off range sentence might be important to the context, the sentence itself wasn't related to the subject of the document. We will refer to such sentences as a *spike* in the rest of this document, as it deviates from the subject's timeline. The far off range of the year in the sentences is referred to as *deviation*. A year is considered as a *spike* if it is the only year with *deviation* in the set of sentences considered.

In the following pages, excerpts from various documents are given, with year values plotted on graphs. Each excerpt will be analyzed and methods to identify *spike* in the values will be determined. It is important to identify that there is exactly one value with

a large deviation, as having two or more values might signify the importance of that particular year, even when the deviation might be large in the set of sentences considered.

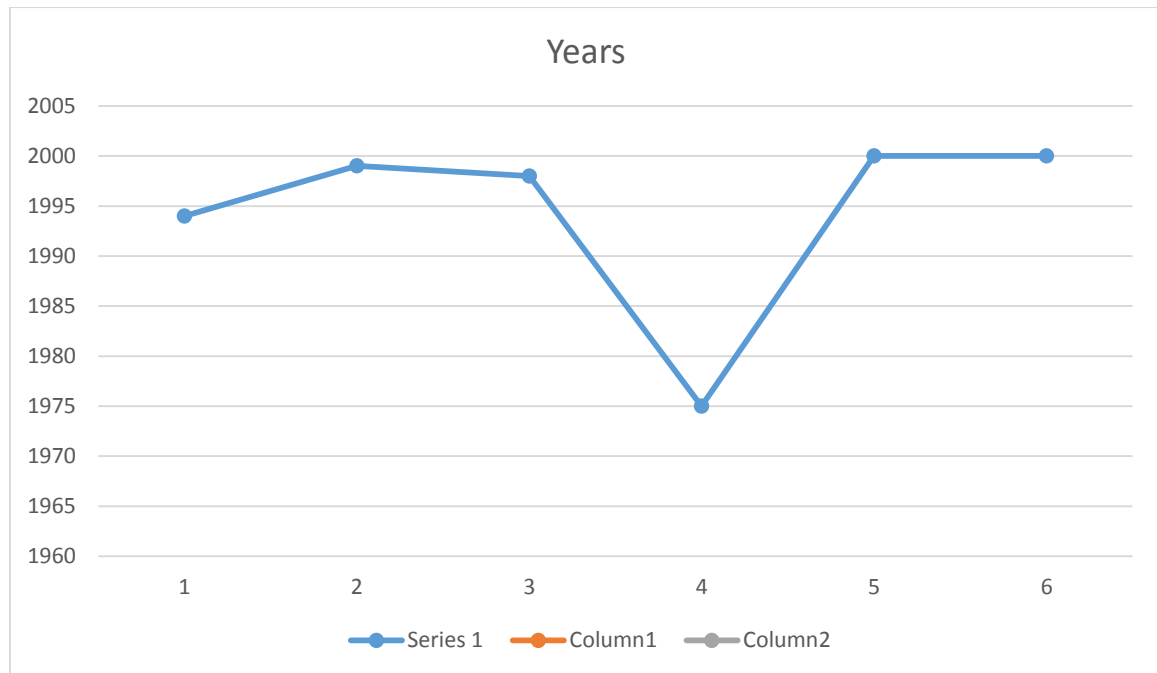


Figure 3.3 Graph showing the years appearing in the below excerpt

“As Bush's brother, Jeb, sought the governorship of Florida, Bush declared his candidacy for the 1994 Texas gubernatorial election. In 1999, Bush signed a state law obliging electric retailers to buy a certain amount of energy from renewable sources (RPS), which helped Texas eventually become the leading producer of wind powered electricity in the U.S. In 1998, Bush won re-election with a record 69% of the vote. He became the first governor in Texas history to be elected to two consecutive four-year terms. For most of Texas history, governors served two-year terms; a constitutional amendment extended those terms to four years starting in 1975. He proclaimed June 10, 2000 to be Jesus Day in Texas, a day on which he “urge[d] all Texans to answer the call to serve those in need”. Within a year, he decided to seek the 2000 Republican presidential nomination.”⁸

In the excerpt in Fig 3.3, it can be observed that the year 1975 is the lone deviation. The context of the sentence also is about a change in rule regarding governor's term in state of Texas that was changed in 1975. Such values need not be considered for

⁸ https://en.wikipedia.org/?title=George_W._Bush

the time line of the document as it is out of context to the subject of the document (George Bush). Thus we will eliminate the sentence “For most of Texas history, governors served two-year terms; a constitutional amendment extended those terms to four years starting in 1975” from consideration.

