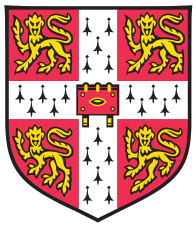
Content Selection for Timeline Generation from Single History Articles

Sandro Mario Bauer



University of Cambridge
Computer Laboratory
St John's College

Supervisors:
Dr Simone Teufel
Dr Stephen Clark

August 2017

This dissertation is submitted for
the degree of Doctor of Philosophy

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text.
It does not exceed 63,000 words in length.

Content Selection for Timeline Generation from Single History Articles

Sandro Mario Bauer

Summary

This thesis investigates the problem of content selection for timeline generation from single history articles. While the task of timeline generation has been addressed before, most previous approaches assume the existence of a large corpus of history articles from the same era. They exploit the fact that salient information is likely to be mentioned multiple times in such corpora. However, large resources of this kind are only available for historical events that happened in the most recent decades. In this thesis, I present approaches which can be used to create history timelines for *any* historical period, even for eras such as the Middle Ages, for which no large corpora of supplementary text exist.

The thesis first presents a system that selects relevant historical figures in a given article, a task which is substantially easier than full timeline generation. I show that a supervised approach which uses linguistic, structural and semantic features outperforms a competitive baseline on this task. Based on the observations made in this initial study, I then develop approaches for timeline generation. I find that an unsupervised approach that takes into account the article's subject area outperforms several supervised and unsupervised baselines.

A main focus of this thesis is the development of evaluation methodologies and resources, as no suitable corpora existed when work began. For the initial experiment on important historical figures, I construct a corpus of existing timelines and textual articles, and devise a method for evaluating algorithms based on this resource. For timeline generation, I present a comprehensive evaluation methodology which is based on the interpretation of the task as a special form of single-document summarisation. This methodology scores algorithms based on meaning units rather than surface similarity. Unlike previous semantic-units-based evaluation methods for summarisation, my evaluation method does not require any manual annotation of system timelines. Once an evaluation resource has been created, which involves only annotation of the input texts, new timeline generation algorithms can be tested at no cost. This crucial advantage should make my new evaluation methodology attractive for the evaluation of general single-document summaries beyond timelines. I also present an evaluation resource which is based on this methodology. It was constructed using gold-standard timelines elicited from 30 human timeline writers, and has been made publicly available.

This thesis concentrates on the content selection stage of timeline generation, and leaves the surface realisation step for future work. However, my evaluation methodology is designed in such a way that it can in principle also quantify the degree to which surface realisation is successful.

Acknowledgements

This dissertation owes its existence to the invaluable support of a large number of people. First and foremost, I would like to thank my supervisors Simone Teufel and Stephen Clark. Without their constant support and guidance, the present dissertation would not have come into being. I am grateful to them for teaching me how to plan, conduct and publish my research, as well as for their support on a personal level. The self-baked loaves of bread and the board game evenings will be remembered! I would also like to thank my examiners Ann Copestake and Advaith Siddharthan for their highly valuable feedback and a very enjoyable and constructive viva.

I was fortunate to meet many very good friends at St John's College. Their presence has made my time in Cambridge a very special one. In particular, I would like to thank (in no particular order) my close friends Jean Maillard, Johannes Bausch and Giovanni Varelli, who have provided invaluable personal support year in, year out. Without them, this thesis would not exist. I would like to thank Ben Woodhams, Katarzyna Sokół, Panagiotis Barkas, Laura Keating, Mubeen Goolam, Natacha Crooks, Avital Rom, Tom Fieldman, Ana Llorens, Daphne Ezer, and many others which I have failed to mention here, for the many hours spent together. My thanks also go to all friends from other colleges, notably Shiva Mihan and Peter Sung Kyu Kim.

I would also like to acknowledge my colleagues in the Natural Language and Information Processing Group (and in the wider Cambridge NLP landscape) who supported me personally and academically. It was them who read paper drafts, gave valuable pointers to relevant literature and academic events, shared their rich experience with planning experiments, and who were always available for a chat or a coffee. In particular, I would like to thank Ekaterina Kochmar, Laura Rimell, Kevin Heffernan, Yiannos Stathopoulos, Meng Zhang, Helen Yannakoudakis, Yimai Fang, Simon Baker, Kris Cao, Tamara Polajnar, Adrian Scoică, Theodosia Togia, Diarmuid Ó Séaghdha, Awais Athar, Dain Kaplan and Yan Huang.

I am very grateful to those who contributed to the work carried out in this thesis. This includes 30 volunteers, who spent hours reading history articles and writing timelines, often despite looming paper deadlines. In particular, I would like to thank Ben Woodhams and Ana Llorens, who simply offered to do further work when other participants dropped out. A further significant contribution to this dissertation was made by Bamber Gascoigne, the well-known presenter of the TV series "University Challenge", who shared his database of historical events with me at no cost. I am also grateful to Microsoft Research, the Computer Laboratory, St John's College Cambridge and the Cambridge Philosophical Society, who all provided financial support for my studies.

My Cambridge experience would not be complete without the amazing time I spent at Microsoft Research in 2015, which has enriched me in so many ways. I thank Filip Radlinski and Ryen White for their excellent and very close supervision on a daily basis. They encouraged me to submit a paper to the World Wide Web Conference, and invested many hours in improving the draft as well as the final version. I would also like to acknowledge my former colleague Konstantina Christakopoulou.

A special thank you goes to Ann Copestake, who provided important advice at a crucial time. Finally, I would like to thank my family, in particular my mother Margit and my grandfather Ernst († 2016) for their incessant support and encouragement, which were of paramount importance at all times.

Contents

Chapter 1

Introduction

How to best study the history of a country or a field of science is a rather personal choice. Many would argue that textbooks, or even original sources, are the best way for developing an informed opinion about historical facts. While textbooks have the advantage of relating different historical events to each other, such texts are generally lengthy and therefore take considerable time to read and digest. This makes it hard to memorise the most important historical information contained in them.

Timelines are an alternative way of presenting history. They are, in essence, lists of dated events which may be presented in textual form or graphically. Timelines are shorter and easier to read than narrative texts, and can help learners to remember the key aspects of a longer textbook text. Graphical history timelines in particular are commonly used as learning aids in the classroom, and may be purchased from a wide range of shops (see Figure 1.1 for an example).

The usefulness of timelines in the offline world has, in recent years, led to the creation of an electronic counterpart. In fact, timelines have become a preferred way of presenting history in a variety of online applications. This includes large-scale, hand-curated encyclopedias such as Microsoft's now discontinued Encarta; specialist efforts like HistoryWorld[1]; rather ad-hoc community-edited timelines on Wikipedia; and new interactive web applications such as Microsoft's ChronoZoom (see Figure 1.2 on page 15 for a screenshot). ChronoZoom allows the user to "zoom into" a historical period and hence to explore content at different levels of granularity.

Certain forms of timelines have also received interest in research. This mainly applies to a setting where a timeline must be constructed from a large number of news articles, which no single human could read on their own. Algorithms have been proposed which can identify the most important articles in such a collection. The events described in these articles can then be placed along a timeline.

For topics where large numbers of history articles are not available, large-scale manual efforts are still the solution of choice. Community efforts such as Wikipedia as well as enthusiasts such as Bamber Gascoigne[2] have compiled lists of thousands of events covering world history from prehistorical times to recent years.

The obvious problem is that writing timelines in this way is extremely tedious and time-consuming. This is reflected by the fact that many Wikipedia timelines are incomplete or extremely short; ChronoZoom only contains content for a small number of topics;

[1] `www.historyworld.com`

[2] Bamber Gascoigne is known for being the original presenter of the well-known British television programme "University Challenge".

Figure 1.1: History timeline used as a learning aid in classroom.

and the hand-curated event collection HistoryWorld, although of high quality, contains a mere 10,000 events for the entire human history. Also, such timelines have to be continuously updated, as new events happen and new evidence about earlier events becomes available. Since this is a laborious task, manually constructed timelines are often outdated in practice. The obvious need for event timelines of this kind and the high cost of writing them manually make it worthwhile to use natural language processing techniques for this purpose. The present thesis in particular investigates how timelines can be created from *single* history articles. Given an input text such as the Wikipedia article "History of Finland", the aim for a system is to identify important content that a human would include in a timeline on the history of Finland.

1.1 Properties of an ideal timeline

I will now delineate the desirable properties of history timelines created in this particular setting. Consider the extract of a hypothetical "ideal" history timeline describing the history of Finland (shown in Figure 1.3). What are the properties of such an ideal timeline?

Firstly, each timeline entry expresses a single *event*, i.e. an indisputable fact that took place in the real world. Each event is arguably *salient* in its own right. Intuitively, an event tends to be perceived as salient if it has precipitated a major change in the history of a concept. This may be the case if the event has ended a long-standing tradition, or if it sets in motion a series of important changes that have an impact well beyond the date when the event took place. A prime example of such an event is the Protestant Reformation, which put an end to the universal role of Catholicism; resulted in cataclysmic

Figure 1.2: Screenshot of ChronoZoom.

1550	Helsinki is founded.
around 1554	The Lutheran Church is established.
1596–1597	A peasant rebellion is suppressed brutally and bloodily in the Cudgel War.
1608	The Lutheran faith is made compulsory.
1696–1699	One third of the population dies in a famine.
1700–1721	The Great Northern War devastates Finland.
1809	Finland becomes a Grand Duchy in the Russian empire.
1899	The famous symphony *Finlandia* by Finnish composer Jean Sibelius is first performed.
1906	A Finnish national parliament is established.

Figure 1.3: Extract of an "ideal" timeline describing the history of Finland.

pan-European wars; and gave rise to a lasting transformation of the European political landscape, to name but a few. A good timeline contains events of various types, giving the reader a complete picture of history in a given time period. For instance, the ideal timeline in Figure 1.3 spans subject areas such as politics, religion, culture and society.

In addition to naming important events, a good timeline should mention all historical figures that play a central role in the time period in question (such as Jean Sibelius in Figure 1.3). This is important since human interpretation of world history often hinges on human actions, and, by extension, on key personalities and their interaction with the

world.

However, a good timeline is more than a list of individually salient events or historical figures. It is in fact a concise, short-hand, fact-oriented representation of the course of history, which can be used as a substitute for narrative texts describing the same historical evolutions much more verbosely. This property suggests that a timeline can be interpreted as a special form of a *summary*.

The difference to a standard summary is that timeline entries are independent of each other both conceptually and linguistically, whereas the sentences in a standard summary should form coherent text. Conceptual independence means that each event in the timeline is understandable in its own right, regardless of other entries or external information. For instance, events should not explicitly or implicitly be presented as a side aspect or consequence of another event. Conceptual independence naturally translates into linguistic independence: If a timeline entry is supposed to be interpretable on its own, it must of course not contain pronouns referring to a person not mentioned in that entry, or similar. Problems such as these could occur if a timeline is constructed by concatenating sentences from a history article, which takes them out of their original context.

A good timeline should also be free of redundant entries. I assume that two entries are redundant with regard to each other if they describe the same event or two closely related events. Such cases should be avoided since the reader will gain little additional information from redundant timeline entries. In contrast, a good timeline describes events that are clearly separate from each other.

1.2 Central problems

Having outlined the properties of an ideal timeline in my setting, I now turn to describing the problems involved in creating timelines with these properties automatically. Firstly, it is necessary to carefully consider the definition of an event. Events have been studied in computational linguistics for a long time. Thus, a considerable body of research is available, covering aspects such as the extraction and classification of events in raw text; the anchoring of events in time; the identification of an event's participants; and many more (Filatova and Hovy, 2001; Grishman et al., 2005; Llorens et al., 2010; Bethard et al., 2012; Kolomiyets et al., 2012). Numerous tasks in this area have also been evaluated using shared evaluations and considerably-sized, publicly available labelled corpora (Linguistic Data Consortium, 2005; Verhagen et al., 2007, 2010; UzZaman et al., 2013; Llorens et al., 2015). Given that the problem of how to correctly define an event also applies in the context of timeline generation, I will first review the literature on event definitions (Chapter 2). The event model I adopt informs both the design of my algorithms in Chapter 6 and the corresponding evaluation methodology in Chapter 5.

Another problem is that the histories of countries, inventions, fields of science, cultural artefacts, food items, cities etc. all have different characteristics. This makes it difficult to design algorithms that work across subject areas. While it is possible to create methods that specialise on a particular type of concept, my ambition in this thesis is to cover as many subject areas as possible.

A further important aspect to consider is evaluation. Evaluation of timelines is difficult for many of the same reasons that complicate summarisation evaluation, next to a range of problems specific to timelines that stem from the presence of dates. Chapter 4 therefore discusses existing summarisation evaluation techniques, before turning to the evaluation of timelines.

Another task worth investigating is the automatic identification of important historical figures that should occur in timelines. This idea is rather novel in this thesis, given that previous research has favoured event-based timelines. A main observation made here was that evaluation and algorithms alike need to operate on the level of entities rather than on the level of individual name mentions, for reasons that will be discussed in Chapter 3. Other practical problems that will need addressing include the presence of different variants of the same person name, as well as the lack of suitable evaluation resources.

Discourse effects are another aspect one should keep in mind when designing a system that creates timelines. As discussed earlier, one of the desirable properties of a timeline is that each entry can be read in a stand-alone fashion. My analysis of naturally occurring timelines confirms that good timelines are indeed often written in this way. However, one could still interpret a timeline as a special discourse object in its own right, assuming that it is usually read from top to bottom. In that case, the timeline should correspond to discourse conventions. For instance, one should strive to avoid cases where explanatory side information is repeated in nearby timeline entries. A human would arguably perceive two subsequent timeline entries starting with "Mariano Rajoy, the prime minister of Spain," as unnatural, since the information that Mariano Rajoy is prime minister is superfluous in the second timeline entry. In this thesis, I do not investigate the problem of constructing a surface representation of events that is suitable for timelines. However, the evaluation resources I provide allow for the testing of this aspect of timeline generation as well.

Timeline entries may also be related in different ways. For instance, a timeline which mentions an ethnic cleansing arguably benefits from some indication of inter-ethnic tensions at an earlier time. In the ideal case, a timeline might even respect certain logical (e.g. causal) constraints between events. One way of modelling such relationships is to encode co-selection constraints between multiple events, which I will do using integer linear programming.

Another central problem that I face is human subjectivity of annotation. I elicit gold-standard timelines from human volunteers, and the problem of subjectivity comes into play because they do not fully agree on which events should be chosen for the timeline. This problem made it necessary to think carefully about the instructions given to human timeline writers, agreement metrics, and other ways of ensuring that the evaluation is sound.

1.3 Thesis outline

This thesis consists of three main parts. The first part (Chapter 2) reviews existing literature on timeline generation. I show that very different problems tend to be conflated under this label. This includes the exhaustive identification of links between events and temporal expressions in a given text; creating a tree structure which encodes transitive temporal relations between events; finding the most salient articles in a large collection of dated news articles; creating a biography timeline for a given historical figure; and many others. In this context, different event models used in the literature will be discussed as well. I will use insights from this chapter to inform my own algorithms, for instance when identifiying instances of events and temporal expressions in text.

The second part of the thesis (Chapter 3) presents a preliminary experiment in the context of timeline generation. I develop a system which selects significant historical

figures that should be contained in timelines. After giving an overview of my approach, I present a corpus built using existing timelines and textual articles harvested from the Web. I then introduce a supervised method for solving this task, which can be seen as a sub-task of timeline generation. Aside from demonstrating that my method outperforms a competitive frequency baseline, I discuss central observations which have shaped the remainder of my work in many ways. In particular, I describe why I will not make use of the corpus presented here for the task of timeline generation.

The third part of the thesis (Chapters 4, 5 and 6) describes my approach to timeline generation, i.e. the task of identifying events relevant for a timeline in a single history article. Evaluation of timeline generation is a substantial research question in its own right. Accordingly, it was not clear how to construct an evaluation resource of acceptable standard in a principled and reliable manner. I therefore develop a comprehensive evaluation methodology for this task, which like some summarisation evaluations uses a deep, semantics-oriented comparison with gold-standard timelines, rather than the cheaper, surface-based comparisons which I argue are inferior. Issues of summarisation evaluation are discussed in detail in Chapter 4.

My evaluation methodology has a central advantage which makes it considerably more useful than these existing methods in practical terms. Concretely, my methodology allows for the evaluation of an unlimited number of system timelines at no additional annotation cost once an evaluation resource has been created. This advantage enables me to test many more methods for timeline generation deeply than would otherwise be possible.

Chapter 5 is dedicated to my new evaluation methodology. I start by defining key concepts, in particular that of a Historical Content Unit (HCU). Next, I give a high-level overview of the process of constructing an evaluation resource. The main steps are the elicitation of human-written timelines; the identification of HCUs in these timelines; and the annotation of links between HCUs and certain words in free-form text. For each step, I present a comprehensive set of guidelines which can be used for creating further evaluation resources of the same kind in the future. I also discuss how timelines are scored based on this evaluation resource. My evaluation methodology will be used in Chapter 6, where I evaluate the performance of various algorithms on the task of timeline generation.

I also present a novel evaluation resource which is based on my methodology. The creation of this resource involved collecting more than 30 human-written timelines using a carefully designed set-up. Because subjectivity is a potential problem, I analyse whether the evaluation resource is replicable, and verify that the concept of HCU weight is meaningful. I also present a separate development resource which can be used to prototype algorithms.

Finally, in Chapter 6, the performance of different timeline generation algorithms is evaluated. I describe both uninformed methods that exploit explicit document structure and the presence of dates, and informed methods that rely on different forms of external knowledge. This includes an improved version of a supervised approach from the literature, and unsupervised methods that take into account aspects such as the article's subject area, syntactic connections between a date and an event, and co-selection constraints between multiple events. The results show that a combination of three unsupervised methods provides the numerically best result, while document structure also proves a strong predictor of events that should figure in timelines. These results have inspired some of the possible avenues for future work that I will discuss at the end of my thesis (Chapter 7).

Chapter 2

Related work on timeline generation

In this chapter, I will give an overview of how timeline generation has been approached in previous work. The term "timeline generation" has been used to describe various tasks in a number of different research areas. These areas include multi-document summarisation, event extraction, temporal information extraction, information retrieval and event classification. For each of these interpretations of timeline generation, I will point out the differences and similarities to the task considered in this thesis.

Section 2.1 gives a general overview of aspects in which approaches to timeline generation differ. Section 2.2 presents different event models used in the literature. In particular, I will describe the TimeML framework, which is concerned with the identification of events, temporal expressions and links between these objects.

TimeML will play an important role in my approach to timeline generation (Chapters 5 and 6). In particular, an existing software package to identify TimeML events in text will be used extensively. In Section 2.3, I will therefore describe a number of shared tasks that evaluated such tools, and will motivate my choice of third-party TimeML tool. In Section 2.4, I will describe more complex research tasks that build on the TimeML framework. These include a way of generating a single connected timeline of TimeML events, and the task of identifying all events in a document collection that a particular entity (e.g. a person) is involved in.

The common feature of all approaches based on the TimeML framework is that the *salience* of events and their relevance to a user or query are irrelevant. Instead, the objective is to identify all event instances and all relations that hold between them. This is different for the works described in Sections 2.5 and 2.6. In Section 2.5, a large body of works in the tradition of multi-document summarisation will be reviewed. In order to identify important events, such approaches tend to exploit the presence of a corpus of multiple news texts for a given time period. Section 2.6 presents an existing supervised approach to event classification, in which TimeML events are classified as important or non-important.

In Section 2.7, I will show that none of the existing methods is applicable to the task investigated in this thesis, and describe the requirements for my approach.

2.1 Overview of related work

In general, works that aim to create timelines from text differ in the following ways:

Overall nature of the task: A key criterion when comparing existing works on timeline generation is the overall nature of the timeline creation process. Timelines built may include either as many entries as possible, or a limited number of entries that are important or relevant in some way. This distinction has major consequences for the system required for constructing such timelines. All systems must first identify content that is suitable for a timeline (e.g. events of a certain kind). Systems that aim to create a timeline of limited length must additionally be able to distinguish more relevant from less relevant content in the source text.

Number of input texts: Whether the task is defined in a single-document context or a multi-document context plays a crucial role. In a text collection, important events are likely to be mentioned more frequently than other events. This redundancy can be exploited by algorithms that select salient events for the timeline. In a single-document setting, such information is not available. Cues about what is or is not important have to be identified in the single article at hand, or harvested from general background corpora. In general, this is a much harder task.

Availability of metadata: The process of building a timeline depends on the nature of the source text. For articles in large news corpora (as opposed to most other texts), publication dates are often available. These dates simplify the process of arranging events on a timeline, as well as the identification of salient events.

The topical focus of the timeline to be built: The timelines constructed may either be generic or query-specific. Generic timelines are expected to broadly reflect the content found in the input article, with no focus on a particular topic or query. Conversely, a query-specific timeline should contain only content that is relevant to a given information need such as "politics" or "church".