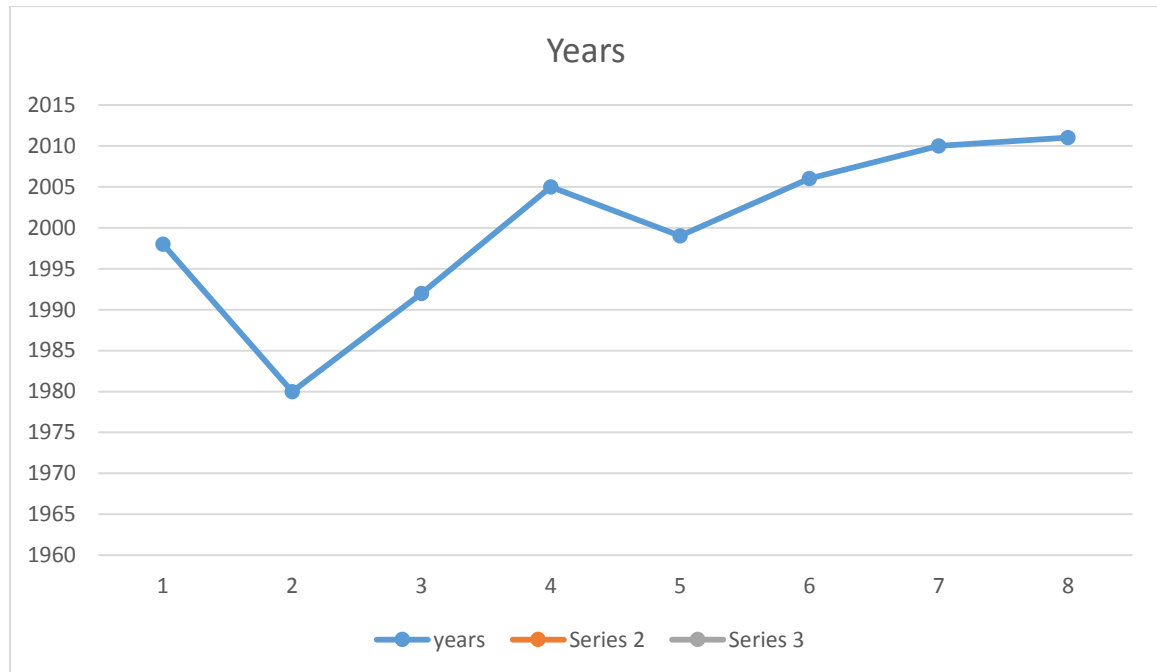


Figure 3.4 Graph showing the years appearing in the below excerpt

“In 1998, he released his second LP (his first album in 30 years) called Vincent LaGuardia Gambini Sings Just for You, which spawned the single “Wise Guy,” a rap number that played on the gangsta theme by referencing Mafia gangsterism.” “Wise Guy” interpolated the 1980 hit “Rapture” by Blondie, and was co-written and produced by the hip-hop production team the Trackmasters. Vincent LaGuardia Gambini Sings Just For You was an album that was both humorous and serious, exploring a variety of genres, though most of it was big band jazz, and which paid homage to his character name from the 1992 film My Cousin Vinny, not only through its album title, but also by its lead track “Yo Cousin Vinny”.

Pesci is associated with the hit Broadway musical Jersey Boys, which premiered there in 2005. In 1999, Pesci announced his retirement from acting to pursue a musical career and to enjoy life away from the camera. He returned to acting when he did a cameo in De Niro’s 2006 film The Good Shepherd. He starred in the 2010 brothel drama Love Ranch, alongside Helen Mirren.

Pesci appeared with Don Rickles in a 2011 Snickers advertisement in which he portrays the alter-ego of a young man who attends a party and becomes agitated by two women.”⁹

⁹ https://en.wikipedia.org/wiki/Joe_Pesci

In the excerpt in Fig 3.4, it can be observed that the year 1980 could be a deviation, but the year 1992 appears close to this, which doesn't make it a huge deviation, thereby marking 1980 as a relevant year. Though the line suggests a reference, having another year close by still marks the year as relevant. There is always such a possibility and in such cases we give the year the benefit of the doubt.

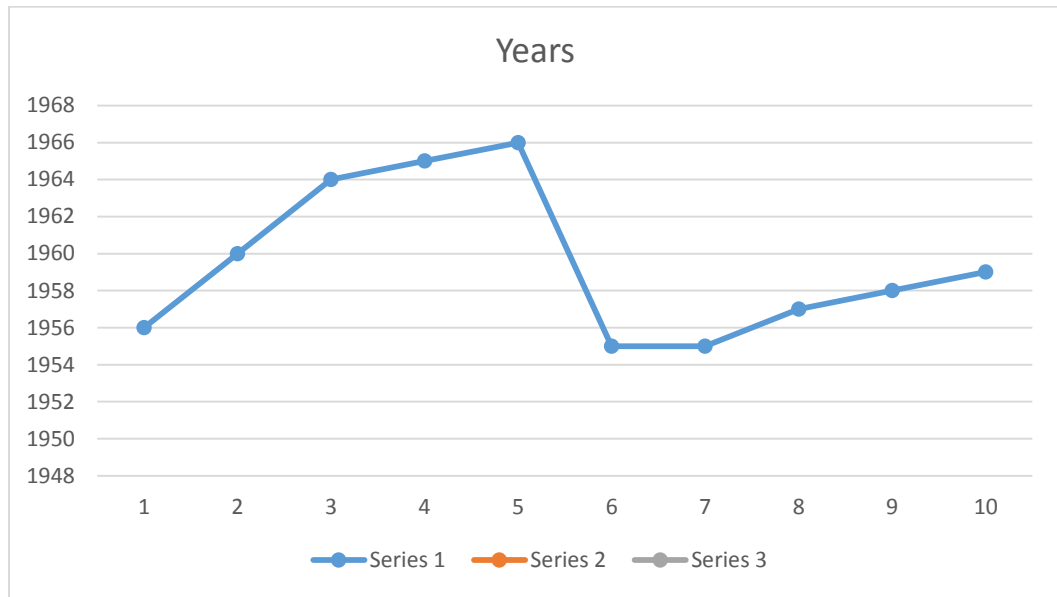


Figure 3.5 Graph showing the years appearing in an excerpt from the Wikipedia document 'Martin Lurther King, Jr.'

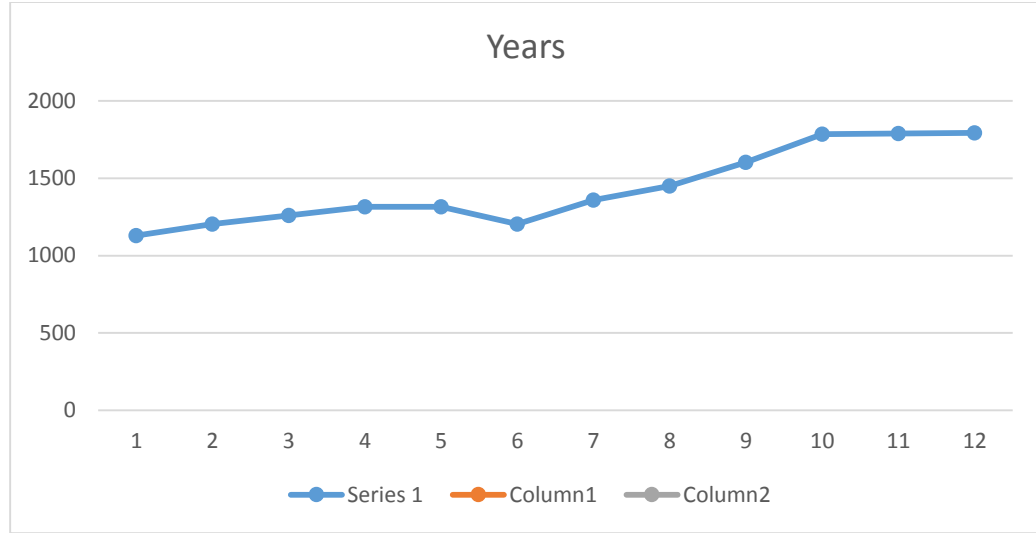


Figure 3.6 Graph showing the years appearing in an excerpt from the Wikipedia document ‘Normandy’

In the Fig 3.5, the year value suddenly deviates but continues with temporal values closer to the deviated value. Such cases should also be considered in identifying the *spike*. This will be done by ignoring deviations that occur on the boundary of the windows, because the values falling in the boundary will be considered again in the next iteration. In Fig 3.5, the year 1955 will not be considered a *spike*, as the sentences appearing after the sentence with 1955 are closer to year 1955. In Fig 3.6, values falling over a thousand years are shown. The algorithm should also support such huge range of values to correctly identify *spikes*.

To identify a *spike* in the sentences of the document, we consider a fixed size moving window which scans through the sentences and plots years as they appear. The window size, which is the number of sentences considered at a time, should be a minimum of 3 to identify a *spike* and we observed better results for smaller window sizes (3-5). In our implementation, we set the window size at 5. The values on the boundary of the window are not to be considered as they might be related to values outside the window.

We use scaling/normalizing to determine *spikes* in the temporal values, based on the chronological order in which narrative documents are written. Based on scaling/normalizing the values that occur within each window, we determine the importance of the temporal values. The values are scaled to range between 0 and 1.

$$y' = \frac{y - \min(Y)}{\max(Y) - \min(Y)}$$

y' = normalized value
 y = current year
 Y = Set of years within the window

After the values within a window are normalized based on the above formula, we analyze the values to see if all the years are having normalized values above 0.5, other than the minimum value of the window. In such a case, eliminate the minimum value as anomalies. Here, 0.5 is considered to give equal importance initially to the latest and earliest year.

Consider the excerpt for Fig 3.3. The years from the sentences, which appear in a single window of size 5 are 1994,1999,1998,1975 and 2000. The normalized value for each year calculated will be-

Max year – 2000, Min year – 1975

1994: $1994 - 1975 / 2000 - 1975 = 0.76$

1999: $1999 - 1975 / 2000 - 1975 = 0.96$

1998: $1998 - 1975 / 2000 - 1975 = 0.92$

1975: $1975-1975/2000-1975 = 0$

2000: $2000-1975/2000-1975 = 1$

As, all the values are observed to be over 0.5, except for the earliest year, the earliest year 1975 is not considered for the snippet in this case.



Figure 3.0.7 Normalized value of years in a window

3.4 Sentence ranking and selection criteria

After filtering the sentences based on the above steps, we obtain only the sentences that are relevant to the subject of the document. The final step is about effectively selecting the best sentences from the subset, which are temporally interesting and summarize the document content. At this point, the time line t_L contains a set of relevant temporal sentences S . A good summary usually contains the most important points from a document. When we consider the temporal aspect of an entity, on a timeline it has a first occurrence and a last occurrence which could be equally important. In our approach, we generate a summary with four sentences – one sentence with the earliest time reference within the document which we refer to as an earliest event, two sentences which relate to the most talked about time within the document which we refer as most important events and one sentence which signifies the latest event mentioned in the document which we refer as latest event.

Latest Event: Svore, K. M. et al. [10], demonstrates by crowd sourcing, interest shown by people in finding the most recent events in a document, especially when the topic is trending. In our approach too, we add one line which is the latest event mentioned about the subject in the document. We make use of the already available set of sentences S , which are tagged with the temporal values that appear in each of the sentences. The sentence with the most recent temporal value will be chosen, s_L .

Most Important Event: In the next part, we plan to append sentences which talk about the most important events relating to the subject. As the resultant sentences obtained after filtering the original document already contain sentences relevant to the subject of the document, we only consider the temporal values of the sentences to determine the most important event. One approach to determining the most important year is the term frequency $f_{(t,d)}$ of a year in a document d . All the temporal values of sentences will be plotted on a normal distribution graph to determine the most important year of the document. Assuming the year with the highest $f_{(t,d)}$ to be most important [4], we propose to add an additional parameter to $f_{(t,d)}$ denoted by λ . The λ value will be the number of sentences without an explicit temporal expression which occur in between two sentences which have an explicit temporal expression with the same year values. This parameter will only be considered if no other temporal value exists between the sentences with the same year. On the basis of studying various examples, the reason for selecting these sentences which do not contain any temporal values is that there is a higher probability that these sentences in between are related to the same year as the sentences on the boundary of this set. For example consider the excerpt –

“After reading the January 1975 issue of *Popular Electronics* that demonstrated the Altair 8800, Gates contacted Micro Instrumentation and Telemetry Systems (MITS), the creators of the new microcomputer, to inform them that he and others were working on a BASIC interpreter for the platform.^[43] In reality, Gates and Allen did not have an Altair and had not written code for it; they merely wanted to gauge MITS's interest. MITS president Ed Roberts agreed to meet them for a demo, and over the course of a few weeks they developed an Altair emulator that ran on a minicomputer, and then the BASIC interpreter. The demonstration, held at MITS's offices in Albuquerque, was a success and resulted in a deal with MITS to distribute the interpreter as Altair BASIC. Paul Allen was hired into MITS,^[44] and Gates took a leave of absence from Harvard to work with Allen at MITS in Albuquerque in November 1975.”¹⁰