The output representation to be used: Timelines may also differ with regard to how events are presented. For instance, a timeline may consist of short noun phrases ("Resignation of Cameron"), or full sentences ("Cameron resigned after he lost the EU referendum"). In practice, timeline entries can be constructed either by extracting text from the source article, or by generating entirely new sentences. A special case occurs where a timeline summarises a corpus in which each document describes a single event. In this case, the creation of a timeline entry involves the identification of a suitable shorthand representation of the article, such as the title of the article or its first sentence.

2.2 Event models

Most existing works on timeline generation interpret a timeline as a list of *events*. However, given that there is no generally accepted definition of what constitutes an event, it is often unclear whether a given text span represents an event or not.

2.2.1 Events in linguistics

A number of influential linguistic theories developed in the 20th century addressed the problem of how to define an event. Davidson (1967) argued in his seminal work "The

logical form of action sentences" that the well-established representation of action verbs as predicates with "slots" for the verb's arguments (i.e. $KILL(killer, killee)$ for the verb "to kill") should be extended with a special event slot filled by an existentially quantified event variable e. For instance, according to Davidson the verb "to kill" should be represented as a three-place predicate $KILL(killer, killee, e)$. This definition allows for an elegant treatment of non-essential adverbial modifiers describing, for instance, the time or the place at which the event took place. In Davidsonian event semantics, these modifiers are no longer part of the predicate, but instead related to the relevant event variable by means of a separate, dedicated predicate, such as $AT(e, 2pm)$ in the following example:

$$\exists e(KILL(killer, killee) \wedge AT(e, 2pm)) \tag{2.1}$$

Works in the *Neo-Davidsonian* tradition, such as Higginbotham (1985, 2000), Parsons (1990, 2000) and Kratzer (1995), adapted the original proposal of Davidson in various ways[1]. In particular, they no longer distinguish between essential arguments (such as $KILLER$ and $KILLEE$ above) and a special event slot. Instead, the predicate representing the action verb now has the event variable e as its only argument, while all other arguments are expressed using separate conjuncts:

$$\exists e(KILL(e) \wedge KILLER(killer, e) \wedge KILLEE(killee, e) \wedge AT(e, 2pm)) \tag{2.2}$$

In this way, all instances of the verb "to kill" can be represented by the same predicate $KILL(e)$, even in cases where arguments commonly as seen as obligatory (such as the verb's subject or object) are missing.

While Neo-Davidsonian semantics initially only covered action verbs (in the spirit of Davidson's original proposal), it was soon established that other types of verbs, notably state verbs, should be analysed in the same way (Parsons, 2000; Chierchia, 1995). For greater clarity, an event in the Neo-Davidsonian tradition is therefore often referred to as an *eventuality*. This term was proposed by Bach (1981) and covers both events and states.

In fact, the nature of eventualities has been studied independently of their role in formally defining the semantics of action verbs. In particular, Vendler (1967) proposed a number of *aspectual types*. These types, often referred to as *aspectual classes* in later works, represent typical time schemata of concrete utterances, i.e. they indicate what the speaker predicates of the event relative to other events in the discourse (Moens and Steedman, 1988). The most widely used classification of aspectual classes for eventualities by Moens and Steedman (1988) explicitly distinguishes *events* from *states*; events are again sub-divided into *culminations*, *points*, *processes* and *culminated processes* (see Figure 2.1 for a graphical overview). The aspectual class of an event is determined by two criteria. Culminations and culminated processes both entail a transition to a new state (called "consequent state") of the world, while points and processes do not. The second criterion is whether the speaker sees the event as punctual or continuous; culminations and points present an event as punctual, while processes and culminated processes result in a continuous interpretation. States, i.e. happenings without defined happenings and ends, fall outside of the aforementioned event classes. Typical verbal constructions for each of the five classes are shown in Figure 2.1.

[1] See Maienborn (2005) for a more detailed description.

	EVENTS		STATES
	atomic	extended	
+conseq	CULMINATION (recognize, spot, win the race)	CULMINATED PROCESS (build a house, eat a sandwich)	understand, love, know, resemble
-conseq	POINT (hiccup, tap, wink)	PROCESS (run, swim, walk, play the piano)	

Figure 2.1: Aspectual classes and typical examples (taken from Moens and Steedman (1988))

However, it is important to understand that, while many verbal constructions tend to evoke a particular aspectual class, this class is not invariable, or dependent only on the verb's lexical semantics. Instead, the "default" aspectual class can be overridden by modifiers in the concrete utterance in question, which can "coerce" an event into a different aspectual class (Moens and Steedman, 1988). For instance, while a proposition such as "I coughed" usually refers to a point, the presence of a progressive auxiliary (such as in "I was coughing") enforces a non-standard interpretation of the coughing as an ongoing process. Moens and Steedman (1988) give a detailed account of when coercion from one aspectual class to another occurs, separately for each pair of classes.

2.2.2 Early event models in Natural Language Processing

The ability to detect instances of eventualities in text automatically is useful for a wide range of downstream tasks in Natural Language Processing, including information extraction and summarisation. In what follows, I will refer to all eventualities as "events", following the terminology conventionally used in the community. In practice, algorithmic approaches to event detection differ in whether they cover state descriptions in addition to events in the strict sense.

Early work on topic detection defines an event as a unique thing that happens at some point in time, along with all necessary preconditions and unavoidable consequences (Allan et al., 1998; Allan, 2002). More concrete definitions were proposed at the 2001 ACL Workshop on Temporal and Spatial Reasoning, which addressed the problem of automatically analysing events and temporal expressions in text. For instance, Filatova and Hovy (2001) divide input sentences into "event clauses" based on a parse tree. A new event is created whenever a node in the parse tree contains both a subject and a predicate. This can be problematic, as this shifts the definition of an event entirely onto the syntax of the sentence, not the semantics, such as the presence of an action or state change in the real world. For instance, in the sentence "Catalonia was invaded and divided", the predicate will contain one subject but two verbs. Hence, by Filatova and Hovy's definition, the invasion of Catalonia and its subsequent division would be interpreted as a single event. If the same content were described in two separate sentences ("Catalonia was invaded. It was then divided"), two separate events would be created for the identical semantics. This is clearly disadvantageous.

An alternative approach by Schilder and Habel (2001) assumes that events are represented by individual words (or spans of words) in text, rather than by clauses. In

particular, verbs and specific nouns can be the lexical bearer of information about the event in question. The latter category includes nominalisations of verbs (such as "construction" or "election"), but also a small number of what the authors call "event nouns". Schilder and Habel give a few examples of such event nouns for the stock market domain (such as "opening of the stock exchange" or "opening bell"), but do not go into further detail about how these event nouns are defined and how they can be identified in text.

2.2.3 TimeML

The de-facto standard formalism for annotating events and temporal expressions in text is the mark-up language TimeML, which will play an important role in later chapters. TimeML has become popular in the community due to several shared tasks in the TempEval series (presented in Sections 2.3 and 2.4). Figure 2.2 shows TimeML annotation for two sentences taken from the Wikipedia article "History of East Timor" (created by the publicly available tool TIPSem-B, which will be described in more detail in Section 2.3.2). I will refer to this example in what follows when presenting central concepts of TimeML.

Pustejovsky et al. (2003a) provide a high-level description of the TimeML language.[2] As far as events are concerned, TimeML follows the intuition of Schilder and Habel (2001) in that individual words are marked up as events, not clauses as in Filatova and Hovy (2001). These words include verbs, nominalisations and certain event-like nouns such as *war* or *crowning*. The exact definition of an event in the TimeML specification is based on the work of Setzer (2001), who considers anything that is anchorable in time to be a potential event. She states that events are usually conveyed by finite verbs or nominalisations. However, Setzer points out that what constitutes an event in a concrete task also depends on the application, as well as the domain and genre of the source text. For instance, according to her definition, the fact that Berlin is the capital of Germany would not constitute an event if it were mentioned in passing, e.g. in a recent newspaper article. However, in the context of an article describing the evolution of Germany over the past two centuries, the fact that Berlin has been the German capital from 1871 onwards is likely interpreted as an event (Setzer, 2001).

The TimeML specification by Pustejovsky et al. (2003a) does not elaborate further on this definition. Events are defined as situations that *happen* or *occur*. Predicates describing *states* or *circumstances* in which something holds true are also regarded as events. The specification distinguishes 7 event classes, ranging from occurrences to perceptions (e.g. *see*, *hear*), states (e.g. *love*) and intentions (e.g. *intend*, *want*). Most events are of class `OCCURRENCE` (see the example sentences in Figure 2.2). Other examples of event classes include `PERCEPTION`, which is used with events that express the physical perception of another event, and `I_ACTION`, a class for intensional actions which introduce an event argument. The intensional action allows the reader to infer something about the introduced event (Pustejovsky et al., 2003a). For instance, event `e470` ("pressured") expresses that the introduced event `e471` ("take") may not actually have taken place.

The event definition of TimeML is very general and excludes few verbal constructions. Consequently, any off-the-shelf TimeML event identification software (for instance, the one described by Saurí et al. (2005)) marks up almost all verbs as events, including stative verbs; only modal verbs (such as "must" in "she must come") and auxiliary verbs

[2] In addition to the general overview in Pustejovsky et al. (2003a), detailed annotation guidelines (including a formal language specification) have been made available (Saurí et al., 2006).

```
The letter <EVENT class="OCCURRENCE" eid="e442">upset</EVENT> Habibie , who
<EVENT class="PERCEPTION" eid="e443">saw</EVENT> it as <EVENT
class="OCCURRENCE" eid="e444">implying</EVENT> Indonesia was a `` colonial
power '' and he <EVENT class="OCCURRENCE" eid="e445">decided</EVENT> in
response to <EVENT class="OCCURRENCE" eid="e446">announce</EVENT> a snap
<EVENT class="OCCURRENCE" eid="e447">referendum</EVENT> to be <EVENT
class="OCCURRENCE" eid="e448">conducted</EVENT> within <TIMEX3
type="DURATION" value="P6M" tid="t57">six months</TIMEX3> .
(...)
Activists in Portugal , Australia , the United States , and elsewhere <EVENT
class="I_ACTION" eid="e470">pressured</EVENT> their governments to <EVENT
class="OCCURRENCE" eid="e471">take</EVENT> action .
```

Figure 2.2: Examples of TimeML event annotations for two sentences in the Wikipedia article "History of East Timor".

in negations (e.g. "did" in "did not know") are excluded.[3] The main benefit of using TimeML over simply working with all verbs is therefore TimeML's additional coverage of certain non-verbal constructions.

TimeML can also be used to annotate temporal expressions and relations. Here too, the language builds on earlier work by Setzer (2001), who proposed to annotate temporal expressions using the *TIMEX* tag. TimeML uses a revised version of this tag called *TIMEX3*. Text spans that can be marked up using a TIMEX3 tag include absolute (such as "13th January 2012") and relative (such as "last Tuesday") temporal expressions, as well as durations (e.g. the temporal expression `t57` ("six months") in Figure 2.2) (Pustejovsky et al., 2003a).

The TimeML language also defines a number of relations that can be annotated in text. *Temporal links* (referred to as *TLINK*) are relationships that hold between two events or between an event and a temporal expression. In contrast, *subordination links* and *aspectual links* (abbreviated as *SLINK* and *ALINK*, respectively) are always defined between two TimeML events. A subordination link (Saurí et al., 2006) is used where an event word modifies the meaning of a subordinated event word. Six types of such links exist. For instance, the presence of the event word "forgot" in the sentence "John forgot to buy some wine" entails that John did not buy wine. In this case, one would annotate a subordination link of type `COUNTER-FACTIVE` between the events "forgot" and "buy".

Aspectual links (referred to as ALINK) are annotated between an aspectual event word and its argument, such as "started" and "doing" in "Mary started doing her homework" (Saurí et al., 2006). Aspectual links can be subdivided into subtypes, in the same way as subordination links. However, the TimeML annotation guidelines only provide examples of such types instead of an exhaustive list.

TimeML has been criticised for its lack of distinction between verbs that refer to a real-world event and other verbs (Bethard et al., 2012; Minard et al., 2015). If all verbs are annotated as events, the event definition is trivialised, as it almost becomes unnecessary to distinguish events from non-events. Such criticisms could be due to confusion over whether the term *event* in TimeML refers to *events* (in the strict sense), or to all eventualities, as

[3]Note that different exclusion criteria have subsequently been proposed for languages other than English. For instance, the TimeML annotation guidelines for Italian state that modal verbs such as *dovere* (must) should be annotated as events, in contrast to English modal verbs (Caselli et al., 2011).

described in Section 2.2.1.

A second problem of the event definition used in TimeML is that the same real-world event may be represented by multiple TimeML events depending on the surface representation. For instance, the beginning of a war mentioned in text is typically represented by two TimeML events, as both "war" and "started" in the sentence "The war started in 1914" will be recognised as TimeML events. Hence, there is no one-to-one correspondence between a real-world event and a TimeML event.

There are different ways of reacting to this lack of one-to-one correspondence. Bethard et al. (2012) use a modified version of the original TimeML annotation guidelines in the context of their work on temporal dependency trees that I will discuss in Section 2.4.1. In these revised guidelines, modal, hypothetical and negated verbs, as well as light verbs and aspectual verbs, are excluded from the definition of an event. Further, it is not allowed to annotate multiple TimeML events in phrasal constructions such as "managed to do" or "did his best to reach them". A second example is a recent shared task on cross-document timeline generation (described in Section 2.4.2), in which events in the gold standard are also annotated using a modified version of the guidelines. In particular, the event definition used here excludes cognitive, counter-factual and other events which cannot easily be placed on a timeline.

2.2.4 Automatic Content Extraction

An alternative event model, used for the *Automatic Content Extraction* (ACE) evaluation effort, interprets an event as a complex semantic unit. Here, the event definition includes the event's arguments. For instance, an event of type BE-BORN has three arguments: the person who was born, the time when the birth took place, and the place where the birth took place. Each event is assumed to have a textual *extent*, in which the event's *arguments* are located. In most cases, this extent corresponds to an entire sentence. The event's *trigger* is the word which expresses the core semantic content of the event; this will mostly be a verb or event-like noun, as in TimeML.

ACE restricts itself to the annotation of events of pre-specified types. In particular, the ACE 2005 evaluation uses 8 event types with a total of 33 sub-types, ranging from common life events such as "marriage", "divorce" and "injury" to events in business and justice (Linguistic Data Consortium, 2005). Many real-world events do not fall in one of these categories and are therefore omitted from processing.

2.3 Temporal information extraction

Having compared various event models used in the literature, I now review existing algorithms that create some form of timeline automatically.

One group of algorithms identify *all* events described in a given text and anchor them in time whenever possible. Such methods do not consider the salience of an event or temporal expression for constructing a timeline. This line of work was pioneered by Filatova and Hovy (2001), who transform news stories that describe the development of a situation into a corresponding timeline representation. They address the problem that the order in which events take place in the real world is often different from the order in which they are mentioned in a news story, a problem which might confuse algorithms that aim to solve downstream tasks such as the identification of causal and other relationships

between events. Their aim therefore is to represent a news story as a grid of underlying events, dated according to their timestamps.

2.3.1 The TempEval shared tasks

The performance of algorithms that fall in this group has been extensively evaluated in the context of the TempEval shared tasks, which make use of the TimeML language. It should be noted, however, that these shared tasks only assessed the identification of relations between certain pairs of events as well as pairs of events and temporal expressions, i.e. it is not compulsory for a system to produce a total ordering of events according to time. Algorithms evaluated on these tasks are therefore not necessarily able to create a single (i.e. connected) timeline of events for a given input text, because there will be many events for which no temporal relation to an event earlier or later in the text has been identified.

The first of these shared tasks, TempEval, addressed the identification of relations between events and timestamps in a setting where both events and timestamps are given (Verhagen et al., 2007). The scope of this task was relatively limited in that events could only be linked to temporal expressions in the same sentence or to the document creation time (DCT) of the containing article. Events whose date could only be inferred from the wider document context can therefore not be anchored in time using this approach. Apart from relations between events and timestamps, the shared task covered relations between two *main events* that occur in subsequent sentences, where the main event was usually taken to be the syntactically dominant verb of a sentence (Verhagen et al., 2007). The objective was to assign one of the relation types proposed in the TimeML standard (such as `BEFORE`, `OVERLAP` and `AFTER`); in total, 13 such types exist.

TempEval-2 (Verhagen et al., 2010) included further tasks in addition to those pioneered by TempEval. These covered the extraction of both events and timestamps from raw text, as well as the identification of relations between events in the same sentence. TempEval-2 was also designed to be multilingual: Evaluation resources for five languages, including non-European ones, were provided. In contrast, TempEval had been limited to English texts. At the same time, the number of relation types for the task of relation identification was reduced to five. Both TempEval and TempEval-2 used a small corpus called TimeBank for evaluation, which contains news articles manually annotated according to the TimeML standard.

In the subsequent TempEval-3 shared task (UzZaman et al., 2013), three subtasks were investigated: identification and normalisation of temporal expressions (task A); identification of events (task B); and identification of temporal relations between events (task C). The key difference to earlier tasks is that a more realistic end-to-end evaluation (task ABC) was performed in addition to the evaluation of individual tasks. In this end-to-end evaluation, the events and timestamps to be related to each other were no longer given, and hence errors made in earlier stages of the pipeline (i.e., in event and temporal expression detection) were propagated to the components which annotate temporal relations. It is expected that the end-to-end evaluation gives a more realistic picture of how a complete temporal information processing system performing all three tasks would fare on unseen text. Another important difference to earlier TempEval tasks is that a much larger evaluation corpus was used in TempEval-3.

The methods evaluated on the TempEval-3 shared task included both rule-based and machine-learning-based approaches, as well as a number of hybrid methods. I will now summarise the main results of this competition.

Rule-based methods performing task A (identification and normalisation of temporal expressions), such as HeidelTime (Strötgen and Gertz, 2010), typically combine regular expression patterns, which are applied to the tokens in a sentence, with further constraints (e.g. on part-of-speech tags). In HeidelTime, a rule covers both the extraction and the normalisation of a time expression, a property which allows the algorithm to perform these two steps concurrently. A post-processing step is however necessary for underspecified, relative time expressions such as "last September", which can only be disambiguated once all absolute time expressions have been recognised. One example of a machine-learning-based method is the algorithm proposed by Filannino et al. (2013), which uses a conditional random field (CRF) to combine morphological information with other features, such as a shallow parse of the sentence, a number of gazetteers covering various types of named entities, and information from WordNet. The evaluation showed that the best rule- and machine-learning-based systems performed comparably on the sub-task of temporal expression identification. For normalising temporal expressions, however, rule-based systems such as HeidelTime outperformed machine learning approaches by a large margin. However, this picture might change in the future, given that more sophisticated machine learning approaches for these tasks could be developed, particularly if a greater amount of training data became available.

For event identification (task B), machine-learning-based systems typically exploit three classes of features: morphosyntactic information, lexical semantic information (e.g. WordNet), and sentence-level semantic information (e.g. semantic role labeling) (UzZaman et al., 2013). The only rule-based system submitted for this task (Zavarella and Tanev, 2013) is an adapted version of an event recogniser for domain-specific events, which uses finite-state machines. In the evaluation, the machine-learning-based systems outperformed the rule-based system. It is hard to judge, however, whether machine learning approaches are inherently better than rule-based methods, given that no further rule-based approaches were evaluated.

Task C (relation identification) can be subdivided into two subtasks: identifying relations in raw text, and classifying them into one of several types. For the subtask of identifying relations, a number of machine-learning-based approaches were proposed which use support vector machines and maximum-entropy models (Bethard, 2013). A second group includes hybrid models such as Laokulrat et al. (2013), which combine simple rule-based processing with support vector machines trained on morphosyntactic and lexical information. On this subtask, a machine learning approach that uses an additional set of gold-standard temporal relations for training (Bethard et al., 2007) performed best.

Only machine-learning-based systems were proposed for the subtask of relation classification, e.g. approaches based on maximum-entropy models and support vector machines (Laokulrat et al., 2013). Again, an SVM model achieved the highest score.

For the new end-to-end evaluation (task ABC), various pipelines combining the aforementioned approaches were proposed. Once more, a system which uses SVM models for all three tasks outperformed models with rule-based or hybrid components.

A further TempEval shared task called QA TempEval took place recently (Llorens et al., 2015). The subtasks in QA TempEval are identical to those investigated in TempEval-3, but systems are scored using an *extrinsic* evaluation methodology. Extrinsic evaluation methodologies verify whether a system's output is useful for a downstream task. In this particular case, systems are evaluated based on how useful the annotation they produce is for the task of question answering.