It can be observed in the excerpt that all the sentences between the first and last sentence also occurred during the same year, 1975. This is prominent in documents classified as narratives. In the above example, the number of sentences with year 1975 - $f_{(1975,d)}$ is 2, and from our approach it will be $f_{(1975,d)} + \lambda = 2+3 = 5$, as there are 3 sentences in between, which are not explicitly identified as happened in 1975, but they did occur in 1975 as they are in the same context as the two sentences with explicit mention of 1975. This score is calculated for the years of all the temporal values present in the document d . After finding the year with the highest score, starting from the top of the document we select the first two sentences in which that year appears. We give preferences to sentences appearing higher up because good narrative documents contain a summary of the document at the top and there is a higher chance the year and event is mentioned by the author owing to its high occurrence in the document.

¹⁰ https://en.wikipedia.org/?title=Bill_Gates

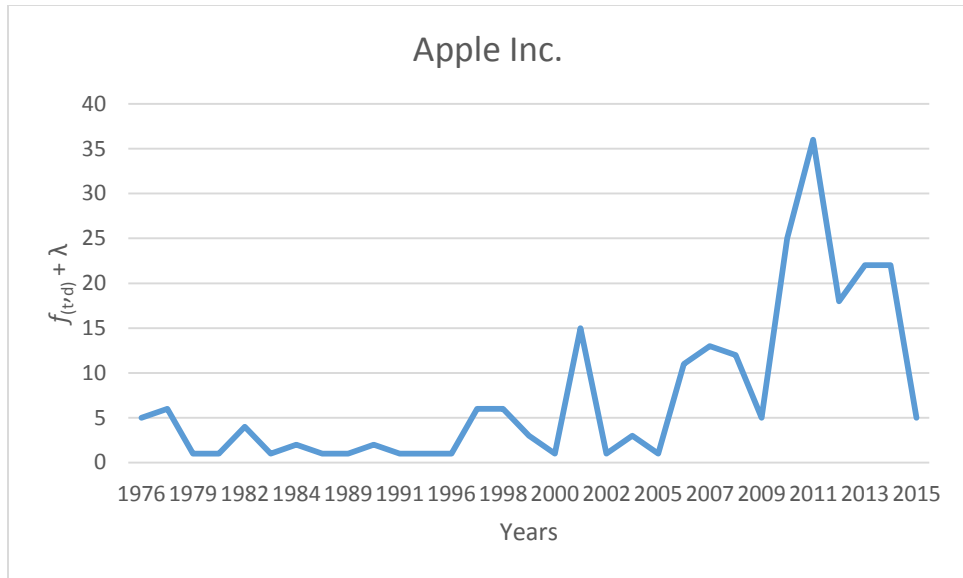


Figure 3.8 Calculated $f(t,d) + \lambda$ values for the wiki document Apple Inc.

Earliest Event: Finally, one sentence relating to the earliest year mentioned in the document, which would relate to the subjects inception, is also added to the snippet to complete the summarization.

Implementation and Evaluation

This section covers the topics related to implementation of our approach discussed in Section 3. It also includes the evaluation of our approach using crowd sourcing. The final section contains detailed discussions on the results of the evaluation process.

4.1 Implementation

The implementation section details the various steps involved in generating a temporal biased snippet with detailed explanation and pseudo code at every step.

4.1.1 Temporal tagging algorithm

We use Heidel Time version available as standalone [11] for tagging temporal data in the documents. These temporal tagged documents will be the basis for further processing in our implementation. This standalone version requires a compatible pre-processing tool which can identify parts of speech and sentence boundaries. For this purpose, TreeTagger [13] which is available as parameter files and accepted by Heidel Time is used. TreeTagger is found to perform better than Markov Model based taggers.

This standalone version is integrated into our approach, automating the creation of temporally tagged documents from the original documents. Fig 4.1 shows the

algorithm with pseudo code used for tagging the documents using Heidel Time and its integration into our implementation.

```
1  Call method InitializeComponent
2
3  Set startPath to Path.GetDirectoryName with Path.GetDirectoryName Environment.CurrentDirectory
4
5  Set docsPath to @"C:"
6
7  Set command to " java -jar "
8
9  Set jarname to @"\\heideltime-standalone-1.8\\de.unihd.dbs.heideltime.standalone.jar "
10
```

Figure 4.1 Pseudo code for Heidel Time integration

```
1  Create new Process
2
3  Set process.StartInfo.FileName to "cmd.exe"
4
5  Set process.StartInfo.Arguments to arguments
6
7  Set process.StartInfo.WorkingDirectory to docsPath plus @"\\heideltime-standalone-1.8\\";
8
9  Set process.StartInfo.RedirectStandardOutput to true
10
11 Call method process.Start
12
13 Initialise reader to process.StandardOutput
14
15 Initialise output to reader.ReadToEnd
16
17 Call method Console.WriteLine with output
18
19 Call method process.WaitForExit
20
21 Call method process.Close
22
```

Figure 4.2 Pseudo code for creating temporal tagged documents using Heidel Time

A new process thread is created to call the standalone Heidel Time application, sending the documents path as input. The temporal tagged documents are stored in a separate path. Shown below is an example of how the temporal tagged documents look—

“.....Apple was founded by Steve Jobs, Steve Wozniak, and Ronald Wayne on <TIMEX3 tid="t3" type="DATE" value="1976-04-01">April 1, 1976</TIMEX3>, to develop and sell personal computers.[5] It was incorporated as Apple Computer, Inc. on <TIMEX3 tid="t9" type="DATE" value="1977-01-03">January 3, 1977</TIMEX3>, and was renamed as Apple Inc. on <TIMEX3 tid="t10" type="DATE" value="2007-01-09">January 9, 2007</TIMEX3>, to reflect its shifted focus towards consumer electronics. Apple (NASDAQ:AAPL) joined the Dow Jones Industrial Average on <TIMEX3 tid="t15" type="DATE" value="2015-03-19">March 19, 2015</TIMEX3>.[6].....”

All sentences with temporal expressions are identified using regular expressions and extracted to a data table containing the actual sentence and its temporal value.

```

1  Foreach string in sentences
2      Lineno++;
3      If line.Contains with "TIMEX3" and line.Contains with "type=\"DATE\""
4          writeline = Regex.Replace(line, "<TIMEX3 .*?>", "");
5          Set writeline to writeline.Replace with "<!DOCTYPE TimeML SYSTEM \"TimeML.dtd\">", ""
6          Set writeline to writeline.Replace with "<TimeML>", ""
7          Set writeline to writeline.Replace with "</TimeML>", ""
8          Set pos to writeline.IndexOf with "type=\"DATE\" value=\""
9          aa = writeline.Substring(pos+19, 4);
10         Set year to Convert.ToInt32 with writeline.Substring pos plus 19, 4
11         Initialise aaa to writeline.Substring with pos plus 23
12         If writeline.Substring with pos plus 23, 1
13             Try
14                 Set month to Convert.ToInt32 with writeline.Substring pos plus 24, 2
15             Catch Exception ee
16                 continue;
17             EndTry
18         EndIf
19         If writeline.Substring with pos plus 26, 1
20             Try
21                 Set day to Convert.ToInt32 with writeline.Substring pos plus 27, 2
22             Catch Exception ee
23                 continue;
24             EndTry
25         EndIf
26         Call method dt.Rows.Add with Path.GetFileName file, writeline, year... "Consider"
27         Set year to 0
28         Set month to 0
29         Set day to 0
30     EndIf
31     sw.WriteLine(writeline);
32 EndIf
33 EndForeach
34

```

Figure 4.3 Pseudo code for extracting sentences with temporal expressions and their temporal value

Every temporal expression's value is available in the 'value' attribute of the TIMEX3 tag. We use one additional column in the data table labeled 'Status' which can contain either 'Consider' or 'Ignore'. The importance of this value will be discussed in Sections 4.1.3 and 4.1.4. After extracting all the sentences with temporal expressions, the temporal tags are cleaned up to maintain text free of any tags for using them in the snippet.

4.1.2 Algorithm for Contextually Irrelevant Expressions Elimination

We use a set of keywords to identify sentences with temporal expressions which contain a temporal value that do not relate to a point in time, as discussed in section 3.2. We maintain the keywords in a file and utilize it at the time of the snippet creation. Each keyword in the file is searched in the sentence list and all hits are marked as 'Ignore' in their 'Status' column. These post processing rules are important to obtain sentences that are considered relevant to the subject of the document. Fig 4.4 shows the pseudo code for identifying these contextually irrelevant temporal expressions in sentences and marking their Status as 'Ignore'.

```

1 String ambiguousKeys;
2   using (StreamReader srStopList = new StreamReader("AmbiguousList.txt"))
3     Set ambiguousKeys to srStopList.ReadToEnd
4   EndStatement
5   Initialise stopWordList to ambiguousKeys.Split with ';'
6   Foreach DataRow in dt.Rows
7     For i is 0, i is less than ambiguousKeys.Length, i increments by 1
8       If position 1 in dr
9         Set 3 of dr to "Ignore"
10      EndIf
11    EndFor
12  EndForeach

```

Figure 4.4 Pseudo code for identifying contextually irrelevant temporal expressions

4.1.3 Algorithm for Elimination of Sentences Irrelevant to Search Terms

Contextually irrelevant sentences are identified based on the sentence's temporal value. Each sentence's temporal value is compared with the temporal value of sentences surrounding it. As discussed in Section 3.4, we use a moving window which reads 5 sentences in each iteration. The 5 temporal values of the sentences are rescaled in the range 0 to 1 and compared to check if a *spike* exists. Fig 4.5 shows the pseudo code which reads the temporal values of sentences in sequence as they appear in the document, identifies the highest and lowest year in each iteration and rescales the values. Pseudo code for identifying lowest and highest year in each iteration is shown in Fig 4.6 and Fig 4.7.

```

1 For i is 0, i is less than dt.Rows.Count minus 5, i increments by 1
2   If not position in dt.Rows
3     Set i to i plus 4
4   EndIf
5   Set year1 to Convert.ToInt32 with position i in dt.Rows
6   Set year2 to Convert.ToInt32 with position in dt.Rows
7   Set year3 to Convert.ToInt32 with position in dt.Rows
8   Set year4 to Convert.ToInt32 with position in dt.Rows
9   Set year5 to Convert.ToInt32 with position in dt.Rows
10  Set yearmarker to 0
11  Set lowestyear to lowest with year1, year2, year3...
12  Set highestyear to highest with year1, year2, year3...
13  If year1 is equal to lowestyear or year5 is equal to lowestyear
14    Else
15      Set syear1 to 1.0 multiplied by ( year1 minus lowestyear ) divided by ( highestyear minus lowestyear )
16      Set syear2 to 1.0 multiplied by ( year2 minus lowestyear ) divided by ( highestyear minus lowestyear )
17      Set syear3 to 1.0 multiplied by ( year3 minus lowestyear ) divided by ( highestyear minus lowestyear )
18      Set syear4 to 1.0 multiplied by ( year4 minus lowestyear ) divided by ( highestyear minus lowestyear )
19      Set syear5 to 1.0 multiplied by ( year5 minus lowestyear ) divided by ( highestyear minus lowestyear )
20      Set ignoreLowest to checkvalues with syear1, syear2, syear3...
21      If ignoreLowest
22        Set i plus yearmarker minus 1 of dt.Rows to
23      EndIf
24    EndIf
25 EndFor