A corpus of human-written queries with accompanying answers is used as the gold stan-

dard. Systems create the TimeML annotation before any human annotation takes place. Based on this annotation, human annotators are then asked to write yes/no questions about the relative ordering of two events, for instance "Was he wounded after becoming general?". Each judge selects a small number of relations that appear relevant to them. For each query, a formalised version in the format "Is A R B?" must be provided, where A and B are TimeML events identified in the text and R is a pre-specified relation type. For the query above, this would result in a query such as "IS *e123* AFTER *e128*", where *e123* refers to the wounding event and *e128* denotes the event of becoming a general.[4] In addition, the annotator must give the answer (yes or no) to the question. Human judges may ask questions involving any two events from the source article. As opposed to the earlier TempEval tasks, they are no longer restricted to choosing relations involving two events in the same sentence. An algorithm is awarded a point if the annotation it has produced is consistent with the human-written answer.

While earlier TempEval tasks only addressed news articles, the evaluation corpus used here includes blogs and history articles as well as news articles.

2.3.2 Choice of TimeML third-party tools

Event mentions in free-form text are an important building block of my evaluation of timeline generation (cf. Chapter 5). Similarly, temporal expressions will play an important role in Chapter 6, where algorithmic approaches to timeline generation are discussed. For identifying such events and temporal expressions in raw text, I use publicly available third-party tools based on the TimeML framework. An advantage of doing so is that these tools have been rigorously evaluated in the context of the TempEval shared tasks. I expect that using such tools results in reliable and reproducible preprocessing of input texts. Another advantage is that TimeML event detectors extract nominalised verbs and lexicalised event nouns in addition to event verbs.

Owing to the nature of my task, I am mainly interested in systems that perform tasks A (temporal expression detection and normalisation) and B (event detection) from the TempEval-3 shared task. The task of identifying and classifying temporal relations (task C) is less relevant in my setting, as systems based on TimeML only create a link between an event and a timestamp if the two are in the same sentence. For my experiments, however, I will require each and every candidate event to be linked to its date of occurrence. For linking events to timestamps, I therefore devise my own heuristic (cf. Chapter 6) instead of using a third-party tool.

A large number of systems have been evaluated in the context of the TempEval shared tasks. The one(s) used for pre-processing text in my task should support at least tasks A and B defined in the TempEval-3 shared task; score highly for the English language; be publicly available as a toolkit; and be free to use for academic purposes. Taking into account these criteria, I decided to use different systems for temporal expression detection and event detection, respectively. For detecting temporal expressions, I opted for the state-of-the-art rule-based toolkit HeidelTime (Strötgen and Gertz, 2010) in its most recent version. In the TempEval-3 shared task, HeidelTime was the highest-scoring system for this subtask, achieving an F-score of 0.78.

For event detection, to the best of my knowledge, TipSem-B (Llorens et al., 2010) is the only publicly available system which was evaluated on the TempEval-3 shared task.

[4]Different event IDs assigned to the same events by different systems are normalised before human annotation takes place.

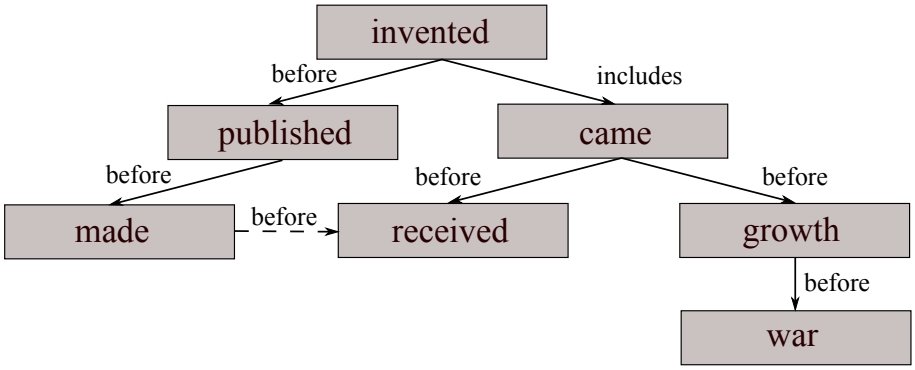


Figure 2.3: Example of the dependency tree annotation used by Bethard et al. (2012).

It is a slightly simplified version of the system TipSem, which won the TempEval-2 shared task. On the subtask of event identification (Task B) in TempEval-2, TipSem achieves an F-score of 0.83, while TipSem-B achieves an F-score of 0.82. The creators of the TempEval-3 dataset used TipSem for pre-processing their data because of its superior performance in TempEval-2. For this reason, TipSem was not formally allowed to take part in the TempEval-3 evaluation, but its performance on the new dataset was reported nonetheless in order to facilitate a full comparison of all existing approaches. TipSem achieved the highest score in TempEval-3 as well, but how much of this is due to its use for pre-labelling data is unknown.

Given that TipSem and its variant TipSem-B performed well in the shared tasks mentioned above, the publicly available version TipSem-B was used as a pre-processing tool for the corpus annotation in Chapter 5, as well as for my experiments in Chapter 6.

2.4 Timeline generation based on TimeML

In the years since the first three TempEval shared tasks (which, despite their differences, were all relatively similar in nature and scope), some approaches were proposed which use the terminology and general intuition of TimeML, but address more complex tasks.

2.4.1 Temporal dependency trees

The work of Bethard et al. (2012) aims to remove a central restriction of the approaches evaluated in the TempEval shared tasks, namely the fact that temporal relations between events are annotated independently of each other. A set of independent local relations cannot easily be transformed into a single timeline, if at all.

Bethard et al. instead define a timeline as a *temporal dependency tree* that encodes partial ordering relations between events. Figure 2.3 shows an example of such a tree. Boxes correspond to TimeML events, and arrows represent relations between two events. Relations represented by solid lines are explicitly signalled in the text by linguistic cues. Dashed edges represent additional relations which could be identified using world knowledge although there is no linguistic cue. Note that only the solid lines form a tree, while the entire graph of events shown can have cycles.[5]

[5]The graph in Figure 2.3 has an (undirected) cycle because the event *received* has more than one parent.

Temporal dependency trees can be used to make inferences about temporal relations. For instance, one can derive from the example tree in Figure 2.3 that the event "made" took place after the event "invented", although this relation has not been explicitly annotated. It is sufficient to know that there exists an event "published" which took place after the event "invented" and before the event "made".

Bethard et al. (2012) elicit a set of gold-standard temporal dependency trees from human annotators. The trees should express only relations explicitly signalled in the text (represented by solid lines in Figure 2.3). In practice, annotators, instead of drawing up a tree directly, identify local relations between two events one at a time. However, since it is only allowed to annotate a new relation between two events if one of them (called a *connection point*) is part of a relation annotated previously, the set of relations elicited can be automatically transformed into a temporal dependency tree. If there are multiple suitable connection points for a newly annotated event, the temporal relation that is easiest to infer from the text should be chosen (Bethard et al., 2012).

Kolomiyets et al. (2012) argue that temporal dependency trees are to be preferred over a total ordering of all events in the text, given that many texts do not specify – even implicitly – a temporal relation between each and every pair of events. But even temporal dependency trees can arguably only be elicited straightforwardly for texts from less complex genres. Bethard et al. (2012) in particular annotate children's stories, which typically progress linearly and exhibit direct cues such as "after" or "before" between neighbouring events. This property makes it easy for human annotators to identify event orderings. In more complex genres such as novels, blogs or news articles, the ordering of events may be underspecified, unknown or not relevant. It is also possible that the text intentionally jumps back and forth in time, or that an event refers back to an event mentioned several paragraphs earlier. In order to annotate event orderings for such texts, the annotator would be required to screen the entire previously read text for possible connection points. This is likely to make the task too complex and arbitrary. Here, restrictions such as the ones imposed by the TempEval shared tasks (where relations may only involve events and temporal expressions from the same or nearby sentences) can be used to keep the annotation effort manageable.

Kolomiyets et al. (2012) were the first to develop a method that transforms an input text into the temporal dependency tree notation proposed in Bethard et al. (2012). They apply two parsing models well-known from the dependency parsing literature, showing that a shift-reduce parser (Nivre, 2008) outperforms a graph-based parser (McDonald et al., 2005). For evaluation, a new corpus of children's stories is annotated using the set of guidelines provided by Bethard et al.

The creation of temporal dependency trees of events is a step towards a more global processing of documents for creating timelines. As with all approaches that build on the TimeML framework, the importance of an event is not a criterion in this work. This is different in the task I approach, where a timeline consists of a limited number of salient events which together cover the entire time period described by the source text.

2.4.2 Cross-document event ordering

The TimeLine shared task in the SemEval series is another example of a more complex form of timeline generation that was approached using the TimeML framework. Here, timeline generation is performed in a special multi-document setting called *cross-document event ordering* (Minard et al., 2015). The aim is to extract, from a corpus of documents,

Date	Event
1932	born
1953	graduated
1977	won
1978	attacked
1988	ran
1995	elected
2002	won
2005	suffered
2007	resigned
2008	founded
2011	convicted

Figure 2.4: Extract of an example timeline of events for the entity "Jacques Chirac".

all events that a given target entity (such as a person) is involved in. These events must then be linked to their time of occurrence and arranged on a single timeline, such as the one shown in Figure 2.4. As the events in the timeline are not taken from the same document, local relations between events need not be identified.

The shared task involved four sub-tasks. Two of these (Track A and Sub-Track A) required systems to identify event mentions automatically, while for the other two sub-tasks (Track B and Sub-Track B), the gold-standard event mentions were provided. The sub-tracks also differed according to whether explicit time anchors (such as the dates in Figure 2.4) had to be annotated (Track A and Track B) or not (Sub-Track A and Sub-Track B).

Evaluation was performed using a semi-automatically created corpus consisting of 37 gold-standard timelines for a total of 90 documents covering three topics ("Airbus and Boeing", "General Motors, Chrysler and Ford" and "Stock Market"). The gold-standard timelines were constructed using a four-step process: First, all occurrences of a target entity (such as "Steve Jobs") in all input documents were annotated manually. These occurrences were used in the second step, the annotation of events in which the target entities participate (including their dates). Using this information, it is possible to automatically create a draft timeline. Human annotators then completed the draft timelines using further information from the text.

Four systems participated in all sub-tasks, while results for Sub-Track A and Sub-Track B were submitted by three of the four systems. All proposed algorithms created timelines using a pipeline architecture combining multiple existing language processing components, such as Stanford CoreNLP (Manning et al., 2014). For two of the four subtasks (Track B and Sub-Track B), a system which uses an existing semantic role labeler as well as a topic modelling approach (Navarro and Saquete, 2015) obtained the highest score. For Track A, a rule-based approach was shown to perform best[6]. For Sub-Track A, a system that creates separate timelines for each input document and then merges them (Caselli et al., 2015) gave the best results.

[6]There is no citation available describing this approach, because participants in this shared task were not obliged to also submit a paper explaining the method used.

2.5 Timeline generation in a multi-document summarisation setting

A further strand of work, which is very different in nature from the approaches described previously, casts timeline generation as a multi-document summarisation task. Here, the aim is to create a timeline of limited length which only contains the most salient information, based on a large corpus of documents. Such a timeline can be used as a browsing interface to the original document collection.

The difficulty of summarisation, and in particular multi-document summarisation, lies in the need to identify salient information that should be part of the summary while avoiding the inclusion of redundant information. It should be noted that the presence of redundant information in the source text is not problematic in itself. In fact, such redundancy is commonly used to identify salient content in the first place, both for single-document (Luhn, 1958; Edmundson, 1969; Paice, 1990; Hovy and Lin, 1999) and multi-document summarisation (Goldstein et al., 2000), as important information is more likely to be mentioned multiple times.

In order to construct a summary that is both relevant and non-redundant with regard to previously added content, early work on multi-document summarisation applied the principle of *maximum marginal relevance* (MMR). The algorithm of Carbonell and Goldstein (1998) greedily adds sentences to the summary that are both relevant and non-redundant with regard to the sentences already added, until the desired summary length has been reached. Ways of calculating relevance and redundancy are discussed in Goldstein et al. (2000).

2.5.1 Identifying important topics

I now turn to describing algorithms which interpret the particular problem of timeline generation as an instance of multi-document summarisation. Swan and Allan (2000) are arguably the first to approach timeline generation as a multi-document summarisation task. The objective of their algorithm is to detect salient topics in a large corpus of news articles, and the time periods during which each topic was most prominent in the news. Examples of topics include a school shooting or the death of an important historical figure. The association metric χ^2 is used in a first step to identify named entities and noun phrases which are overrepresented in a given time frame. In order for this to work, each article must have an annotated publication date. In a second step, the named entities and noun phrases are grouped into *topics* using an agglomerative clustering approach.

Swan and Allan identified three criteria that a good set of clusters should satisfy: The clusters should correspond to what humans intuitively perceive as a news topic; the named entities and noun phrases in the corpus should make it easy to automatically assign a label[7] to each cluster; and the clusters created should be similar to manually created clusters.

To test whether the method described above is able to produce such clusters, a human evaluation based on the TDT-2 corpus (Cieri et al., 1999) was performed. This corpus was created in the context of Topic Detection and Tracking (TDT), the task of detecting the appearance and evolution of topics in a stream of broadcast news stories (Allan, 2002).

[7] A label is a kind of headline for the cluster and allows a reader to quickly understand what the cluster is about.

Examples of such topics include elections and accidents. Each topic in the corpus groups together a number of directly related events.

Experiments revealed that while annotators perceived that a large number of automatically created clusters corresponded to a single news topic, agreement between annotators on individual cases was very low. This suggests that humans have different intuitions on what constitutes a news topic. For the task of deciding whether automatically created cluster labels were indicative of the topic discussed by the cluster, agreement between annotators was stronger. However, the generally-held view was that the created labels tended to be poor. In contrast, humans found it easy to link a given cluster to a gold-standard topic in the TDT-2 corpus. This suggests that the algorithm of Swan and Allan creates a set of clusters that is similar to what humans would produce when asked to create clusters manually.

2.5.2 Timeline generation based on a user query

Chieu and Lee (2004) investigate the creation of a timeline of events that are relevant to a user query from a large corpus of news articles. Their definition of an event differs from commonly used event models (cf. Section 2.2) in that an event is assumed to be represented by an entire sentence. Chieu and Lee follow Swan and Allan in using association metrics to identify events that were frequently discussed in a given time period. However, instead of using document creation times directly, they extract timestamps for each event from the surrounding article text. To this end, they compare a number of ways of linking an event to its time of occurrence using document creation times. For instance, the date expression "three days ago" is assumed to mean *three days before the news article was written*.

It is also assumed that the same real-world event can be expressed by multiple sentences. For this reason, the algorithm involves the detection of paraphrases that refer to the same event. The particular approach used here assumes that two sentences are less likely to be paraphrases if the events they describe took place on different dates.

Evaluation was performed using a corpus of gold-standard timelines. In a first step, human experts were asked to construct timelines for a number of queries. Due to the size of the document corpus, it was not possible for them to exhaustively scan all documents for relevant events. They were therefore allowed to use external information sources such as the Web for this task. In a second phase, human annotators were simultaneously provided with gold-standard timelines and a number of system-generated timelines for a given query. Their task was to rank these timelines by quality. In addition, annotators were asked to assess several quality criteria, such as comprehensibility, conciseness and representativeness, for each timeline. The evaluation showed that the system-generated timelines were comprehensible, concise and representative of media coverage. However, agreement between different human annotators was found to be low.

2.5.3 Dependencies between events

Yan et al. (2011b) introduce the concept of "news evolution", which refers to existing dependencies between the events in a timeline. In their approach, timelines are constructed such that they describe groups of interdependent events in the context of a given news topic. For instance, a good timeline describing the outbreak of influenza in a particular year should, in their view, cover both the development of a vaccine and the first use of

that vaccine. Special attention is given to semantic relations between events, such as cause/effect relations, or cases where one event is a further elaboration of another. Yan et al. argue that the presence of timeline entries that stand in such a relation makes it easier to understand the evolution of the underlying news topic.

Timelines are created by composing smaller "component summaries", which group together events that depend on each other. To this end, the algorithm calculates a utility score for each candidate component summary. As the scores of different component summaries depend on each other, an iterative optimisation method is used to calculate a globally optimal solution. A more efficient algorithm for the problem, which optimises an objective function using gradient descent, is proposed in Yan et al. (2011a). Tran et al. (2013) present an alternative approach based on this idea which uses a linear regression model.

Evaluation of the aforementioned works is commonly performed by comparing system-generated timelines to gold-standard timelines. For instance, Tran et al. (2013) perform a web search to acquire gold-standard timelines written by professional journalists. For each topic, multiple timelines are collected in order to reduce bias. Overlap between a system timeline and gold-standard timelines is calculated using surface-oriented overlap metrics such as ROUGE[8] (Lin, 2004). Tran et al. (2013) report a small improvement over the results obtained by Yan et al. (2011b) and over previous approaches such as that of Chieu and Lee (2004) described above.

2.5.4 Salience of dates

The method of Nguyen et al. (2014) assumes that the date when an event took place also plays a role in determining the event's salience. In particular, they assume that salient events tend to happen on salient dates, i.e. on dates with a high number of events, because a salient event is often mentioned together with related side events.

Their algorithm scores each event based on the salience of its textual description, the salience of its annotated date, and the relevance to a user-specified query of both the event and the date. The problem of redundancy is addressed using a simple re-ranking algorithm which considers term overlap between candidate sentences, such that only the highest-ranking sentence for a particular date is included in the final timeline. The approach is evaluated against a gold-standard corpus of newswire texts written by AFP journalists, using ROUGE metrics. As opposed to the dataset used by Tran et al. (2013), this corpus however only contains a single gold-standard timeline for each topic. Given that no direct comparison to previously published methods is performed, it is difficult to establish, for instance, whether their method outperforms the approach by Tran et al. (2013).

The sub-problem of identifying salient dates in a time-tagged corpus of documents has been investigated separately also. Earlier approaches (Kessler et al., 2012) score each date individually, taking into account features such as the overall frequency of a date in a large corpus, as well as the presence of back-references to a date in articles published years later. Events for which back-references exist often have a high impact on subsequent history. For dates that are relative to the document creation time (DCT) of the containing article, such as "ten years ago", the difference between the date of the event to be scored and the DCT is used as a feature, based on the same intuition. Kessler et al. also investigate a variant of the task in which the timeline is constructed in response to a user query.

[8]Surface-oriented evaluation metrics will be discussed in Chapter 4.

Here, the relevance of a document containing a candidate date to that query is another important feature.

Kessler et al. (2012) experiment with several alternative approaches to ranking dates, such as a boosting algorithm and various information retrieval methods. Evaluation is performed by ranking all dates according to their scores and calculating Mean Average Precision (MAP). The same corpus of timelines written by journalists as in Nguyen et al. (2014) is used as the gold standard. The boosting algorithm is shown to outperform all IR models as well as a range of baselines.

The sub-task of date selection is also investigated independently by Tran et al. in the context of their work on timeline generation mentioned earlier (Tran et al., 2013). They use a linear regression model and report an improvement over a baseline that ranks dates by the number of articles published on them.

Tran et al. (2015) extend this work by removing the assumption that the importance of each date can be judged independently. They use a random-walk model to encode the intuition that a back-reference to a date is more significant if it occurs in an article published on another important date. This idea is based on the observation that timelines often contain "substories" of events that are causally linked to each other. An important event such as the resignation of the Egyptian president Mubarak may result in related events being considered important also (e.g. the developments preceding Mubarak's resignation). Evaluation is performed using the corpus of gold-standard timelines used by Tran et al. (2013). Methods that take into account dependencies between dates are shown to outperform approaches that judge each date independently.

2.5.5 Related tasks

There exist a number of tasks that are similar to timeline generation from multiple documents. For instance, Allan et al. (2001) formulate the problem of creating a *temporal summary* for a given news topic based on a stream of news stories. Each sentence in such a summary should describe a new event relevant to the given news topic. Summaries are created in an online setting, i.e. gradually as further news items arrive. This property allows a human to monitor gradual shifts in news coverage of a given topic. The algorithm of Allan et al. uses language models to select sentences that are both useful, i.e. related to the news topic, and novel, i.e. not describing an event already covered by earlier sentences. Sentences that do not express any event also have to be removed.

Evaluation is performed based on a manually created list of events for 11 news topics in the TDT-2 corpus described above. The results suggest that as far as usefulness of the selected sentences is concerned, language models cannot beat a simple round-robin baseline, which selects sentences from the beginning of each article. In contrast, they are successful at reducing redundancy between selected event sentences.

A further related task is proposed by Smith (2002), who create a history browser application for a historical period such as the American Civil War. Their approach relies on the availability of rich resources (such as books) for a time period in question, which may not be assumed to exist for less popular topics. They use association metrics to identify interesting combinations of dates and places in a corpus of documents describing the historical period in question. Next, phrases corresponding to these date-place pairs are identified. Out of the annotation metrics used, mutual information is found to result in a bias towards events rarely mentioned in the corpus, while log-likelihood and χ^2 lead to a balanced selection of less and more frequently mentioned events. These insights

Structural	Digit presence in the sentence
	Position of the sentence in the article normalized by the number of sentences in the article
	Maximum length of any event word in the sentence
	Number of events in the sentence
	Number of "to be" verbs in the sentence
Linguistic	Presence of an event in the perfective aspect in the sentence
	Percentage of events in the sentence that have been assigned class "occurrence"
	Negation presence in the sentence
	Percent of events in the sentence that are verbs
	Percent of events in the sentence that are in some past tense
Named entities	Sum of the named entity weights in the sentence
Semantic	Maximum similarity of any event word in the sentence to the first two nouns in the article
TextRank	TextRank rank of the sentence in the article, divided by the number of sentences in the article

Figure 2.5: Sentential features used in the original method by Chasin et al. (2014).

are based on manual inspection of the produced date-place combinations. While this approach assists users in browsing a corpus, it does not produce a summary of the most salient events, nor does it focus on events specifically. In fact, the phrases include events ("Fire of London") as well as place names ("College Oxford") and person names ("Charles II").

2.6 Identifying important TimeML events

As opposed to the approaches discussed in previous sections, Chasin et al. (2014) focus on the selection of events from a *single* input article. In the context of a study on the visualisation of historical events in text, they propose an approach which classifies TimeML events into important and unimportant ones.

TimeML events in a source article are identified using the event recogniser Evita (Saurí et al., 2005). For each TimeML event, a number of generic features (given in Figure 2.5) are computed. In practice, given that all these features are defined on the sentence rather than on the event level, all events in the same sentence are assigned the same feature vector. Based on these features, an SVM model is then trained using a small annotated corpus of history articles.

2.6.1 Features

The features used can be subdivided into five groups: structural features, linguistic features, semantic features, a named entity feature, and a feature based on TextRank.

Five structural features are used. The first feature considers whether the sentence in which the event occurs contains a digit. Digit presence is an easy way of capturing sentences that contain a date. A further feature captures the position of the containing sentence in the article. In articles discussing a single war or battle (which are used in their evaluation), the key events are mentioned in the first few sentences of the article. The maximum length of any event word in the same sentence, the number of events found in

the containing sentence and the number of instances of the verb "to be" in that sentence are also used as features.

Linguistic features consider certain linguistic properties of event verbs, such as aspect. Verbs with progressive aspect are unlikely to be good candidates for a timeline, given that timelines contain accomplished events leading to a new result state rather than ongoing events. Another feature captures the class attribute of the TimeML events in the sentence (cf. Section 2.2) as detected by the event recogniser. TimeML events of class "PERCEPTION", "REPORTING", "STATE" or "ASPECTUAL" do not represent real-world actions or state changes and are therefore unlikely to be good candidates for a timeline. Similarly, a feature is constructed if there is a negated event, given that negated events are also highly unlikely to be included in a timeline. A further feature considers whether any event verb is used in any past tense. Verbs in the present tense often point to opinions of historians rather than to events that happened in the past, and are therefore less suitable for a timeline.

The named entity feature exploits the fact that events co-occurring with important named entities are likely to be salient. Named entities which occur multiple times in an input article are assumed to be more important than less frequently mentioned entities. In particular, the weight of a named entity is defined as the number of times this named entity is mentioned in the article divided by the total number of named entities in the article.

The semantic feature captures the semantic similarity between an event word and certain keywords which are supposed to represent the general topic of the article. The metric used is the "vector pairs" semantic similarity measure in the WordNet module `WordNet::Similarity` (Pedersen et al., 2004). In this setting, if the word "war" occurs in the first sentence of a Wikipedia article, event words such as "fought" receive a high semantic similarity score. A disadvantage of this approach is that keywords can also include proper nouns such as "Würzburg" occurring in the article title, for which an entry in WordNet is not available.

The final feature is calculated using the TextRank algorithm (Mihalcea and Tarau, 2004), which was proposed as an approach to general single-document summarisation. TextRank is based on the PageRank algorithm (Brin and Page, 1998), which simulates a random walk between web pages. PageRank assumes that a user browsing the web has two options: following one of the links on the current page, or "jumping" to a completely new web page, e.g. by entering a URL directly into the web browser. The model is implemented as a directed graph $G = (V, E)$, where the nodes V represent pages and the edges E represent directed links between pages. The importance of a page is represented by the PageRank score of a corresponding node V_i. This score is calculated iteratively based on the scores of all adjacent nodes (i.e. pages linked to this page). The original version of PageRank in (Brin and Page, 1998) defines the importance of a page as follows:

$$S(V_i) = (1 - d) + d \cdot \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \tag{2.3}$$

where d is a free parameter which represents how likely a user is to follow one of the links on the current page (as opposed to jumping to a completely new page). In Brin and Page (1998), this parameter was set to 0.8. In what follows, I describe this initial version of PageRank.

The first summand, $(1-d)$, is the likelihood that the user jumps to the page represented by node V_i from any other page. This value is assumed to be the same for all pages. The

sum in the second summand represents the votes given by other pages linking to the page represented by node V_i. Intuitively, if such a page has itself a high score, it is likely that the page represented by node V_i is also important. The other page's score is therefore used as the vote. Each vote is normalised by the total number of links on that page: If a linking page contains a high number of outgoing links (e.g. a long list of links), these links are considered to be less indicative of the linked pages' importance.

Since the formula for calculating a node's weight requires the node weights of other, neighbouring nodes, PageRank is a recursive algorithm. The node weights are first initialised to random values, and after a sufficiently high number of iterations the distribution converges, i.e. the node weights are stable in subsequent iterations.

TextRank applies this basic intuition to text summarisation. Two fragments of a text can arguably be related to each other just as pages are connected via hyperlinks. This is the case if the two text fragments express the same or similar information. Intuitively, fragments which contain information that other sentences also contain are more important, and should therefore be part of the summary. However, it is not straightforward to decide when there should be a link between two text fragments. Mihalcea and Tarau argue that, as opposed to PageRank, where the link matrix is binary (since links between pages exist or not), links between text fragments vary in strength. A link between two almost identical sentences should be stronger than the link between two sentences that merely share a person name, for instance.

The score of a sentence in TextRank therefore takes into account edge weights between text fragments as well:

$$S(V_i) = (1-d) + d \cdot \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} S(V_j) \tag{2.4}$$

where w_{ji} and w_{jk} each represent weights between two sentences.

Each node V_i corresponds to a sentence, and the edge weight between two sentence nodes S_i and S_j is calculated using a similarity function, such as the number of common tokens in the two sentences divided by the logarithmic lengths of the two sentences:

$$Similarity(S_i, S_j) = \frac{|\{t_k | t_k \in S_i \wedge t_k \in S_j\}|}{log(|S_i|) + log(|S_j|)} \tag{2.5}$$

where t_k refers to a token. While node weights are updated as part of the iterative algorithm, edge weights are set upfront and remain constant.

Chasin et al. (2014) assign each event the TextRank score of the containing sentence. The similarity function used slightly differs from the one in the original TextRank algorithm, as pairs of sentences in which one of the sentences is short (less than 10 tokens in length) are given a weight of zero. For all other sentence pairs, similarity is calculated based on word overlap, but lemmas are used rather than original tokens.

2.6.2 Results and Discussion

Chasin et al. (2014) tested their approach on a set of 13 history articles which were all taken from a single domain (wars and battles). More importantly, each article considered discusses only a *single* war or battle.

For all TimeML events in these articles, gold-standard labels (important vs. non-important) were first elicited separately from human annotators. During evaluation, however, the presence of a single event with a positive gold-standard label in a sentence

results in all events in that sentence being interpreted as positive examples.[9] The F-score obtained using this set-up was 51.0%.

According to the authors, the reliability of this result is in doubt because agreement between annotators is low. If one annotator's annotations for individual events are treated as ground truth and each of the remaining annotator's annotations are compared to that ground truth, the average inter-annotator F1 score is only 42.6%. The authors state that the creation of the gold standard was complicated by the fact that a single historical event (the beginning of a war) is often represented by multiple TimeML events (such as "war" and "began" in "The war began"), a problem which was discussed in Section 2.2 (cf. page 24). Also, some TimeML events do not correspond to real-world events at all (cf. Section 2.3 on page 25). In such cases, it is difficult or even impossible for human annotators to establish whether a particular TimeML event is important or not.

Another aspect to keep in mind is that in sets of monothematic articles (such as the one on wars and battles used for evaluation), there exist a set of verbs, such as "beat" or "win", which can be assumed to cover *all* articles' key events. In contrast, general history articles such as the ones I investigate in this thesis discuss many different types of events. Approaches like the one of Chasin et al. (2014) may therefore be less suitable for my task. I will test this hypothesis in Chapter 6.

The results obtained may also be influenced by the quality of the event detector Evita (Saurí et al., 2005), which was used to detect events and their properties (such as polarity and aspect). Evaluation of this tool took place on the TimeBank corpus (Pustejovsky et al., 2003b). While accuracy for polarity and aspect was found to be close to 100%, performance for determing the event class is lower (about 80%). Since the overwhelming majority of events belong to class "OCCURRENCE", this number says little about the performance of the system on the less frequent classes, however.

The TempEval shared tasks, which were proposed several years after the publication of Evita, specifically tested the ability of TimeML event recognisers to determine the correct event class, among other subtasks. In the context of these evaluations, it was shown that the performance of Evita on this subtask is much below that of modern event detectors (UzZaman and Allen, 2010).

2.7 Requirements for my approach

Let us now turn to the task investigated in this thesis: the creation of a timeline from a single, general history article.

The existing methods presented in this chapter are not directly applicable to this task, although some of the underlying ideas (such as the presence of important dates mentioned more than once which must be covered in the timeline) could potentially be useful in my setting also.

The works reviewed in Section 2.5 presuppose the existence of a large corpus of history articles, in which important information is expected to be mentioned multiple times. However, an electronic version of such a resource is currently only available for events that happened in the most recent decades, when news articles began to be collected in electronic archives. It is for this reason that algorithms of this type have so far only been evaluated on recent events.

[9] This treatment is necessary since all events in the same sentence are assigned the same feature vector in the SVM model, and all training examples with the same feature vector should have the same label in order to allow the SVM model to converge.

As this thesis focuses on the creation of timelines from *single* history articles, judgments about which entities or events are added to the timeline should be based mostly on the input article at hand (possibly with some general background information to be gleaned beforehand, but not dependent on the existence of massive parallel corpora of event descriptions such as news articles). In some situations, it may not even be desirable to use parallel corpora, as the input article could focus on different events than the ones that are most salient according to the corpus. Consequently, the approaches described in Section 2.5 are not adequate for the task approached in this thesis.

The TimeML-based approaches described in Sections 2.3 and 2.4 extract *all* events and temporal expressions contained in an input text, whereas the objective in my task is to extract only salient content. A problem of TimeML-based methods is that the event definition used does not ensure a one-to-one correspondence between TimeML events and actions or state changes in the real world. In particular, many TimeML event words, such as stative verbs, do not express real-world events at all, and often a single real-world event can be represented by multiple TimeML events in the text.

The limitations of TimeML also apply to the supervised approach by Chasin et al. (2014), which does focus on the identification of important events and which works on a single document, such as my intended approach. Consequently, annotators often find it hard to estimate whether a given event is important or not. There are other important differences between the work of Chasin et al. (2014) and my intended work. Most importantly, the algorithm by Chasin et al. has been designed to work with texts discussing only one type of physical event (such as a war or battle) as opposed to general history topics (such as the history of France). The individual events described in such an article are closely related to each other and often happen within a short time span, since they are all relevant in the context of the single main event. Chasin et al.'s approach is therefore unlikely to generalise across domains. In contrast, the more general history articles that I consider (such as "History of Austria") contain a large number of events unrelated to each other. They often span long time periods (up to thousands of years), include abstract events as well as different types of concrete events, and provide an overview of the entire history of, for instance, a country or a field of science. The topical focus of such articles can also change as the narrative moves ahead in time.

The approaches I will discuss in this thesis are different from the aforementioned lines of work in various ways. In my setting, a timeline is created by selecting a limited amount of content from a single history article which spans the entire history of a concept.

I start by describing an initial experiment (Chapter 3) that is inspired by this type of timeline generation, although only a sub-task of full timeline generation is addressed here. In particular, the experiment is aimed at selecting significant historical figures that should be part of a timeline on a given topic. I use a supervised approach based on support vector machines to solve this task. As no suitable training data was available, I construct a corpus of timelines and textual articles from existing data on the web. Using this resource, a classifier is learned which identifies historical figures of importance. Due to the size of my corpus, the classifier can make use of lexicalised features.

The remainder of the thesis discusses the full task of content selection for timeline generation, i.e. the selection of a limited number of events in an input article. Algorithms that perform this task will be presented in Chapter 6. However, since the evaluation of full timeline generation is more problematic than assessing a ranking of historical figures, I will first present a comprehensive evaluation methodology for timeline generation in Chapter 5. The main advantage of this evaluation methodology is that the necessary

annotation of human-written timelines is a one-time effort. Therefore, the evaluation of new timeline generation algorithms comes at no further cost. Chapter 6 then presents two types of timeline generation algorithms. Uninformed methods exploit information about section structure as well as the presence of dates. Informed methods either rely on unsupervised features (such as the article's subject area, syntactic links between events and dates and co-occurrence constraints between events) or use an annotated training corpus.

2.8 Chapter summary

In this chapter, I described existing work on timeline generation. We saw that radically different approaches are conflated under the label "timeline generation". I gave an overview of the most salient approaches in each tradition and described why none of these existing families of approaches are applicable to the task investigated in this thesis. I also outlined the remit of the individual chapters in the remainder of the dissertation.

Chapter 3

Identifying significant historical figures

This chapter describes an initial experiment which I conducted in order to investigate whether a supervised machine learning approach could potentially be used to address the task of timeline generation.[1] Concretely, I use an SVM classifier to identify significant historical figures that should be mentioned in a timeline on a given topic. While this task looks straightforward to solve for well-known figures such as Winston Churchill or Barack Obama, which are mentioned millions of times on the Web, it is not an easy task for historical periods that lie further in the past (e.g., Antiquity or the Middle Ages), as well as for articles about less well-known topics.

Knowing which people to include in the timeline promises to be a good starting point for constructing a full event timeline, as people play a central role in most historical events. The aim of the method described in this chapter is to assign a score to each historical figure in a history article. This score reflects the importance of that person for a timeline representing the input article. A downstream system (not discussed here) could then create timeline entries for events in which the most salient historical figures are involved. The produced ranking of historical figures could hence potentially be used as a content selection component of a timeline-specific single-document summariser.

The chapter is structured as follows: Section 3.1 discusses relevant literature on named entities in the context of summarisation. Section 3.2 gives a high-level overview of the experiment described in this chapter. In Section 3.3, I describe a novel corpus which links history articles on Wikipedia to corresponding timelines downloaded from the Web. This corpus will be used to test algorithms on the task of identifying significant historical figures. In Section 3.4, a supervised approach for identifying important historical figures is presented, including the features used. In Section 3.5, a competitive baseline is introduced, which uses the frequency of mentions of historical figures in text. In Section 3.6, I evaluate my approach using the corpus presented in Section 3.3. I first describe the evaluation metrics used, and then show that the proposed method significantly outperforms the baseline. Section 3.7 reports some observations with regard to the experiment which are relevant for the work in the remainder of the thesis. In particular, constructing a corpus by matching data from different sources poses a potential problem for data quality, and the

[1]The work presented in this chapter has been published previously (Bauer et al., 2014). The third author of the publication, Thore Graepel, provided valuable advice in the early stages of the project, but was not directly involved in the research reported in the present chapter.
The evaluation resource described in this chapter is available from the author upon request.

necessary cleaning steps may be as costly as obtaining training data using human timeline writers, a solution that at first glance may look more time-consuming. I conclude that the approach to timeline generation presented in this thesis (cf. Chapters 5 and Chapters 6) should not be based solely on the corpus presented in Section 3.3.

3.1 Motivation and related work

The decision to focus on the identification of important historical figures initially, before turning to timeline generation in Chapters 4 to 6, is motivated by the interpretation of timeline generation as a summarisation task. A range of existing approaches to single- and multi-document summarisation have investigated the importance of named entities for both summarisation and generation (Paice, 1990; Otterbacher et al., 2002; Siddharthan et al., 2011). Such studies rest on the insight that named entities in text differ in their *information status*. Prince (1992) proposed a classification of named entities according to their information status (often called *givenness hierarchy*) that is widely used in Computational Linguistics. Although other classifications exist (Gundel et al., 1993; Grosz et al., 1995), I will only present the hierarchy introduced by Prince.

The first distinction considers the hearer's (or reader's) assumed familiarity with a named entity in question. Entities that the hearer is expected to be familiar with at the time of reading are called *hearer-old*, while other entities are said to be *hearer-new*. A second classification assesses the status of a named entity with respect to the discourse model of the preceding text. *Discourse-old* entities have already been evoked in the prior discourse stretch, while *discourse-new* entities have not. A number of named entities fall outside this binary distinction in that their existence is *inferrable* from the context, i.e. the preceding text is assumed to have introduced these entities implicitly, relying on the reader's world knowledge. For instance, the existence of an oven can be inferred in a statement such as "Linda was baking a cake." It is thought that the reference to the action of baking taking place implicitly introduces the oven used for this action into the discourse stretch.

The information status of named entities in text is of particular interest for the task of creating a summary. This is because the surface form used to refer to an entity in text (commonly called *referring expression*) is often influenced by the entity's information status. Intuitively, entities that are known to the reader at a given location in the text require a less verbose explanation, while references to discourse-new and hearer-new entities often contain additional information needed to uniquely disambiguate the intended referent (Dale, 1992), such as appositions or (in the case of persons) a job description or profession.

These differences in wording pose considerable problems in text summarisation. Given that a summary is typically created by concatenating short text snippets from one or multiple source texts, the information status of a named entity in the resulting summary is often different from that in the source. If the original referring expression is not compatible with the information status in the summary context, the created text may be unintelligible (if essential information is missing) or needlessly repetitive. In the latter case, the summary is also likely to be less informative than it could be, since other salient information has to be omitted due to the word limit imposed. Summarisation algorithms designed to rectify such phenomena must therefore be able to manipulate the extracted text fragments such that any referring expressions are in line with the corresponding entity's information status *in the summary* (rather than in the source text).

Siddharthan et al. (2011) conducted a corpus study to identify typical properties of referring expressions for discourse-new and discourse-old entities, respectively. The focus of this data analysis was on references to people in news stories. This is in contrast to comparable exercises that focus, for instance, on the main characters of certain Wikipedia articles (Belz et al., 2009). Based on this analysis, they proposed an algorithm for the task of creating discourse-new and discourse-old entities in the context of multi-document summarisation. Using a small amount of labelled data, they also learned classifiers which distinguish hearer-new from hearer-old entities, and *major* from *minor* characters. The three aforementioned distinctions were then used in a comprehensive algorithm for creating referring expressions automatically.

Similar studies have been conducted for other genres. For instance, Nissim et al. annotated a corpus of English conversations and showed that a decision tree classifier outperforms a rule-based approach on the task of assigning one of three information status categories to the named entities evoked in the discourse (Nissim et al., 2004; Nissim, 2006). Rahman and Ng (2011) later extended this approach with lexical and syntactic features.

Markert et al. (2012) went beyond the aforementioned corpus studies in that a collective classification algorithm was used to infer the information status jointly for multiple named entities. This approach was shown to outperform approaches based on local classifiers alone, such as the one by Nissim (2006). Secondly, they proposed a more fine-grained annotation scheme for the information status of entities in news articles. Such a detailed classification provides important cues for downstream tasks, notably the identification of bridging anaphora (Hou et al., 2013, 2014). Knowledge about these anaphora can in turn inform models of local coherence, in particular the entity grid (Barzilay and Lapata, 2008), which has been used to assess summary quality. Also following the intuition of Markert et al. (2012), Rösiger and Teufel (2014) proposed a fine-grained annotation scheme for scientific articles and assessed the performance of an existing co-reference resolver trained on this task using domain-specific labelled data.

While the aforementioned works do not address the problem of identifying salient historical figures for timeline generation, certain intuitions underlying these studies are relevant in the context of the present chapter. In particular, Siddharthan et al. (2011) show that referring expressions in news articles, and hence the immediate context of person names, also depend on whether the referenced entity is perceived as important by the author of the news story. For instance, the number of relative clauses and copula, as well as the presence of certain pre-modifiers, seem to be indicative of whether an entity is a figure of *major* or *minor* importance. The definition of importance used by Siddharthan et al. is specific to news articles, since a major person there must be salient in the context of the *entire* news story. In contrast, history articles typically cover a wide range of different topics and historical figures. Even salient persons are therefore likely to be mentioned only in a small part of the input document.