```

Figure 4.5 Pseudo code for rescaling temporal values of sentences

```

1 If year1 is less than or equal to year2 and year1 is less than or equal to year3 and year1 is less than or equal to year4 and year1 is less than or equal to year5
2   Set yearmarker to 1
3   Return year1
4 Else if year5 is less than or equal to year2 and year5 is less than or equal to year3 and year5 is less than or equal to year4
5   Set yearmarker to 5
6   Return year5
7 Else if year2 is less than or equal to year3 and year2 is less than or equal to year4
8   Set yearmarker to 2
9   Return year2
10 Else if year3 is less than or equal to year4
11   Set yearmarker to 3
12   Return year3
13 Else
14   Set yearmarker to 4
15   Return year4
16 EndIf

```

Figure 4.6 Pseudo code for identifying the lowest temporal value

```

1 If year1 is greater than or equal to year2 and year1 is greater than or equal to year3 and year1 is greater than or equal to year4 and year1 is greater than or equal to year5
2   Return year1
3 Else if year5 is greater than or equal to year2 and year5 is greater than or equal to year3 and year5 is greater than or equal to year4
4   Return year5
5 Else if year2 is greater than or equal to year3 and year2 is greater than or equal to year4
6   Return year2
7 Else if year3 is greater than or equal to year4
8   Return year3
9 Else
10   Return year4
11 EndIf

```

Figure 4.7 Pseudo code for identifying the highest temporal value

We set the threshold as 0.5, to signify equal importance of the lowest and the highest year in each iteration. The status of the sentence with lowest rescaled temporal

value in each iteration is set to 'Ignore' if – (i) It is not the first or last sentence in the set and (ii) the rescaled temporal values of all the other sentences is over 0.5 in the set, which signifies that all years occur around a close range except the lowest year. The lowest year sentence here is not considered for snippet. Set refers to the 5 sentences considered in each iteration. All the sentences with status marked as 'Ignore' are not considered for the final snippet. Fig 4.8 shows the pseudo code for comparing the rescaled temporal values in the set and checking if the values are over or below 0.5.

```
1  Initialise count to 0
2  If syear1 is greater than 0.5
3      Count++;
4  EndIf
5  If syear2 is greater than 0.5
6      Count++;
7  EndIf
8  If syear3 is greater than 0.5
9      Count++;
10 EndIf
11 If syear4 is greater than 0.5
12     Count++;
13 EndIf
14 If syear5 is greater than 0.5
15     Count++;
16 EndIf
17 If count is equal to 4
18     Return true
19 Else
20     Return false
21 EndIf
```

Figure 4.8 Pseudo code to identifying a spike in temporal values

4.1.4 Algorithm for Sentence ranking and summarizing

The final step in our approach is the actual creation of snippet. As discussed in Section 3.4, the snippet contains a combination of four sentences- one sentence with a temporal expression which appears earliest on the timeline, two sentences considered to be the most important event on the timeline and one sentence with temporal expressions that appears as the latest event on the timeline. These are selected from the set of sentences with status not marked as 'Ignore' from the previous steps.

The two sentences considered to be the most important event are calculated based on the number of sentences with the same year value in their temporal value. Here, in addition to sentences with temporal values, sentences which do not have an explicit temporal value but occur in between two sentences with the same year in their temporal value are considered to have the same year value as the two sentences (Value of $f_{(Year,d)}$ + λ , discussed in section 3.4). Fig 4.9 shows the pseudo code for calculating the number of sentences under each year mentioned in the document.

```

1 Set dataGridView1.DataSource to dt
2 Create new DataTable
3 Call method dtyears.Columns.Add with "Document"
4 Call method dtyears.Columns.Add with "Year"
5 Call method dtyears.Columns.Add with "Count"
6 For i is 0, i is less than dt.Rows.Count minus 1, i increments by 1
7   If position i in dt.Rows
8     If Convert.ToInt32 with position i in dt.Rows is equal to Convert.ToInt32 with position in dt.Rows
9       Call method dtyears.Rows.Add with position i in dt.Rows, position i in dt.Rows, Convert.ToInt32 position in dt.Rows...
10      minus Convert.ToInt32 with position i in dt.Rows
11     Else
12       Call method dtyears.Rows.Add with position i in dt.Rows, position i in dt.Rows, 1
13     EndIf
14   EndIf
15 EndFor
16 Set dataGridView2.DataSource to dtyears
17 Initialise newSort to from row
18 Create new DataTable
19 Call method dtYearGroup.Columns.Add with "Document"
20 Call method dtYearGroup.Columns.Add with "Year"
21 Call method dtYearGroup.Columns.Add with "Count"
22 Foreach var in newSort
23   Call method dtYearGroup.Rows.Add with v.Document, v.Year, v.Sum
24 EndForeach
25 Set dataGridView3.DataSource to dtYearGroup
26 Call method luceneIndexSS
27 Set dtYearGroup.TableName to "yearsgroup"

```

Figure 4.9 Pseudo code to calculate number sentences for each year appearing in a document

The sentences with the earliest and the latest years in the document, along with the most important year are selected. In cases where multiple sentences with the same year are in the document, the sentence that appears higher in the document is selected. This is based on our observation that in several narrative documents, an introductory paragraph is present and writers mostly tend to mention the most important aspects of the document here. Fig 4.10 shows the pseudo code for selecting the sentences based on their temporal value and order of appearance in the document. These selected sentences form the snippet.

```
1  Foreach DataRow in tempFiles.Rows
2      Set documentid to position 0 in dr
3      Set tempTitle to position 1 in dr
4      Set tempContent to position 4 in dr
5      Set orgTitle to tempTitle
6      Set orgContent to tempContent
7      Initialise minyear to Convert.ToInt32 with dtYearGroup.AsEnumerable
8      Initialise maxyear to Convert.ToInt32 with dtYearGroup.AsEnumerable
9      Initialise chk to from r
10     Initialise impyear to 0
11     Foreach var in chk
12         Set impyear to Convert.ToInt32 with v.Year
13         break;
14     EndForeach
15     Initialise snipOld to ""
16     Initialise sold to from row
17     Foreach var in sold
18         Set snipOld to v.snipOld.ToString
19         break;
20     EndForeach
21     Set snipOld to clearsnp with snipOld
22     Initialise snew to from row
23     Foreach var in snew
24         Set snipNew to v.snipNew.ToString
25         break;
26     EndForeach
27     Set snipNew to clearsnp with snipNew
28     Initialise simp to from row
29     Initialise scout to 0
30     Foreach var in simp
31         Set snipImp to snipImp plus v.snipImp.ToString plus "..."
32         Scout++;
33         If scout is equal to 2
34             break;
35         EndIf
36     EndForeach
37     Set snipImp to clearsnp with snipImp
38     Set orgContent to snipOld plus "..." plus snipImp plus snipNew
39
```

Figure 4.10 Pseudo code to generate temporal biased snippet

4.2 Evaluation

We evaluate our approach using the weighted Kappa [35] measure. The Kappa measure without weights considers all disagreements equally. Using weights, disagreements of varying degree can affect the scores. So, the weight would be higher for a higher disagreement.

We consider three judgement decisions for evaluating the quality of our proposed snippet – ‘Good Snippet’, ‘Acceptable Snippet’ and ‘Not Good Snippet’. We choose weights in a way whenever there is an agreement on ‘Not Good Snippet’ for a snippet, we penalize the score and when there is an agreement on ‘Good Snippet’ we make it a positive impact. As shown in Table 4.1, a weight of zero is given for agreements on ‘Good Snippet’ and a weight of 1 is given for agreements on ‘Not Good Snippet’.

	Good Snippet	Acceptable Snippet	Not Good Snippet
Good Snippet	0	1	2
Acceptable Snippet	1	0	1
Not Good Snippet	2	1	1

Table 4.1 Weights for Kappa Measure

We calculate Weighted Kappa Measure using –

$$\kappa_w = 1 - \frac{\sum_{i,j} w_{ij} p_{ij}}{\sum_{i,j} w_{ij} e_{ij}}$$

w_{ij} – Weights

p_{ij} – Observed probabilities

e_{ij} - Expected probabilities

	Judge A				
Judge B		Good Snippet	Acceptable Snippet	Not Good Snippet	
	Good Snippet	0.433	0.066	0	0.499
	Acceptable Snippet	0.033	0.2	0.1	0.333
	Not Good Snippet	0	0.1	0.06	0.16
		0.466	0.366	0.16	1

Table 4.2 Observed probabilities

	Judge A				
Judge B		Good Snippet	Acceptable Snippet	Not Good Snippet	
	Good Snippet	0.233	0.183	0.083	
	Acceptable Snippet	0.155	0.122	0.055	
	Not Good Snippet	0.077	0.016	0.027	

Table 4.3 Expected probabilities

Table 4.2 shows the judgements(observed probabilities) given by Judge A and Judge B based on comparing 30 search results snippets generated with our approach against current existing search systems for the same queries. The queries are listed in Appendix A. Table 4.3 shows the expected probabilities calculated based on observed probabilities. Consider the expected probability for both Judge A and Judge B on 'Good Snippet', it is calculated by multiplying the observed value that Judge A marks a snippet as 'Good Snippet' which is 0.466 as seen in Table 4.3, and the observed value that Judge B marks a snippet as 'Good Snippet' which is 0.499. Multiplying these two values, we get

0.233 which is the expected probability. All other expected probabilities are calculated using this procedure. Table 4.4 shows the actual observed values, the judgements made by the judges. Of the 30 queries considered for evaluation, both the judges marked 13 as good snippets and 9 as good and acceptable snippets created using our approach. The final Weighted Kappa Measure value K_w is **0.53**. Landis et al. gave various labels for scores and a score >0.8 is considered **almost perfect** and between 0.4 and 0.6 is considered as **moderate**.

	Judge A				
Judge B		Good Snippet	Acceptable Snippet	Not Good Snippet	
	Good Snippet	13	2	0	15
	Acceptable Snippet	1	6	3	10
	Not Good Snippet	0	3	2	5
		14	11	5	30

Table 4.4 Observed Values

Conclusion and Future Work

In this thesis, we presented an alternate approach for generating search result snippets for general search terms. We used temporal data present in the pages to introduce a novel approach to summarize the page contents. We implemented algorithms for removing noise in the data. We compared our results to how existing search engines display search results snippets for general search terms and our approach has a Weighted Kappa Measure of 0.53. Based on the interpretation of Weighted Kappa Measure values given by Landis et al., a score >0.8 is almost perfect and between 0.4 and 0.6 is considered as moderate. Some of the limitations observed are – (i) sometimes years earlier than the subject's existence tend to appear in the snippet, (ii) sometimes a sentences which refers a previous sentence appears in the snippet, and (iii) some temporal values which do not relate to a point in time, for example software versions like 'Visual Studio 2013', are hard to identify with rules. To address these issues, complex natural language processing techniques would be required, using which could increase the Weighted Kappa Measure.

In future, we will explore the possibility of tagging every sentence in a page with a temporal value based on its context, even when there are no explicit or relative references to temporal values in the sentence, for a more accurate creation of the timeline. We also plan to extend our approach for generating answers for questions with temporal aspect. For example, "What are Obama's achievements in 2014". We will also explore creating meaningful partial sentences in the sentences for compactness.