For the task addressed in this chapter, I therefore do not make direct use of existing works such as Siddharthan et al. (2011). However, taking heed of their results, I develop a machine learning system which makes extensive use of features in the immediate textual context of name mentions in order to predict the salience of historical figures.

3.2 Overview of the approach

The supervised approach described in this chapter produces a ranked list of names, given an input article. The n top-ranked names are selected for the timeline, where n is the

desired length of the timeline. To construct this ranking, each of the persons mentioned in a textual article is assigned an *importance score* which indicates how important that person is for a timeline on the subject.

To identify mentions of person names, I will use a named entity tagger. Consider the following sentence taken from an example Wikipedia article: "(...) the Hohenstaufen empire under *Frederick I Barbarossa* reached its peak in (...) the marriage of his son *Henry* (...) to *Constance* (...)." All the strings in italics have been recognised as persons by the named entity tagger. However, as the same person may be mentioned more than once in the same article, evaluation will be performed on the level of persons (types) rather than based on named entity mentions in text (tokens).

My method assumes that the surface text in a history article gives cues as to which content is important enough to be contained in the corresponding timeline. Intuitively, features which might be helpful for this task include lexical cues (e.g. certain verbal constructions or adverbs underlining the importance of events), syntactic cues (such as whether a verb in question is mentioned in the main clause of the sentence), document structure (for instance, where in a section important events tend to be mentioned), and semantic cues (such as co-occurrences of the name of a historical figure and the title of the source article).

3.3 Corpus construction

I now turn to the description of my new corpus, which I will use for training and evaluating methods that identify important historical figures in history articles.

History timelines as defined in Chapter 1 are lists of self-contained event descriptions, each with a date associated. A large number of such timelines exist on the Web. For many timelines, it is also possible to find one or more history articles that discuss the same topic as the timeline. If such pairs of articles and timelines are available, it is possible to interpret the presence or absence of certain content from the gold-standard timeline as an indication that a particular historical figure in the source article is important or not. However, no corpus of timelines and textual articles that was of acceptable size and quality existed. I therefore compiled such a resource myself from existing data on the Web.

The main challenge in constructing a corpus is to identify articles of acceptable quality, and timelines that can be assumed to describe the same subject matter as a given history article. Once suitable timeline/article pairs have been identified, further problems arise: The name of a person may be spelt in different ways in a history article and a corresponding timeline. For instance, a given history article may contain the name "George H. W. Bush", while the corresponding timeline refers to the same historical figure as "George Bush Senior". Such and other inconsistencies may adversely affect the quality of the resulting corpus and, by extension, of any evaluation performed using the corpus. Care must therefore be taken during corpus construction to ensure that the problems described are minimised.

3.3.1 Selection of articles and timelines

To construct a corpus of history articles and corresponding timelines, a search for websites containing history timelines was performed using a standard search engine (Google).

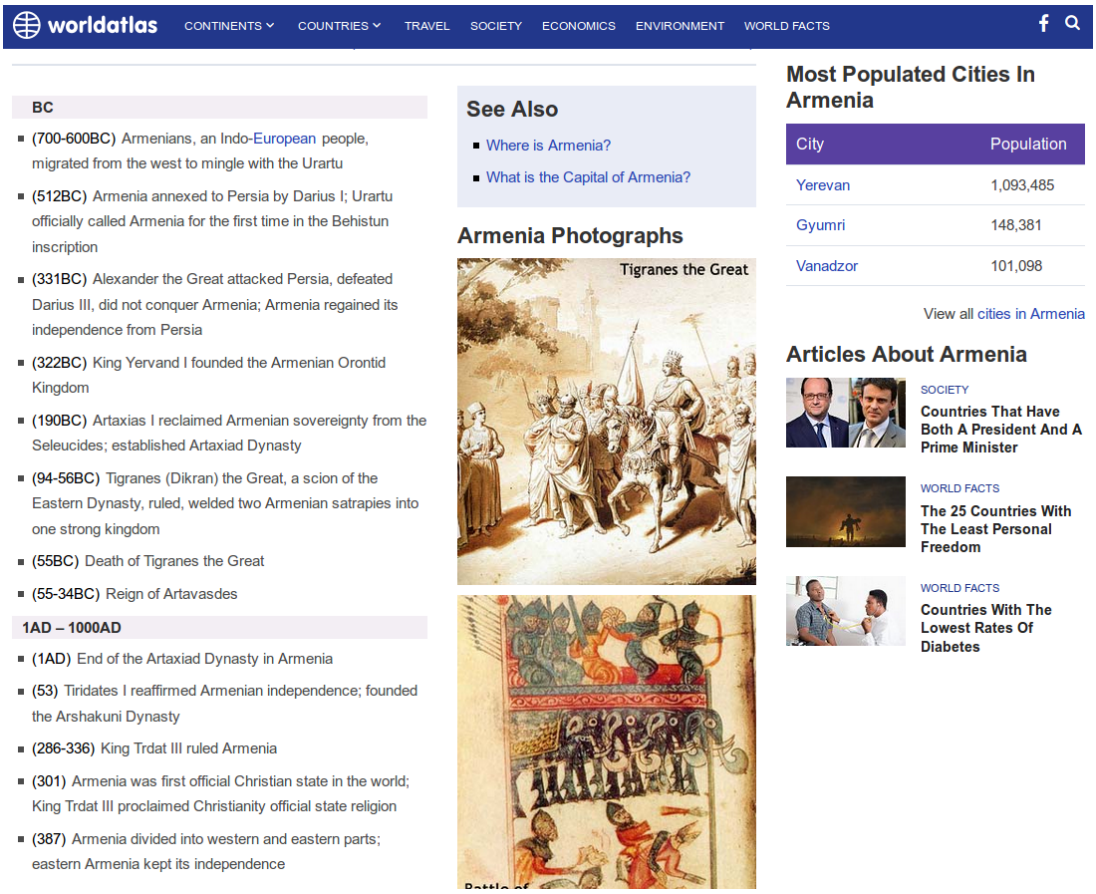


Figure 3.1: Example of a timeline downloaded from *WorldAtlas*.

3.3.1.1 Timelines

Figure 3.1 shows an example of a timeline downloaded from the website *WorldAtlas*. In general, there are a small number of specialist websites that contain a large number of timelines, and a "long tail" with only a few timelines each. Processing these web pages is time-consuming, as the timelines have to be extracted from standard HTML markup, and this markup differs from website to website. For instance, some pages contain a table in which the first column gives the date of the event and the second column contains the event description. On other pages, a date and the corresponding event description are separated by a semicolon; and so forth.

Given these complications, it is preferable to focus on a few websites with a higher number of timelines, and to write scripts that can extract hundreds of them automatically, because each group of pages will follow the same markup pattern. To do this, I compiled a list of suitable websites. All websites containing more than 100 timelines were considered for further processing. Each timeline was transformed into a machine-readable list of (date, timeline entry) pairs by a specialised script written for that website. An example of such a list is shown in Figure 3.2. In total, 1333 timelines were processed in this way.

Manual corrections had to be made for a considerable number of timelines, as occasionally the processed timelines had inconsistencies such as single timeline entries spanning multiple lines; empty extra columns in HTML tables; incorrect use of special characters; and so forth. A further 9 timelines from three sources (*datesandevents.org*, *Historymole* and *The Guardian*) were processed and added entirely manually.

In addition to publicly available websites, a further source was used: all timelines

Date	Timeline entry
700-600BC	Armenians, an Indo-European people, migrated from the west to mingle with the Urartu
512BC	Armenia annexed to Persia by Darius I; Urartu officially called Armenia for the first time in the Behistun inscription
331BC	Alexander the Great attacked Persia, defeated Darius III, did not conquer Armenia; Armenia regained its independence from Persia
322BC	King Yervand I founded the Armenian Orontid Kingdom
190BC	Artaxias I reclaimed Armenian sovereignty from the Seleucides; established Artaxiad Dynasty
94-56BC	Tigranes Dikran the Great, a scion of the Eastern Dynasty, ruled, welded two Armenian satrapies into one strong kingdom
...	...
1998	Ter-Petrosian resigned over opposition to his efforts to find a compromise with Azerbaijan over Nagorno-Karabakh; nationalist Robert Kocharian elected president
...	...

Figure 3.2: Processed version of the *WorldAtlas* timeline on the history of Armenia.

published on the website `www.historyworld.com`, which were obtained in the form of a spreadsheet directly from the author, Bamber Gascoigne, who is well-known as the original quizmaster on the TV programme "University Challenge". A problem was that this data was given as a single list of events shared across multiple timelines, rather than as self-contained timelines. For instance, an event like the French Revolution is part of the timelines on French and European history alike (indicated by two subject labels for this timeline entry). I transformed this list of entries into individual timelines by grouping together all entries with the same label. This implies that some of the final timelines share a number of entries. The transformation resulted in a further 246 timelines. In total, 1588 timelines were collected. Figure 3.3 details, for each data source, the number of timelines processed.

As a final step, timelines from different data sources which discussed the same topic were merged manually into a single timeline containing the union of all timeline entries. This decision was based on the intuition that any timeline entry, whether it occurs in all timelines present or not, describes an important event.

3.3.1.2 History articles

The next step consisted in matching the timelines with corresponding textual Wikipedia articles discussing the same topic. For instance, Figure 3.4 shows a section of the Wikipedia article "History of Armenia". This article can be matched to the timeline on Armenian history shown in Figure 3.1.

I identified matching articles manually. In the general case, only Wikipedia articles whose title starts with "History of" were considered. In a small number of cases, no such article was available, but the topic's main article (i.e. an article without "History of" in the title) was similar to a "History of" article, i.e. it focused on the topic's history and was structured chronologically. In such cases, this main article was matched to the timeline.

In order to ensure that the timeline and textual article discuss the same topic, the titles

Website	# Timelines
Wikipedia	442
timelines.ws	383
Worldatlas	308
Historyworld	246
BBC	200
datesandevents.org	5
Historymole	3
The Guardian	1
Total	**1588**

Figure 3.3: Number of timelines processed for each website.

> Ter-Petrosyan was forced to step down in February 1998 after advocating compromised settlement of the conflict over Nagorno-Karabakh which many Armenians regarded as undermining their security. Ter-Petrosyan's key ministers, led by then-Prime Minister Robert Kocharyan, refused to accept a peace plan for Karabakh put forward by international mediators in September 1997. The plan, accepted by Ter-Petrosyan and Azerbaijan, called for a "phased" settlement of the conflict which would postpone an agreement on Karabakh's status, the main stumbling block. That agreement was to accompany the return of most Armenian-controlled Azerbaijani territories around Karabakh and the lifting of the Azerbaijani and Turkish blockades of Armenia.

Figure 3.4: Sample paragraph from the history article "History of Armenia".

of an article and a corresponding timeline must contain the same concept (e.g. "History of Croatia" and "Timeline of Croatian history"). In most cases, there is also a link at the top of the timeline page which points to a corresponding textual history article (such as: "To read about the background to these events, see *History of France*"). This suggests that an independent user of Wikipedia has recognised the timeline page as being a representation of the content in the textual article. In total, 668 articles could be matched to one of the timelines in the corpus.

3.3.2 Matching person names in timelines and textual articles

A simple way of evaluating my approach would be to use a named-entity classifier to identify mentions of persons in the gold-standard timeline, and to assume that each such named entity corresponds to a separate historal figure. Identifying individual instances of named entities (people, organisations, locations, dates and others) in a given input text is straightforward using tools such as the Stanford CoreNLP suite (Manning et al., 2014). The task of an algorithm would be to select a subset of all person mentions contained in a corresponding textual article. The algorithm would be penalised for each named entity in the gold-standard timeline that it fails to select.

However, evaluation based on named entities alone would not be meaningful, as there is no one-to-one correspondence between historical figures and named entities. Different spellings of the same name, missing diacritics and similar low-level issues may result in failure to match all mentions of the same historical figure to each other. Such effects lower the quality of evaluation and can also result in legitimate positive training examples

being treated as negative examples, which is likely to have a detrimental effect on system performance at test time. For instance, the name of King Ludwig II. might be spelt "King Ludwig" in a history article and "Ludwig II." in the corresponding timeline. An algorithm that correctly selects "King Ludwig" in the textual article would be penalised, because "King Ludwig" is not contained in the timeline. This is an unsatisfactory situation that should be avoided.

There is also a problem if both the timeline and the history article contain the two named entities "King Ludwig" and "Ludwig II.", respectively, but the two names have not been recognised as referring to the same person. An algorithm classifying both names as important would then be doubly rewarded, although it has recognised only one important historical figure correctly. Due to these problems, names that refer to the same person need to be matched in order for evaluation to be meaningful.

A manual analysis of the named entities in both the timelines and the textual articles suggested that such inconsistencies were widespread. For instance, the politician "Robert Kocharian" mentioned in the last timeline entry in Figure 3.2 (cf. page 48) is referred to as "Robert Kocharyan" in the corresponding history article (cf. Figure 3.4 on page 49). This problem is a disadvantage of merging documents from multiple sources into a single corpus.

I mitigate the effect of the aforementioned problems by using the concept of *name sets*. A name set is defined as the set of all (unique) name variants that refer to the same historical figure (e.g. "Clinton", "Hillary Clinton", "Senator Clinton") in a given article or timeline. Each name variant can occur in multiple named entities in the text. Algorithms have access to the named entities in the source article and their context. However, evaluation will be performed exclusively on the level of name sets. In this way, an algorithm selecting the same historical figure multiple times receives only a single point. Ensuring that this is the case is particularly important for persons mentioned many times in a history article, which would otherwise receive disproportionate importance.

Name sets are constructed separately for each timeline and history article. For each name set in the history article, I then a create a link to the name set in the timeline that refers to the same historical figure, if such a name set exists. An illustration of the desired output can be found in Figure 3.5.

The creation of name sets from name mentions is related to the well-known task of *entity linking* (also known as *named entity disambiguation*). Entity linkers map name mentions to entities defined in a knowledge base. Numerous algorithms for solving this task have been proposed in recent years; overviews may be found in Shen et al. (2015) and Carmel et al. (2014).

However, I did not use an off-the-shelf entity linker, as the matching attempted here is different from standard entity linking. For my task, the objective is to link two names to each other if they refer to the same person, independently of whether that person is referenced in a knowledge base. In particular, mapping names to entities in a knowledge base is not a requirement. However, if a knowledge base exists, it can still be useful for detecting co-referring mentions. For instance, it may provide information about different spellings of the name of a well-known historical figure in different languages.

A further difference is that publicly available entity linkers such as AIDA (Hoffart et al., 2011) operate on single texts or even on a single sentence only. My case is special in that entities need to be linked across documents. Here, one can exploit the fact that a timeline and a textual article on the same topic can be expected to discuss a similar set of historical figures. It is unlikely, for instance, that a mention "James Baker" in a

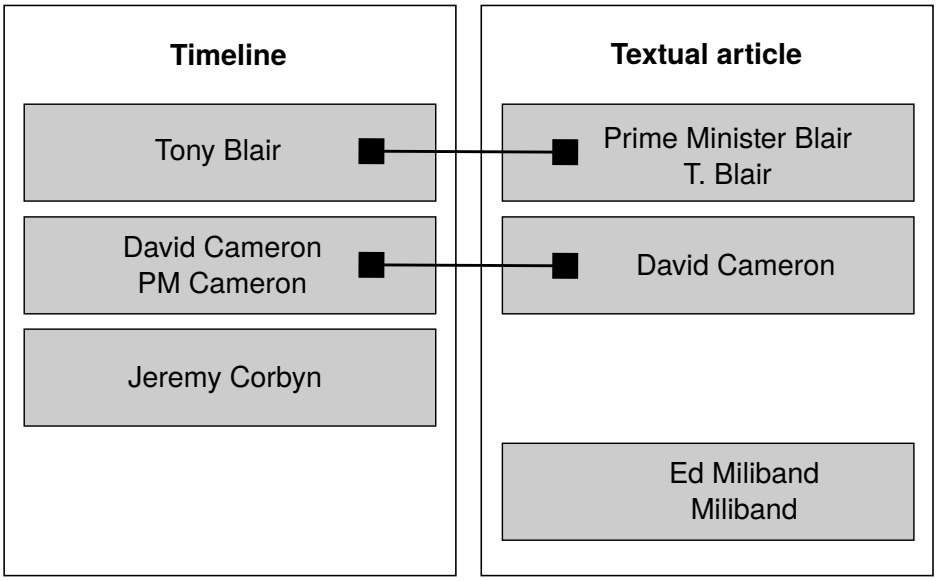


Figure 3.5: Name sets in timelines and textual articles.

history article and a mention of this name in the corresponding timeline refer to different persons.

To construct name sets, I use the following heuristic, which is applied to a timeline and a history article on the same topic: The first step is to identify all named entities in a timeline and the corresponding history article using the Stanford named-entity recogniser (Manning et al., 2014). The named entities are then divided into three groups, depending on their length measured in words. The first group contains all single-word names, which generally are last names ("Blair"). The second group contains all two-word names, which are mostly combinations of first and last names ("David Cameron"). The third group contains longer names. These often contain titles or middle names ("David William Donald Cameron"). Names were grouped in this way to facilitate the matching of full names to their short-hand versions in later stages of the heuristic.

The procedure then first considers two-word entities, i.e. those which can be expected to consist of a first and a last name. For each of them (both in the timeline and the textual article), a new name set is constructed in the first instance.

The next step is the creation of name variations for all (two-word) names in the existing name sets. An initial comparison of the names used in the timeline and a corresponding textual article revealed that a large number of matching errors can be attributed to different writings of one and the same person name – umlauts may be missing; diacritics may have been omitted; and so forth. To account for these phenomena, further versions of all names that contain such special characters are created and added to the relevant name set. For instance, for "François Mitterrand", a different version "Francois Mitterrand" is created. If more than one of these phenomena occurs, I use the superset of all possible changes as the name set. Hence, the name set constructed from "Alex Cartañá" (a name which contains two special characters) will be extended by the alternative spellings "Alex Cartaná", "Alex Cartaña" and "Alex Cartana". Further to that, I add additional variants with prefixed nobility titles (e.g. "Emperor Franz Ferdinand" for "Franz Ferdinand"). For this, I use a list of nobility titles available on Wikipedia. I also add further variants obtained from the YAGO knowledge base (Suchanek et al., 2007), if the current mention can be unambiguously matched to a single YAGO entity. For instance, for the named

entity "David Cameron", all further name variants contained in YAGO (such as "David William Donald Cameron" and "David W. D. Cameron") are added to the name set.

A merging step then ensures that if any of these alternative spellings have already occurred elsewhere in the document, they will not form a new, separate name set. Instead, all these named entities will be merged together into a single name set and hence be assumed to represent the same historical figure.

In the next steps, name sets from the timeline and from the textual article that represent the same historical figure are matched to each other. The procedure ensures that a name set on one side (timeline or textual article) is linked to a name set on the other side whenever they have at least one name in common, and that no name set is linked to more than one name set on the opposite side. The result is a structure like the one shown in Figure 3.5, where only a subset of all name sets in the timeline and the textual article are linked to a name set on the opposite side.

Finally, the other two groups of named entities are merged with the existing name sets. For multi-word names, I optionally remove any middle names (all tokens other than the first and the last). In this way, "David William Donald Cameron" can be matched to "David Cameron". Single-word names are compared to the last token of any existing name in any of the existing name sets. This allows for "Juppe" to be matched to "Alain Juppé".

Figure 3.6 gives a summary of the heuristic. While this heuristic is not perfect, I expect it to cover the vast majority of cases, and to produce a number of name sets that is considerably lower than the number of named entities. I verified the effectiveness of this heuristic by manually analysing all name sets created for the 20 articles that form the development set, as I will describe in Section 3.3.4. In particular, I analysed whether all name variants in a name set refer to the same historical figure, and whether two name sets that are linked to each other represent the same historical figure. I found that the heuristic successfully corrected a large number of errors that would have resulted from a naive processing based on the surface strings of named entities alone. One can therefore assume that this heuristic has made the evaluation framework more accurate.

Consider the example shown in Figure 3.7 (see page 54), which shows a number of example name sets for the Wikipedia article "History of Armenia" and the corresponding timeline on *WorldAtlas*. Although the spelling of the name of Robert Kocharyan in the history article is different from that in the timeline (as shown in Figure 3.1 and Figure 3.2), all occurrences of the person's name are associated with the same name set as a result of the heuristic described above.

3.3.3 Filtering of articles

Some article/timeline pairs are not suitable for processing and should be excluded from the corpus. I will now describe how such pairs are identified.

3.3.3.1 Problems

When constructing the corpus, I assume that each gold-standard timeline contains a selection of content from the corresponding textual article. In particular, it should be necessary to perform information reduction in order to construct the timeline based on the article.

As described in Section 3.3.1.2 (see page 48), the pairs of timelines and articles are selected such that both documents can be expected to discuss the same topic. However, in

1. Identify all named entities using the named-entity recogniser by Manning et al. (2014).
2. Automatically divide the named entities so identified into single-, two- and multi-word named entities.
3. For each two-word named entity:
   (a) Create a new empty name set and add the named entity's string representation to it.
   (b) Create and add alternative versions of this original two-word name:
      - a version of the original name with all umlauts replaced by the corresponding vowel;
      - a version of the original name with all umlauts replaced by the corresponding vowel followed by 'e' (i.e. 'ae' for 'ä');
      - a version of the original name with all diacritics removed;
      - versions of the original name with titles of nobility added or removed;
      - all alternative mention names retrieved from the YAGO knowledge base;
      - the superset of these changes.
   (c) If any of the names that are now in the name set is contained in one of the other name sets already created, merge the two name sets.
4. For each name set created for the textual article:
   (a) If any of the names in a name set constructed for the corresponding timeline is contained in the current name set, create a link between the two name sets.
   (b) If further name sets constructed for the timeline contain one or more names from the current name set, merge all these name sets (and maintain the link to the name set created for the textual article).
5. Repeat step 4 for each name set created for the timeline.
6. For each multi-word name in the timeline and the textual article:
   (a) Try to match it to an existing name set (as is or by removing middle names).
   (b) If this is not possible, create a new name set for this name.
   (c) Repeat step 4 for the name set to which the name was added.
7. For each single-word name in the timeline and the textual article:
   (a) Match it to an existing name set if it is equal to the last word of any of the existing names in that name set (e.g. match "Trump" to "Donald Trump"). If there are multiple such name sets, pick the one with the higher number of mentions in the relevant text.
   (b) If there is no matching name set, create a new name set for this single-word name.
   (c) Repeat step 4 for the name set to which the name was added.

Figure 3.6: Heuristic for matching entity mentions to name sets.

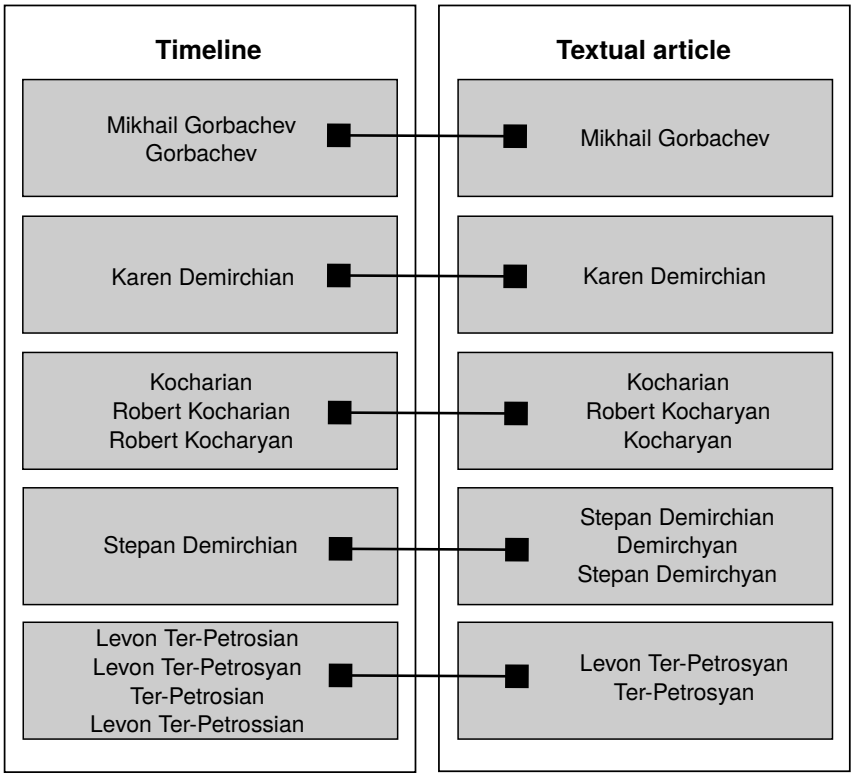


Figure 3.7: Examples of linked name sets for a timeline on the history of Armenia (left) and the Wikipedia article "History of Armenia" (right).

some cases, this criterion is not sufficient, as the timeline contains almost all the content discussed in the history article. In such cases, no information reduction is required in order to construct the timeline from the textual article.

A second problem is that it is not known which textual resources the timeline authors used when writing the timeline. A timeline may therefore contain content that is not mentioned in the corresponding textual article. If there is a large amount of such content, the timeline and textual article may not be a good match, although they discuss the same topic.

Often, there are also more subtle problems. A timeline may cover a different timespan from the one described in the textual article. For example, there are timelines containing only events that happened from the 20th century onwards, while the corresponding history articles start with the Middle Ages.

A textual article may also have been extended by very recent events, but the timeline has not been updated to reflect these changes. All newly added historical figures in the textual article would then automatically be interpreted as negative examples, as they are not contained in the timeline. This problem would negatively affect both the quality of evaluation and the performance of machine learning approaches, which rely on training examples being labelled correctly.

3.3.3.2 Filter rules

In order to find a good set of article/timeline pairs that do not suffer from the aforementioned problems, I use a number of filter rules which are applied after name sets have been created.

The first rule requires that a minimum number of 5 person entities (represented by their name sets) from the textual page be present in the timeline. This criterion aims to eliminate timelines or textual articles that are either too short, or, although their name suggests otherwise, do not exhibit at least some overlap with the textual page. Moreover, cases where there was a processing error are removed in this way.

I also exclude pairs of timelines and textual pages where the proportion of people in the textual article that are also mentioned in the timeline is below 0.1 or above 0.9. If this number is too low, the timeline is likely not to be a good representation of the textual article's content. If it is too high, constructing the timeline from the textual article does not require substantial information reduction. In many such cases, the timeline is very long and therefore likely to mention a large number of unimportant events in addition to important events.

In particular, there are a number of timelines on Wikipedia that are not succinct descriptions of the important events known for a topic, but rather detailed event logs (such as a minute-by-minute list of things that happened before the first airplane hit the World Trade Center in 2001). The individual events mentioned in such a timeline are not necessarily salient, as the objective is to add as many events as possible to the timeline. Such timelines are not adequate for my task, since I assume that the events mentioned in a timeline are salient. In order to exclude such event logs, I apply a length threshold: If the number of names found in the timeline is more than 5 times as high as that of the number of names in the textual article, I assume that the timeline is an event log; the candidate/timeline pair is then removed.

After applying all these constraints, 279 pairs of timelines and textual pages from the original corpus are retained.

3.3.4 Training, development and test sets

The 279 pairs of timelines and textual articles are divided into training, development and test sets. 20 pairs were held out for development, and 30 for testing. Due to the nature of my corpus, it is not a good choice to use a random split (e.g. in a cross-validation setup). The timelines in my corpus have very different lengths, both in absolute terms and in terms of the length ratio of timeline to article. In a sound evaluation, one should make sure that the training and test sets contain a selection of articles representative of the entire corpus.

I therefore opt for the following approach: I first calculate the *entity coverage* of each article with respect to the corresponding timeline. Entity coverage is defined as the proportion of all name sets in the timeline that are also present in the textual article. I then stratify the split with respect to entity coverage. This procedure ensures that each set contains both timelines that are long compared to their textual article counterparts, as well as timelines that exhibit relatively little additional content.

3.4 My method

In this section, I present a system which identifies significant historical figures in history articles. More concretely, the system is trained to identify figures that should be contained in a history timeline on the same topic.

3.4.1 Linguistic processing

The textual articles as well as the timelines in the corpus need to be pre-processed. This includes basic operations such as tokenising and POS tagging, as well as dependency parsing and co-reference resolution. Co-reference chains are created for textual articles only, as timeline entries are assumed to be independent of each other.

In a first step, tokenisation, POS tagging and named entity recognition are performed using the Stanford CoreNLP toolkit (Manning et al., 2014), for both textual articles and timelines. The POS-tagging component of the toolkit is based on a maximum-entropy model (Toutanova et al., 2003), while the named-entity tagger by Finkel et al. (2005) uses Conditional Random Fields (Lafferty et al., 2001) and a Gibbs Sampling (Geman and Geman, 1984) approach.

The textual articles are then parsed by the C&C parser (Clark and Curran, 2007), which relies on the Combinatory Categorial Grammar (CCG) formalism. This parsing paradigm follows the intuition that each word in a sentence can be assigned a lexical category that captures more syntactic information about that word's syntactic role than ordinary POS tags. There are only very few atomic categories in CCG (such as S for sentence or NP for noun phrase), while the rest of the categories are complex, such as $S \setminus NP$. These categories can be thought of as functions; $S \setminus NP$ is a category that takes a noun-phrase (NP) to its left and returns a sentence (S); this category corresponds to an intransitive verb. Since the number of categories that can be assigned to a word is potentially very large, the C&C parser uses a sequence tagger, called supertagger, to assign a limited number of plausible categories to each word before the actual parsing algorithm, which is based on the CKY algorithm (Kasami, 1965), is run. To identify the most likely parses, C&C learns a log-linear model with long-range dependency features. This log-linear model is trained on a special version (Hockenmaier and Steedman, 2007) of the standard Wall Street Journal (WSJ) treebank (Marcus et al., 1993).

Co-reference relations between noun phrases in the textual articles are annotated using the Stanford CoreNLP toolkit, which implements a rule-based approach by Lee et al. (2013).

3.4.2 Named entity scoring

I will now describe how I score name sets by their importance. Each named entity is first scored individually using a support-vector-machine (SVM) classifier (Cortes and Vapnik, 1995). Support vector machines are maximum-margin classifiers: Given a labeled training data set, they place the optimal hyperplane in the space which separates points with different labels optimally, such that each point has the largest possible distance ("margin") to that separating hyperplane.

Basic SVM classifiers are designed for two-class classification problems. By default, they only output a binary prediction (1 or -1). Platt (2000), however, proposed a well-known method for computing probability estimates for individual data points. Chang and Lin (2011) implemented this method in the software packages LibSVM and LibLINEAR (Fan et al., 2008), which I use for my experiment. LibLINEAR is a variant of LibSVM which allows for faster training of models that use a linear kernel. The C and gamma parameters of the SVM classifier are tuned on the development set using *grid search*.[2]

[2]I use the grid search tool provided as part of LibSVM and LibLINEAR.

This procedure automatically determines the best-performing set of parameters in a cross-validation setup.

The features I use will be described in Section 3.4.4. I experiment with a number of different kernels when tuning the algorithm on the development set; this includes linear, radial-basis-function, sigmoid and polynomial kernels, all of which are implemented in LibSVM.

3.4.3 Name set scoring

After individual named entities have been scored, I move to scoring name sets. This is a separate step because the SVM classifier operates on individual named entities, but the final system must assign a single score to each historical figure. This score denotes the general relevance of that figure for a timeline on the same topic, which is independent of the context of a particular mention.

There are several options for calculating the score of a name set. One could use the sum of all scores assigned to the mentions of a name set in text. However, this would result in a bias towards historical figures that are mentioned more frequently than other named entities.

An alternative would be to use the average score of all named entities of a name set, which does not suffer from this problem. A manual inspection of the data suggested, however, that even important persons are not necessarily presented as important every time they are mentioned in text. Often, they are also mentioned in the context of less important events. Such contexts, which are less indicative of a person's importance, should not detract from the general importance of a historical figure. For this reason, the average of all named entity scores is not an adequate choice for the name set score, as it considers all mentions equally.

I therefore decided to use as the score of a name set the maximum score assigned to any mention of that name set in the text. This measure assumes that not all mentions in the text must reflect the person's importance.

Besides a continuous notion of importance of a name set for a timeline, I also define an absolute quality criterion. All name sets that fulfill this criterion are considered relevant for the timeline in this case, while all others should not be added to the timeline. In order for a name set to be considered relevant, at least 50% of all named entities in that name set must have received a score above a threshold value tuned on the development set. This choice too is inspired by the observation that the importance of a historical figure must not be represented in the context of each mention of that figure in text (as described above).

3.4.4 Features

In this section, I describe the features used to train the supervised machine learning system for named entity scoring described in Section 3.4.2.

Three different groups of features are used. The first group is composed of structural features (see Figure 3.8) such as the frequency of a name in the input text, the position of each named entity in the containing sentence, and the position of the named entity's sentence (in the containing paragraph) and paragraph (in the containing subsection).

The reason for considering frequency is obvious: Important historical figures are likely to be mentioned multiple times in the text, as opposed to less important figures which

ID	Description
frequency	the frequency of each name in the history article
frequency_dividedByAllNEs	the frequency of each name in the history article divided by the total number of named entities
indexOfWordInSentence	the position of the named entity in the containing sentence (word index of the first word of the NE)
indexOfCurrentStanfordNE	the index of the named entity in question in the list of all named entities in the current sentence
indexOfCurrentSentence	the index of the sentence that contains the named entity in question in the list of all sentences of the current paragraph
indexOfCurrentParagraph	the index of the paragraph that contains the named entity in question in the list of all paragraphs of the current subsection
numberOfNEsInSentence	the number of named entities in the containing sentence
occursInHeader	whether or not the current named entity is in the Wikipedia document's header
entity_linked_to	1 if the person name in the Wikipedia article contains a hyperlink to the person's Wikipedia article
numberOfDifferentSpellings	the number of different writings of the current name (i.e. the size of the corresponding name set)

Figure 3.8: Structural features.

ID	Description
unigrams	all unigrams (lemmatised) in a window size of +/- 3 around the mention, e.g. "abdicate"
unigrams_dist	all unigrams (lemmatised) in a window size of +/- 3 around the mention, annotated with the distance in words from the NE, e.g. "abdicate_+1"
bigrams	all bigrams (lemmatised) in a window size of +/- 3 around the mention, e.g. "abdicate_after"
bigrams_dist	all bigrams (lemmatised) in a window size of +/- 3 around the mention, annotated with the distance in words from the NE, e.g. "abdicate_after_+1"
postags	all POS tags in a window size of +/- 3 around the mention, e.g. "VBD"
postags_dist	all POS tags in a window size of +/- 3 around the mention, annotated with the exact distance, e.g. "VBD_-1"

Figure 3.9: Lexical features.

tend to occur only once. The features whose names start with "indexOf" reflect intuitions about where in the text humans tend to place important information (names of persons, in this case). For instance, a historical figure which has shaped the history of an entire historical period may be more likely to be mentioned early on in a paragraph.

A further feature considers the number of named entities in the containing sentence. Sentences with a high number of named entities often contain an enumeration of names. It is unlikely that a person mentioned in such a list is presented as particularly salient in the context of a historical event.

The feature "occursInHeader" specifies whether the person name in question is present in the Wikipedia article's header section. This header section is likely to contain salient persons, as it often summarises the most important information contained in the main article text.

I also examine whether any of the person's mentions in text is connected to a separate

ID	Description
dep_degree1_in	the types of all incoming dependency edges of any word that is part of the named entity (e.g. "dobj")
dep_degree1_out	the types of all outgoing dependency edges of any word that is part of the named entity (e.g. "xcomp")
dep_degree2_in	the types of all incoming dependency chains of length 2 starting from any word that is part of the named entity (e.g. "dobj_det")
dep_degree2_out	the types of all outgoing dependency chains of length 2 starting from any word that is part of the named entity (e.g. "xcomp_det")
dep_is_local_or_global_subject	indicator features that indicate whether any of the words in the named entity are the subject of any clause of the current sentence or of the main clause
dep_is_local_or_global_object	indicator features that indicate whether any of the words in the named entity are the object of any clause of the current sentence or of the main clause
dep_numOfsisterDeps	the number of "sister dependencies" of the same type (i.e. dependencies of the same type starting from a "parent token" of the current token in the dependency graph). This measures how long the list of names is that a name occurs in), and categorical features that fire if there are at least one, two or three sister dependencies.

Figure 3.10: Syntactic features.

Wikipedia page describing only that person via a hyperlink. The presence of such a hyperlink may be a cue that the person is important, as it is unlikely that a separate Wikipedia page exists for minor characters that are only mentioned in passing.

The final feature in this group is the number of different spellings that are known for a historical figure following the construction of name sets (as described in Section 3.3.2). A high number of alternative spellings may be caused by the fact that the historical figure is important.

The second group of features (shown in Figure 3.9) exploits linguistic cues. For instance, a word like "abdicate" in the immediate surroundings of a name might point to the fact that the person in question was an important king (otherwise his abdication would not be mentioned explicitly in the text). I use a standard set-up in which features are created for all unigrams and bigrams in a window of +/-3 words around the named entity; one version of the features ignores distance to the named entity in words annotated (and hence only takes into account presence anywhere within this window); a second set of features is distance-specific (i.e. there will be a feature "former_-1" indicating that "former" is the word immediately before the entity, and similarly features "former_-2", "former_+1" etc.). Feature patterns of part-of-speech-tags occurring in a window around the named entity are created in a similar way.

The third group of features (given in Figure 3.10) considers syntactic dependencies between words that are part of the named entity and words in the rest of the sentence. Such features are inspired by intuitions about the way in which important historical figures tend to be described in text. For example, it is possible that such figures are often mentioned as the subject of a sentence because, as persons in power, they are more likely

ID	Description
part_of_corefchain	1 if the current name is part of a coreference chain, 0 otherwise
longest_corefchain	the length of the longest co-reference chain the current name mention is part of
corefchain_min5sentences	1 if the longest co-reference chain is longer than 5 sentences, 0 otherwise
corefchain_min_x_InBetweenSentence	1 if there are at least x sentences between the sentences of the co-reference chain involving a person name, 0 otherwise
corefchain_lengthOfBiggestGapMin_x	1 if there are at least x sentences between any two sentences of the co-reference chain involving a person name, 0 otherwise
corefchain_distSpanSentencesMin_x	the distance (in number of sentences) between the first and last sentences of any co-reference chain the current mention is part of

Figure 3.11: Discourse features.

to have actively performed notable actions. Features are also constructed for all chains of dependencies starting from or ending in a word that belongs to the named entity in question. A final feature ("dep_numOfsisterDeps"), which is based on the same intuition as the structural feature "numberOfNEsInSentence", considers whether the named entity occurs in a list of other named entities, and if so, how long that list is. Persons whose names occur in long lists, such "A", "B" and "C" in "Further important discoveries were made by A, B and C" can be expected to be less important than other persons.

The fourth group of features (shown in Figure 3.11) exploits discourse phenomena, in particular co-reference chains in which the named entity in question is involved. It seems plausible that important historical figures are mentioned repeatedly in a given text in the form of pronouns, as they are likely to have been involved in more than one important event. I also expect historical figures that are mentioned in different parts of a document, possibly with large stretches of text between the individual mentions, to be more important than other persons. I therefore construct features that encode the length of co-reference chains represented by the number of occurrences, as well as features that detect chains spanning a high number of sentences. A further feature is created for chains whose occurrences are far apart in terms of the number of sentences between each of them.

The fifth group of features (given in Figure 3.12) models co-occurrence cues found in the text. For instance, features are created when a person name and the topic of the article co-occur in the same sentence ("conceptName_thisSentence" and "conceptName_allSentences"). In that case, the text may express that the person in question had an impact on the topic of the article. For instance, in "Odoacer's rule came to an end when the Ostrogoths, under the leadership of Theodoric, conquered Italy" (taken from the article "History of Italy"), the co-occurrence of "the Ostrogoths" and "Italy" in the same sentence may hint to the Ostrogoths being central to Italian history. In a similar fashion, the final two features incidate that the article topic (e.g. "Italy") co-occurs with the person name in a sentence in the person's own Wikipedia article ("target_Sentence_withConcept") or in a section title in that article ("target_Title_withConcept"), provided that such an article exists.

ID	Description
conceptName_thisSentence	1 if the name of the concept co-occurs with the name of the person in the same sentence, 0 otherwise ("History of" is removed from the article title, so that concept names are "Germany" or "Biology")
conceptName_allSentences	1 if the name of the concept co-occurs with the name of the name of the person in **any** sentence in the textual document, 0 otherwise
target_Sentence_withConcept	a feature that indicates whether the name of the concept co-occurs with the name of the name of the person in **any** sentence in the person's Wikipedia article (this feature only applies if the above hyperlink exists)
target_Title_withConcept	a feature that indicates whether the name of the concept occurs in a section or subsection title in the person's Wikipedia article (if the name is linked to this article)

Figure 3.12: Semantic features.

3.5 Baseline and semi-oracle results

In this section, I introduce a baseline to which my method will be compared. I also describe how I will analyse the impact of errors made during named entity recognition on the performance of both the system and the baseline.