Bibliography

- [1] Manning, C., & Raghavan, P. (2008). Preface. In *Introduction to information retrieval*. New York: Cambridge University Press.
- [2] Alonso, O., Strötgen, J., Baeza-Yates, R. A., & Gertz, M. (2011). Temporal Information Retrieval: Challenges and Opportunities. *TWAW*, 11, 1-8.
- [3] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 832-843.
- [4] Alonso, O., Baeza-Yates, R., & Gertz, M. (2009). Effectiveness of temporal snippets. In *WSSP Workshop at the World Wide Web Conference—WWW* (Vol. 9).
- [5] Chang, A. X., & Manning, C. D. (2012, May). SUTime: A library for recognizing and normalizing time expressions. In *LREC* (pp. 3735-3740).
- [6] UzZaman, N., & Allen, J. F. (2010). Event and temporal expression extraction from raw text: First step towards a temporally aware system. *International Journal of Semantic Computing*, 4(04), 487-508.
- [7] Strötgen, J., & Gertz, M. (2010, July). HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation* (pp. 321-324). Association for Computational Linguistics.
- [8] Strötgen, J., Gertz, M., & Popov, P. (2010, February). Extraction and exploration of spatio-temporal information in documents. In *Proceedings of the 6th Workshop on Geographic Information Retrieval* (p. 16). ACM.
- [9] UzZaman, N., & Allen, J. F. (2010, July). TRIPS and TRIOS system for TempEval-2: Extracting temporal information from text. In *Proceedings of the 5th International Workshop on Semantic Evaluation* (pp. 276-283). Association for Computational Linguistics.
- [10] Svore, K. M., Teevan, J., Dumais, S. T., & Kulkarni, A. (2012, August). Creating temporally dynamic web search snippets. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval* (pp. 1045-1046). ACM.
- [11] Zell, J., Fay, A. & Strötgen, J. (2013). HeidelTime Standalone Version Manual
- [12] Strötgen, J., & Gertz, M. (2010, July). HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation* (pp. 321-324). Association for Computational Linguistics.
- [13] Schmid, H. (1994, September). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the international conference on new methods in language processing* (Vol. 12, pp. 44-49).
- [14] Cucerzan, S. P., & Richardson, M. R. (2009). *U.S. Patent No. 7,512,601*. Washington, DC: U.S. Patent and Trademark Office.
- [15] Nygaard, V. R., Turchetto, R., Chan, J. M. Y., Biemann, C., Ahn, D. D., Burbank, A. R., ... & King, T. H. (2014). *U.S. Patent No. 8,788,260*. Washington, DC: U.S. Patent and Trademark Office
- [16] Verstak, A. A., & Acharya, A. (2012). *U.S. Patent No. 8,145,617*. Washington, DC: U.S. Patent and Trademark Office.
- [17] Tombros, A., & Sanderson, M. (1998, August). Advantages of query biased summaries in information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 2-10). ACM.
- [18] Singhal, A. (2012). Introducing the knowledge graph: Things, not strings. Official Google Blog. Retrieved November 7, 2013.

- [19] Turpin, A., Tsegay, Y., Hawking, D., & Williams, H. E. (2007, July). Fast generation of result snippets in web search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 127-134). ACM.
- [20] Goldstein, J., Kantrowitz, M., Mittal, V., & Carbonell, J. (1999, August). Summarizing text documents: sentence selection and evaluation metrics. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 121-128). ACM.
- [21] Campos, R., Dias, G., & Jorge, A. M. (2011, March). What is the Temporal Value of Web Snippets?. In *TWAW* (pp. 9-16).
- [22] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E. R., & Mitchell, T. M. (2010, July). Toward an Architecture for Never-Ending Language Learning. In *AAAI* (Vol. 5, p. 3).
- [23] Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., & Etzioni, O. (2007, January). Open information extraction for the web. In *IJCAI* (Vol. 7, pp. 2670-2676).
- [24] West, R., Gabrilovich, E., Murphy, K., Sun, S., Gupta, R., & Lin, D. (2014, April). Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web* (pp. 515-526). ACM.
- [25] Meyers, P. (n.d.). Knowledge Graph 2.0: Now Featuring Your Knowledge. Retrieved July 3, 2015.
- [26] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008, June). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (pp. 1247-1250). ACM.
- [27] Suchanek, F. M., Kasneci, G., & Weikum, G. (2007, May). Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web* (pp. 697-706). ACM.
- [28] Kuzey, E., & Weikum, G. (2012, April). Extraction of temporal facts and events from Wikipedia. In *Proceedings of the 2nd Temporal Web Analytics Workshop* (pp. 25-32). ACM.
- [29] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). *Dbpedia: A nucleus for a web of open data* (pp. 722-735). Springer Berlin Heidelberg.
- [30] Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A. M., Shaked, T., ... & Yates, A. (2004, May). Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of the 13th international conference on World Wide Web* (pp. 100-110). ACM.
- [31] Weld, D. S., Wu, F., Adar, E., Amershi, S., Fogarty, J., Hoffmann, R., ... & Skinner, M. (2008, July). Intelligence in Wikipedia. In *AAAI* (Vol. 8, pp. 1609-1614).
- [32] Adar, E., Teevan, J., & Dumais, S. T. (2009, April). Resonance on the web: web dynamics and revisitation patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1381-1390). ACM.
- [33] Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2), 159-165.
- [34] White, R. W., Ruthven, I., & Jose, J. M. (2002, August). Finding relevant documents using top ranking sentences: an evaluation of two alternative schemes. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 57-64). ACM.
- [35] Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4), 213.
- [36] Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, 159-174.

Appendix A

List of Search Queries

Joe Pesci
apple
fifa
baltimore
new york
obama
wright state
tokyo
bush
formula one
gandhi
nasa
olympics
oxford
plymouth
Super bowl
us economy
vietnam war
volkswagen group
mototr speedway
industrial revolution
ford
ohio university
microsoft
ibm
usps
walmart
wimbledon
issac newton