3.5.1 Baseline

I use a baseline which considers how often a person name is mentioned in the textual article. More exactly, the ranking is provided by the total frequency in terms of named entities summed over all names in a name set. In this way, the baseline can abstract away from different name variants of the same person. Since a high number of mentions intuitively indicates importance, I expect this baseline to perform competitively.

In some cases, two or more name sets have the same total number of occurrences in the text. To resolve these ties, the name set with the earliest occurrence of a named entity in the article text is given preference. This is a reasonable heuristic given that most articles start with an introduction that mentions the most important names.

I also evaluate this baseline when an absolute quality criterion (defined in Section 3.4.3) is used instead of a ranking to distinguish relevant from non-relevant persons. The threshold value required in this case is tuned on the development set in the same way as for my method.

3.5.2 Semi-oracle results

In a pipeline, testing of an individual component can be more informative if follow-on errors from earlier stages in the pipeline are removed. In order to test how well the algorithm and the baseline perform if named entity recognition is perfect, I also created an oracle version of the named entity recognition in which erroneous `PERSON` instances have been removed. I did not add `PERSON` instances which had not been recognised by the system, as this process would have required a large-scale annotation effort. In Section 3.6, where I evaluate my method, the results produced using the corrected named

entity recognition will be shown as "Semi-oracle" (cf. Figure 3.13 on page 65).[3] In contrast, "Automatic" refers to results obtained with the full pipeline, where named entity recognition is performed automatically by the Stanford named entity recogniser.

3.6 Evaluation

Having described my system and the baseline, I now turn to evaluating my approach. Evaluation is performed on all name sets created from the textual article. Historical figures that are only mentioned in the timeline are disregarded for the purpose of evaluation.

3.6.1 Evaluation metrics

I will use two alternative metrics to evaluate my approach.

3.6.1.1 MAP

The first metric is Mean Average Precision (MAP), which is well-known in the field of Information Retrieval (IR). It is based on a number of simpler metrics commonly used for evaluating two-class classification problems. I will describe these metrics first.

In many two-class classification problems, the goal is to predict which out of a number of examples share a certain property. Examples that share the property are called *positives*, while other examples are called *negatives*. In such a setting, two kinds of errors can be made: Positive examples that the algorithm mistakenly classifies as negative are called *false negatives*, while negative examples classified as positive are called *false positives* (fp). Positive examples that were correctly classified are called *true positives* (tp), and correctly classified negatives are called *true negatives* (tn).

The relative frequency of each of these cases in an experiment is used to calculate the well-known evaluation metrics *precision* and *recall*. Precision measures how many of the examples that were classified as positive are indeed positives. It is defined as follows:

$$P = \frac{tp}{tp + fp} \tag{3.1}$$

In contrast, recall measures how many of all positives were classified as positive by the system:

$$R = \frac{tp}{tp + fn} \tag{3.2}$$

The weighted harmonic mean of these two metrics is often used to measure the overall performance of a system. It is called *F-score* and defined as follows:

$$F_\alpha = \frac{P \cdot R}{(1-\alpha)P + \alpha R} \tag{3.3}$$

α is commonly set to 0.5, which results in:

$$F_{0.5} = 2 \cdot \frac{P \cdot R}{P + R} \tag{3.4}$$

In what follows, I refer to $F_{0.5}$ as "F-score", for simplicity.

[3] I use the term "semi-oracle" since the oracle does not cover the remaining steps of name set generation.

Mean Average Precision (MAP) is a measure that evaluates the quality of a ranking of retrieved documents based on multiple precision values. I choose MAP since I want to evaluate the ability of the algorithm to create timelines of arbitrary length, not of a given, fixed length. MAP is calculated as the average over precision values taken at all recall levels, and therefore gives a summary account of the quality of timelines of any possible length.

In an information retrieval setting, Average Precision for a single query q_j is given as:

$$\mathrm{AP}(q_j) = \frac{1}{m_j} \sum_{k=1}^{m_j} \mathrm{Precision}(R_{jk}) \tag{3.5}$$

where m_j is the number of *relevant* documents in the ranking, R_{jk} is the set of ranked retrieval results from the top result until a relevant document d_k, and $\mathrm{Precision}(R_{jk})$ is the precision calculated over the k first documents returned for query q_j. This means that, in order to calculate average precision for a query q_j, one traverses the entire ranking R_{jk} from top to bottom. Whenever a relevant document occurs, precision over all documents retrieved so far is calculated. Finally, all precision values are averaged over the number of measurements taken.

Average Precision gives more weight to higher ranks, since these are used to calculate a higher number of individual precision values in Equation 3.5. This property is desirable as the most highly ranked name sets will be included in timelines of many different sizes, i.e. both in very short timelines that only contain the most important historical figures as well as in longer timelines. On the other hand, persons further down in the ranking will only occur in longer timelines.

In my setting, each name set is interpreted as a document. To measure performance across the entire set of timeline/article pairs Q (which is equivalent to a set of queries in the Information Retrieval setting), I calculate Mean Average Precision (MAP), which averages the AP scores obtained for each of the timeline/textual article pairs:

$$\mathrm{MAP}(Q) = \frac{1}{Q} \sum_{j=1}^{|Q|} \mathrm{AP}(q_j) \tag{3.6}$$

3.6.1.2 Precision, recall and F scores

Both my method and the baseline also define an absolute quality criterion for each name set individually. Name sets that fulfill the criterion are considered (equally) relevant, while all other name sets are considered non-relevant. Based on this binary classification into relevant and non-relevant name sets, I calculate standard precision, recall and F-score values (as described in Section 3.6.1.1) for all name sets micro-averaged across all timeline/document pairs. The threshold score value needed for the absolute quality criterion was tuned on the development set for both the method and the baseline; the optimal values were found as 0.6 and 0.1, respectively.

3.6.1.3 Significance testing

In order to ensure that a difference in performance between the system and the baseline is statistically significant, I use the (paired) Wilcoxon signed-ranks test (Wilcoxon, 1945). Each timeline/document pair in the test set is interpreted as a single data point; hence there are 30 data points in total.

The signed-ranks test is a non-parametric test which is more powerful than the easier sign test, as it takes into account information about the magnitude of the differences between methods for individual data points (Siegel and Castellan, 1988). The test statistic relies on the difference scores d for each data point:

$$d_i = X_i - Y_i \tag{3.7}$$

where X_i and Y_i are the performance scores obtained using two different methods for a data point i. Next, these differences are enumerated in ascending order according to their absolute values, i.e. without taking into account their sign: the smallest difference is assigned rank 1, the next largest rank 2, and so on. Data points where there is no difference between methods (i.e. $d_i = 0$) are discarded, and for cases where two data points have the same (non-zero) value for d_i, the average rank is assigned to each of the d_i. Next, two sums are calculated: T^+ is the sum of all ranks that correspond to positive difference scores d_i, i.e. where method X has outperformed method Y, and T^- sums up the remaining ranks.

To calculate the final test statistic z for sample sizes larger than 15, one can exploit the fact that T^+ is approximately normally distributed, which implies that the test statistic

$$z = \frac{T^+ - \mu_{T^+}}{\sigma_{T^+}} \tag{3.8}$$

is approximately normally distributed too. The mean u_{T^+} and variance σ_{T^+} needed in Equation 3.8 are calculated based on the number of data points, as follows:

$$\mu_{T^+} = \frac{N(N+1)}{4} \tag{3.9}$$

$$\sigma_{T^+} = \frac{N(N+1)(2N+1)}{24} \tag{3.10}$$

where N is the number of data points. The test statistic z can then be used to calculate the p value necessary for deciding whether a difference in performance can be regarded as statistically significant.

3.6.2 Results

The overall results are given in Figure 3.13. For both sets, the machine learning system ("SVM") significantly outperforms the frequency-based baseline. For the test set, the MAP score of the machine learning system is 0.600, while the baseline only achieves a score of 0.525. This improvement is statistically significant according to the Wilcoxon signed-ranks test ($p = 4.408e^{-5}$). For the development set, the machine learning system outperforms the baseline independently of whether the name sets are constructed fully (Automatic) or partly (Semi-oracle) automatically.

The individual results achieved on the 30 articles in the test set are shown in Figure 3.14 (see page 66). The SVM model outperforms the baseline on 23 articles. For the remaining 7 articles, the baseline achieves a higher score. However, the difference between the two scores is negligibly small in 4 out of these 7 cases.

All results shown for my method are obtained using a linear kernel; using a polynomial, a sigmoid or a radial-basis-function kernel results in worse performance. This result is to

Data set	Name sets	System	MAP	P	R	F
Development	Automatic	SVM	0.645	0.543	0.662	0.597
Development	Automatic	BL	0.581	0.492	0.491	0.491
Development	Semi-oracle	SVM	0.666	0.535	0.662	0.591
Development	Semi-oracle	BL	0.602	0.505	0.492	0.499
Test	Automatic	SVM	0.600	0.455	0.639	0.532
Test	Automatic	BL	0.525	0.338	0.612	0.435

Figure 3.13: Overall results.

be expected, as with a large number of features (such as the lexicalised features in my approach), linear kernels are known to work best (Hsu et al., 2003).

3.6.3 Ablation study

In order to understand whether the system's ability to find important historical figures can largely be attributed to a single feature, I performed an ablation test. The development set and the semi-oracle name sets were used for this purpose. In all configurations, the MAP score is above 0.64, which is close to the performance obtained with all features (0.666). The feature whose removal resulted in the biggest drop (below 0.65) was "frequency" (cf. Figure 3.8 on page 58).

I also ran the system with only a single feature enabled, for all features described previously. The performance of all these configurations was below that of the baseline. As was to be expected, using only the single feature "frequency" leads to an overall performance close to the baseline.[4] These results show that there is no single feature that is responsible for the performance improvement.

3.7 Outlook on the rest of the thesis

The results obtained in Section 3.6 demonstrate that a supervised approach, combined with an appropriate corpus of history articles and corresponding timelines, can in principle be used to learn which content is worthy of being mentioned in a timeline. However, the system developed here only produces a ranked list of historical figures; no timelines are created. In what follows, I will therefore discuss whether a similar supervised approach should be taken for the actual task of timeline generation, i.e. the selection of events for the purpose of constructing a timeline.

I start by giving a summary of the difficulties observed while constructing a parallel corpus of timelines and textual articles:

1. **Timeline content missing from the article:** It is often difficult to ascertain to what extent a timeline in the corpus reflects the content of the corresponding textual article. This is a general limitation of a semi-automatic corpus construction along the lines chosen here.

   The problem arises because it is not known which sources were available to the timeline writer(s) at the time of writing. Hence, content that does not appear in

[4]The performance is not necessarily equal to that of the baseline since the baseline additionally includes a way of dealing with ties, as described in Section 3.5 on page 61.

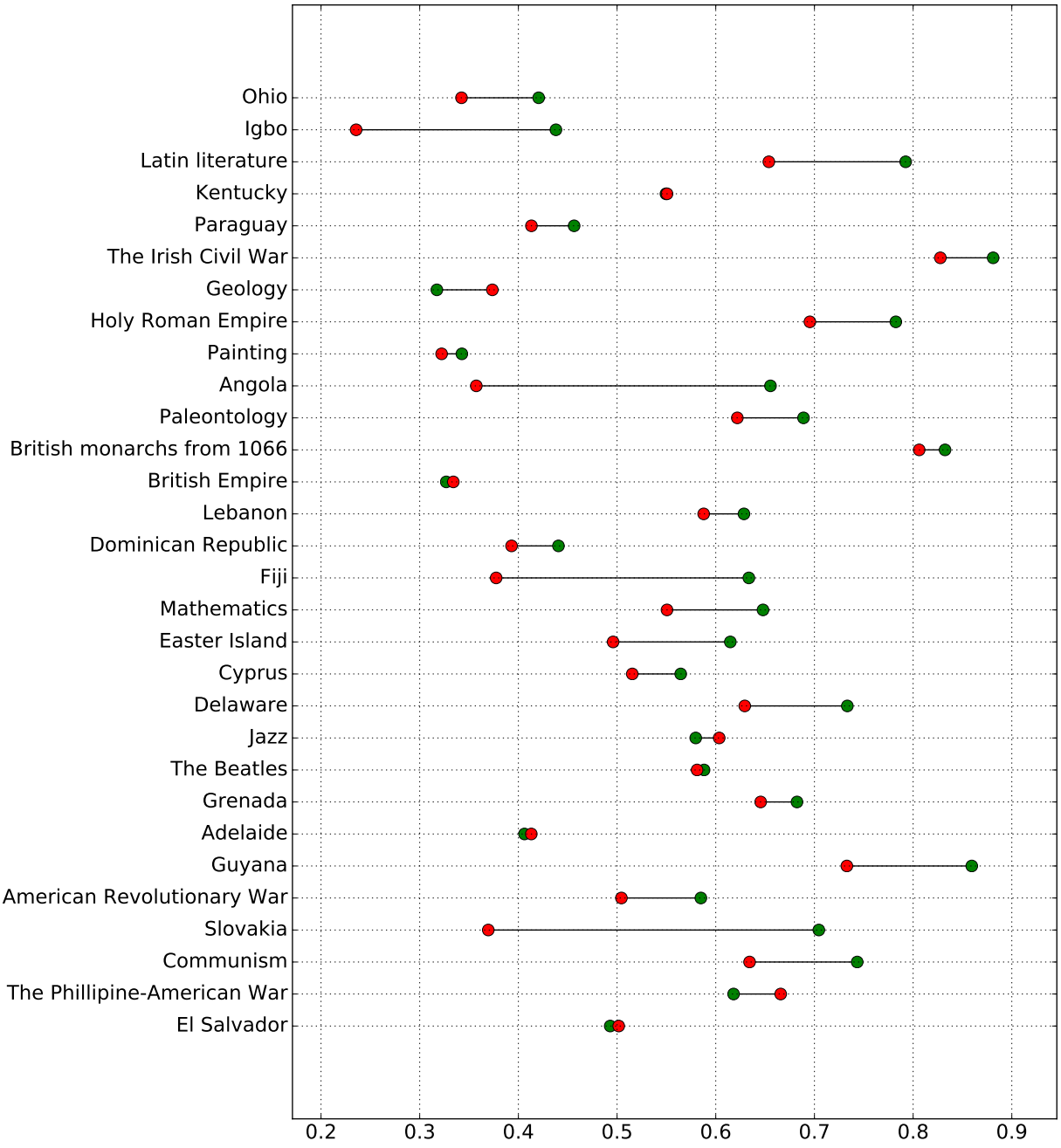


Figure 3.14: Performance of the SVM model (green) and the baseline (red) on the 30 timeline/article pairs in the test set.

the timeline may be missing not because it is irrelevant; the timeline writer may simply not have used the relevant sources.

Reasonable care was taken to mitigate this problem when constructing the corpus described in this chapter, e.g. by using filter rules. I assume that the history article in the corpus was often at least one of the sources used to create the timeline, since Wikipedia is well-known and widely used. Still, in many cases I observed that there is content in the timeline which is missing from the corresponding textual article.

2. **Coverage of different timespans:** The accuracy of evaluation can be negatively affected in cases where a timeline covers a different timespan than the one in the

corresponding textual article. If the timeline merely contains events up to the year 2010 (for instance, because it has not been updated after this), all content in the textual article describing more recent events will automatically be marked as non-relevant during evaluation. In order to guarantee a perfect match between timeline and textual article in terms of the time span described, such mismatches would have be to identified manually for each article/timeline pair in the corpus.

3. **Imbalanced selection of content:** It is not guaranteed that the timeline writer has tried to achieve a balanced selection of important content for the entire historical period covered by the source article at hand. For instance, it is possible that a disproportionately large amount of content was selected from recent periods or from a particular subject area (such as politics).

4. **Different lengths of timelines:** Timelines downloaded from the web have vastly different lengths. This is a problem since with overly long timelines, the fact that an event from the textual article is also contained in the timeline does not necessarily mean that the event is important. For the corpus introduced in this chapter, this problem was mitigated by imposing constraints on the length ratio between a timeline and the corresponding textual article. However, this is only a compromise and not a general solution to this problem.

5. **Low-level noise:** There are often multiple variants of a person's name. If these name variants are not matched to each other, the quality of evaluation is negatively affected. I used a heuristic to perform this matching when creating my corpus, but clearly any such heuristic is imperfect.

6. **Single gold-standard timeline:** There are many history topics for which only a single gold-standard timeline is available on the Web. However, given that content selection tasks are known to be subjective, it would be preferable to use multiple gold-standard timelines.

These problems would be exacerbated with the more complex task of timeline generation, given that one and the same event can be expressed in many ways. In order to match different wordings of one and the same event to each other, a simple heuristic such as the one presented in Section 3.3.2 is not sufficient, as there is no known way of reliably detecting paraphrases automatically. Each event mentioned in a timeline would therefore have to be manually matched to its counterpart in the corresponding history article. This task would be prohibitively expensive even for a moderate number of articles, as for each timeline entry, a human annotator would have to search the entire history article for matching event mentions. Second, it is unclear how an annotation task should be set up. In many cases, an annotator would have to make complicated inferences in order to decide whether two statements are paraphrases of each other. This process is likely to be unreliable. Moreover, an event expressed in a timeline entry might only be mentioned implicitly or in passing in the surface text. It is unclear whether such event mentions should be linked to the corresponding timeline entry or not.

The key criterion for deciding whether the corpus developed in this chapter should be used for timeline generation is whether the huge annotation effort incurred is justified by the outcome. In particular, one would have to be certain that such an annotation study would produce a resource of high quality, which could be trusted to provide a reliable gold standard for the evaluation of timeline generation. The limitations described

above make this seem rather unlikely, and so I did not pursue this approach. Instead, I decided to create a new corpus of timelines written by human timeline writers (described in Chapter 5).

3.8 Chapter summary

In this chapter, I presented a preliminary experiment which can be interpreted as a preparatory step towards timeline generation: the identification of important historical figures in a history article that should be included in a timeline. I first presented a new corpus of several hundred pairs of timelines and history articles. I then developed a way of constructing name sets from the named entities in the two texts. This procedure allows for evaluation of algorithms on the level of real-world entities (persons) rather than based on individual named entities. Based on this matching, I filtered out pairs of timelines and textual articles that do not meet certain quality criteria. I subsequently presented a supervised learning approach which identifies important figures automatically, based on a wide range of features. These include structural features (such as the frequency of a name in the text or the position of the containing sentence in the paragraph), lexical features (such as words in the name's context and their part-of-speech tags), syntactic features (such as the types of syntactic dependencies between the words of the person name and other words in the sentence), discourse features (such as the length of co-reference chains involving the person name in question), and semantic features (such as co-occurrences of the person name with the topic of the article). I showed that this approach significantly outperforms a competitive frequency-based baseline. Finally, I discussed several problems that arise from the construction of a corpus from existing timelines and textual articles found on the Web, such as the presence of entities in a timeline which are not mentioned in the textual article. I concluded that the construction of a parallel corpus in this way would not be a good choice for timeline generation, as many of these problems are exacerbated in this more complex task. In particular, it would be very difficult and time-consuming to achieve a high-quality mapping between multiple mentions of the same event.

In the next chapter, I will formulate the requirements of an evaluation methodology for timeline generation. The chapter starts by reviewing existing work on summarisation evaluation, given that the task can be interpreted as a variant of single-document summarisation. I will then describe desirable properties of an evaluation methodology that combines the advantages of various existing approaches.

Chapter 4

Related work on summarisation evaluation

In the remainder of the thesis, I will investigate methods that create a timeline by selecting *events* rather than named entities. I treat timeline generation as a special *single-document summarisation* task. This chapter is dedicated to the question of how one could evaluate timelines consisting of events, in particular by borrowing from the summarisation evaluation literature.

Single-document summarisation is the process of creating a condensed version of a source document which contains the most salient content from that document and expresses this content in a concise and coherent manner. A timeline is a kind of summary, as it is a compressed version of the original history article. However, timeline generation also differs from summarisation, since a timeline is not free-form text, but an ordered list of semi-independent timeline entries.

The question of how to evaluate single-document summarisers has been discussed controversially and remains an unsolved problem to this day. In this chapter, I first describe different types of summaries and their properties (Section 4.1), and then review existing literature on summarisation evaluation (Section 4.2). Methods that rely on a human gold standard and their limitations are discussed in greater detail, as this type of method is most appropriate to my task. In Section 4.3, I describe how existing approaches to summarisation evaluation deal with the inherent subjectivity of human annotation. Section 4.4 discusses the requirements of an evaluation methodology for the task of timeline generation. I will introduce such a methodology in Chapter 5.

4.1 Types of summaries

Different types of summaries exist. An important distinction is often made between *extractive* and *abstractive* summaries (Rowley, 1982; Cremmins, 1996). An extractive summary is created by concatenating text fragments contained in the source article; no generation of new text is required. Extractive summaries typically lack coherence, as the text fragments that form the summary are taken from different parts of the source document. Common problems are dangling anaphors and gaps in the summary's rhetorical structure (Mani, 2001). For instance, it is possible that a summary sentence contains a pronoun which refers to a noun phrase missing from the summary.

An abstractive summary, on the other hand, contains sentences created specifically for the summary. For instance, a mobile phone test report that is used as the input text

to a summarisation algorithm may describe in detail the many advantages of a phone's camera. An abstractive summary for this text might contain a new sentence such as "The quality of the phone's camera is outstanding".

Two strategies for performing abstractive summarisation have been discussed in the literature. The first option is to parse the source text and perform operations such as sentence compression, sentence fusion and paraphrasing. Such methods manipulate existing text, but do not *generate* completely new text based on an underlying generation plan (Hahn and Mani, 2000). In contrast, "deeper" approaches generate text which verbalises the result of inference steps made based on the source text, or transform the text into an abstract meaning representation, from which the summary sentences are then generated. For instance, information extraction techniques can be used to fill domain-specific templates; subsequently, natural language generation algorithms generate new sentences from these templates (Radev and McKeown, 1998; White et al., 2001; Genest and Lapalme, 2012).

Summaries can also be classified according to whether they are *informative* or *indicative*. Informative summaries are shorter substitutes of the source text (Hahn and Mani, 2000). Accordingly, informative summaries belong to the same text genre as the original: An informative summary of a report is again a report. A summary of this kind allows the reader to process the most salient information contained in the source text more quickly. The objective of indicative summaries, on the other hand, is to provide a list of the most important information sources on a topic (Hahn and Mani, 2000). These summaries thus belong to a different text genre from the original text, and fulfil a similar function to that of tables of content or even indices. The reader will be required to follow up the pointers given in the summary in order to access the information contained in the original texts.

A futher distinction can be made between *generic* and *user-* or *query-focused* summaries (Hahn and Mani, 2000). Generic summaries are constructed without a particular use case in mind. They are not tailored to the needs of a group of users or a specific subject domain. User- and query-focused summaries on the other hand are constructed with respect to a given information need. For instance, a user-focused summary of a mobile phone test report should only describe the mobile phone's camera provided that the user is not interested in the other components of the phone. A different example of a query-focused summary is a summary written in simple language.

4.2 Overview of summarisation evaluation methods

Two major groups of evaluation methodologies exist (Jones and Galliers, 1996; Jing et al., 1998; Mani, 2001). *Intrinsic* approaches judge the quality of a summary in its own right, while *extrinsic* approaches measure the quality of a summary by evaluating its usefulness for a downstream task.

4.2.1 Extrinsic evaluation methods

Although the evaluation methodology for timeline generation that will be presented in Chapter 5 is intrinsic, I will start by giving an overview of extrinsic evaluation methods. Extrinsic approaches can be subdivided into two groups. Methods in the first group directly measure the usefulness of a system summary for a given downstream task. For example, one may a set up an experiment where a human uses the information contained in a system summary to answer questions on the corresponding, unseen source text (Morris

et al., 1992; Hahn and Mani, 2000). A good summary is one that allows the human to answer a high number of questions about the original text. Further examples of such tasks are given by Mani et al. (1999). They describe a relevance decision task in which humans use the information in the summary to judge the relevance of the source text with regard to a topic. A further task assesses the ability of humans to categorise the unseen source text into one of ten topics (such as "Bioconversion" or "Worldwide tax sources") based on the summary.

A second group of extrinsic approaches measure the human effort needed to change a system summary in a such a way that it becomes useful for a downstream task. The evaluation by Hovy and Marcu (1998) uses a variant of the Shannon Game, which is inspired by Shannon's measures well-known from information theory (Shannon, 1948).[1] Humans were asked to recreate a paragraph in the source text either based on the summary, based on the full source text, or from scratch. The number of keystrokes needed to perform this task in each scenario was then measured. A low number of keystrokes indicates that the information retained by the summary is high.

The advantage of extrinsic approaches is that summaries receive a similar score if they are equally useful for the downstream task, even if they are very different from each other. However, extrinsic evaluation methodologies have limitations, too: The quality of a summary is measured only with regard to a single downstream task. A summary may be useful for this particular downstream task, but much less useful for other tasks. Further, extrinsic evaluations are often time-consuming to set up (van Halteren and Teufel, 2003).

4.2.2 Intrinsic evaluation methods

I now turn to describing intrinsic evaluation methods. Such methods can assess two properties of a summary: the summary's content and aspects of text quality. Content evaluation measures to what extent the summary covers the most salient content in the source, whereas text quality evaluation assesses aspects such as readability, grammar and coherence (Mani, 2001; Steinberger and Jezek, 2009). Intrinsic approaches either evaluate the quality of a summary directly using subjective scores elicited from humans, or by comparing the summary to a human gold standard. In some cases, the use of a gold standard rests on the assumption that an *ideal summary* can be directly elicited from a single human, which can be problematic.

4.2.2.1 Human judgments of summary quality

One way of performing intrinsic evaluation is to elicit human judgments for the summary. For instance, humans may be asked to assess certain aspects of a summary's quality on a numeric scale, e.g. from 0 to 5. The advantage of such methods is that the analysis can cover quality criteria which are particularly hard to evaluate automatically, such as coherence and readability (Mani, 2001; Lin, 2004). Human judgments can be collected in a number of settings: The human annotators can be given access to the source; to a reference summary; or to a sample summary and the score assigned to it (Mani, 2001). Each of these sources of information helps the annotators judge the quality of the summary.

A well-known evaluation scheme based on human judgments was proposed in the context of the Document Understanding Conferences (DUC). In this evaluation effort, human annotators evaluated both coverage and text quality. In order to assess coverage

[1] A good explanation of Shannon games can be found in Hassel (2004).

for a sentence in the gold-standard summary, the annotator first identified all sentences in the system summary that expressed at least some of the content of the gold-standard summary sentence. Next, the degree to which that content is expressed needs to be estimated on a discrete scale of four options (*all*, *most*, *some* and *hardly any*). Text quality was evaluated independently of the reference summary. Assessors rated grammaticality, cohesion and coherence on a discrete scale (*all*, *most*, *some*, *hardly any*, and *none*) similar to one used for the evaluation of coverage (Lin and Hovy, 2002).

4.2.2.2 Shallow gold-standard comparison

Other methods allow for an automatic comparison of reference and system summary. This has the advantage that new system summaries may be scored without the need for additional human judgments. Evaluation is typically restricted to assessing the informativeness of the summary, as the analysis of other aspects is hard to automate.

In such methods, the first step is to elicit a human-written summary. Next, similarity between the system summary and the human summary is calculated using a similarity metric. Nowadays, humans are asked to write their own summaries from scratch, and surface-oriented overlap metrics are used. In earlier work, the human-created summary was only a sentence-based extract from the source text, and the distance metric between system summary and human summary was simply the number of sentences co-selected by the two summaries (Rath et al., 1961). The modern equivalent, ROUGE, uses a range of overlap metrics based on n-grams and longest common subsequences (Lin, 2004).

In ROUGE, the surface texts of a system and a gold-standard summary are compared on the level of tokens, using a family of metrics based on n-grams and longest common subsequences of tokens. Such a comparison is based on the implicit assumption that dissimilar wording between two statements always entails that the meaning is dissimilar also. Clearly, this assumption does not hold in practice. Two sentences such as "Prince William got married to his girlfriend" and "William and Kate tied the knot", which are very different in wording, express exactly the same event.

The only human input required for ROUGE is a set of gold-standard summaries written by humans. Creating these summaries is a one-time effort, which does not have to be repeated as new system summaries are evaluated. Further, no annotation is required on the system summaries themselves. This is an advantage, as new algorithms can be tested without incurring any further annotation effort.

I will now give an overview of the different ROUGE metrics. ROUGE-N considers n-grams of varying length, where N is the length of the n-grams; hence bigrams are used in ROUGE-2, trigrams in ROUGE-3, and so on. The score of the system summary is then calculated as

$$\text{ROUGE-N} = \frac{\sum_{S \in \{\text{ReferenceSummaries}\}} \sum_{\text{gram}_n \in S} \text{Count}_{match}(\text{gram}_n)}{\sum_{S \in \{\text{ReferenceSummaries}\}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)} \tag{4.1}$$

where S is a sentence occurring in one of the reference summaries, $gram_n$ is an n-gram in a sentence S, $Count(gram_n)$ is the number of occurrences of $gram_n$ in that sentence, and $Count_{match}(gram_n)$ is the number of such occurrences that also occur in the system summary.

This variant of ROUGE builds on BLEU, a similar n-gram-based metric used for evaluating machine translation systems (Papineni et al., 2002). BLEU measures how many n-grams in a candidate translation occur in a reference translation; it is hence

precision-oriented. Conversely, ROUGE is recall-oriented: It measures how much content from the human gold-standard summaries occurs in a system-generated summary. With this metric, it is therefore crucial for fairness to only ever compare system summaries of exactly the same length.

ROUGE-L is another variant of ROUGE. It uses the longest common subsequence (LCS) of reference summary sentences and system summary sentences in lieu of n-grams, i.e. it does not differentiate between subsequences of consecutive words and subsequences with gaps in them. ROUGE-W, a variant of ROUGE-L, assigns a higher weight if there are consecutive matches between words in the two sentences.

I will now explain the difference between ROUGE-N and ROUGE-L. ROUGE-N compares n-grams of a given, fixed size, but does not give a specific reward if there are longer common subsequences of words. For instance, if a gold-standard summary and the system summary share an entire sentence that is ten words long, ROUGE-2 only recognises that there are 9 bigrams that occur in both sentences, but does not take into account that the largest common n-gram is in fact a 10-gram. On the other hand, LCS-based methods only take into account a single (longest) subsequence of words in a pair of sentences (Lin, 2004), while other subsequences are not considered. For example, if a sentence in a gold-standard summary reads as "The election of Trump as President of the United States was a surprise to many political analysts", and the system summary contains the sentence "Many political analysts were surprised by the election of Trump as President of the United States", LCS-based methods would only recognise that there is a common longest subsequence of length 10 ("the election of Trump as President of the United States"), but would not take into account other common subsequences (here: "many political analysts"). In contrast, ROUGE-N would detect the presence of common n-grams in all parts of the sentence.

Two further ROUGE variants (ROUGE-S and ROUGE-SU) consider skip-bigrams, i.e. in-order pairs of words in a sentence. In "Sally is young", there are three skip-bigrams: (Sally, is), (Sally, young) and (is, young). ROUGE-SU is a special version of ROUGE-S which considers unigram overlap in addition to skip bigrams. ROUGE-S and ROUGE-SU are similar to ROUGE-L in that non-consecutive word pairs are taken into account. However, a fixed n-gram size is used as in ROUGE-N.

Although some variants of ROUGE, such as ROUGE-S, cover simple modifications of the source text (such as the insertion or deletion of a word), surface-oriented methods are unable to assign points if the system summary uses a wording to express a given fact that this completely different from the wording used in the gold-standard summaries. This limitation does not apply to a different class of evaluation methodologies, which I will describe next.

4.2.3 Deep evaluation methodologies

An alternative way of scoring a summary using a gold standard starts by identifying *meaning units* in summaries. A system summary is rewarded if it expresses a high number of meaning units contained in the human gold-standard summary. Such methods are able to abstract away from different surface representations of the same content, as they award points even if the system summary expresses a meaning unit contained in the gold-standard summary using different words.

In practice, however, the annotation cost of such methods is often prohibitive, as each system summary has to be annotated from scratch. A further problem is that the definition of meaning units in such methods is often vague. Existing approaches

represent the semantics of a sentence using elementary discourse units (Carlson et al., 2001; Harman, 2002), first-order-predicate logic-style semantics (van Halteren and Teufel, 2003), or vaguely defined "Semantic Content Units" (SCUs) (Nenkova et al., 2007).

One of the most widely used deep evaluation methodologies is the pyramid method (Nenkova and Passonneau, 2004; Nenkova et al., 2007). Here, semantic content units (SCUs) have to be created manually for both system and gold-standard summaries.

SCUs are constructed empirically. Initially, one SCU is created for each sentence in the gold-standard or system summary. An SCU is then split if any two summaries express different parts of the meaning of an existing SCU. Using this approach, a system summary is guaranteed to be penalised for omitting content mentioned in a gold-standard summary.

For instance, let us assume that the system summary contains the following sentence: "A terror attack in the US killed 10 people." If only one of the gold-standard summaries contained content from this sentence, a single SCU with this meaning would be created. Now consider the case where there are three gold-standard summaries, each containing some of the content mentioned in the above system summary sentence:

- *Sentence in gold-standard summary 1:* There was a terror attack in the US.
- *Sentence in gold-standard summary 2:* A terror attack in the US killed 10.
- *Sentence in gold-standard summary 3:* A terror attack killed 10.

In this case, three SCUs are created to account for the fact that the gold-standard summaries mention different parts of the system summary sentence.

- *SCU 1:* There was a terror attack.
- *SCU 2:* The terror attack was in the US.
- *SCU 3:* 10 people were killed.

This approach has the undesirable side effect that the relative importance of content for evaluation increases if it is represented by multiple content units. For instance, the single system summary sentence above is represented by three content units. For many other sentences, only a single content unit would be created.

The SCUs constructed from the gold-standard summaries are then grouped into *tiers*, based on the number of gold-standard summaries expressing them. An SCU in tier n is expressed in n gold-standard summaries, and is said to have SCU weight n. A tier n usually contains fewer SCUs than tier $n-1$, as it is unlikely that a high number of annotators express the same content. This structure can be visualised in the form of a *pyramid* (see Figure 4.1). In this visualisation, each SCU is represented by a black dot. The layer at the bottom represents tier 1, and layers further up in the pyramid correspond to tiers with higher n. The width of each layer relates to the expected number of SCUs in each tier, with layers for lower tiers being wider, as they normally contain a higher number of SCUs. Nenkova and Passonneau assume that a pyramid shape will naturally evolve for summaries, and this is generally empirically the case.

Once SCUs have been annotated for a new system summary, evaluation can be carried out automatically using the following procedure: It is assumed that an ideal summary can be formed by adding the most highly weighted SCUs to the summary, until the target summary size $nSCU$ has been reached. The target summary size is measured in terms

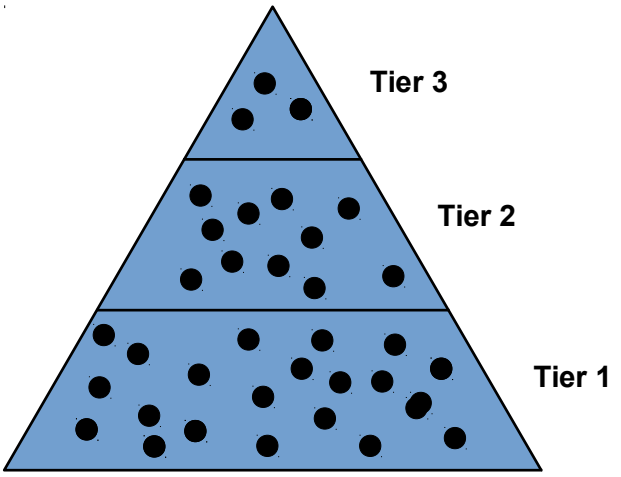


Figure 4.1: Pyramid with SCUs (represented as black dots).

of the number of SCUs expressed in the text. The maximum score that a system can achieve for a given input document d and a given target summary size $nSCU$ is the sum of the weights of the most highly weighted SCUs up to the size limit:

$$maxscore(d, nSCU) = \sum_{i=j+1}^{nt} i \cdot |T_i| + j \cdot (nSCU - \sum_{j+1}^{nt} |T_i|) \tag{4.2}$$

with

$$j = \max_i (\sum_{t=i}^{nt} |T_t| \geq nSCU) \tag{4.3}$$

where T_i is the set of all SCUs with weight i; j is the index of the next-lower pyramid tier; and nt is the index of the lowest pyramid tier all of whose SCUs can be included in the summary.

For instance, in a scenario where there are 4 SCUs of weight 3, 10 SCUs of weight 2 and 23 SCUs of weight 1, and where the target summary size is 15 SCUs, *maxscore* is calculated as $4 \cdot 3 + 10 \cdot 2 + 1 \cdot 1 = 12 + 20 + 1 = 33$. In practice, it may prove difficult to choose the parameter $nSCU$ appropriately if the goal is to ensure that the system summary does not exceed a certain number of words or sentences.

The score given to the system summary is then calculated as follows:

$$score(d, nSCU) = \frac{\sum_{s \in S_d} w(s)}{maxscore(d, nSCU)} \tag{4.4}$$

where S_d is the set of all SCUs represented in the system summary, and $w(s)$ is the SCU weight of SCU s. A score of 1 means that the summary is optimal in that it contains the $nSCU$ most highly weighted SCUs in the pyramid.

Since SCUs have to be created manually for each system summary, followed by a manual annotation of links to an existing pyramid, the pyramid method is too expensive for many real-world evaluations, in particular for day-to-day algorithm development. Harnly et al. (2005) and Passonneau et al. (2013) attempt to automate these expensive steps. The method of Harnly et al. (2005) finds the set of contiguous text spans in each system-summary sentence that produces the highest overlap with the SCUs in an existing pyramid. These text spans are then used as the system summary's SCUs.

The algorithm starts by listing all possible contiguous spans of words for each sentence in the system summary (called *candidate contributors*). For instance, for the sentence "Canada was founded" the candidate contributors "Canada", "Canada was", "was founded" and "Canada was founded" are created. Next, for each candidate contributor and SCU in the pyramid, a score $set\_sim(\hat{c}, C)$ is calculated:

$$set\_sim(\hat{c}, C) = combine_{c_i \in C}(span\_sim(\hat{c}, c_i)) \tag{4.5}$$

where $\hat{c}$ is the candidate contributor to be scored, C is an SCU in the pyramid, c_i is a contributor of that SCU, *span_sim* is a function calculating a similarity score between a candidate contributor and a contributor in the SCU, and *combine* is a function which combines the different scores into a single score for the candidate contributor. For each possible partitioning, the sum of all scores is calculated. The highest-scoring set is identified using a dynamic programming algorithm.

A number of alternatives for the functions *combine* and *span_sim* are explored. The best-performing method uses a *span_sim* function which counts the number of overlapping unigrams between the candidate contributor and the contributor from the SCU. The best-performing method for *combine* uses the minimum of all scores obtained, i.e. the candidate contributor must achieve a high score with *all* existing contributors. Functions that consider the maximum or the average of all scores result in lower performance.

According to Harnly et al. (2005) themselves, the fact that the mapping is based on unigram overlap is a disadvantage of their algorithm, as it is impossible to map a candidate contributor to an SCU if one or more human summarisers have used a paraphrase to describe the same content. This problem is exacerbated by the *combine* function which performed best. This method produces a low total score even if only a single gold-standard summary uses a paraphrase instead of the identical wording. In a way, this limitation defeats the purpose of a "deep" evaluation methodology, which is supposed to capture cases where the same idea is expressed using different words. Unigram overlap is not ideal also because it ignores word order.

Passonneau et al. (2013) attempt to address some of these limitations by using a 100-dimensional latent vector representation instead of unigram overlap to compare candidate contributors to existing contributors in SCUs. They also perform an additional evaluation in order to analyse the algorithm's performance on assigning a candidate contributor to the correct existing SCU chosen by human annotators. Even when a latent vector representation is used, the correct SCU can only be identified in 50% of all cases. Without a qualitative analysis, it is not possible to judge whether the algorithm can indeed deal with abstraction and reformulation.

A second limitation of the pyramid method is that SCUs are tied to a given, fixed set of summaries due to the way in which the SCUs are created. If further summaries are added at a later stage, existing content units may have to be separated and re-defined (van Halteren and Teufel, 2003).

4.3 Subjectivity of human content selection tasks

All evaluation methods presented are potentially prone to the problem of subjectivity of human content selection. In particular, methods that presume the existence of a gold standard assume that there is an "ideal" summary that all other summaries can be compared against. Intrinsic evaluations that do not use a gold standard, for instance methods that

rely on human judgment of summary quality, are subject to human subjectivity as well. Whenever a new evaluation method is proposed, it is therefore necessary to demonstrate that any human input that the method depends on is replicable.

Subjectivity in human annotation can be reduced by using *guidelines* which are given to the human participants (Carletta, 1996). Guidelines communicate the experimenter's interpretation of the annotation to the annotators. With a good set of guidelines, different annotators will have a similar interpretation of the task and therefore produce more similar annotations. An annotation study that involves guidelines typically consists of four stages: the development of suitable guidelines, the training of annotators who did not participate in constructing the guidelines, the collection of the final judgments, and a formal reliability study (Teufel, 2010).

4.3.1 Quantifying subjectivity

I will now first discuss common methods for measuring subjectivity, before turning to subjectivity in summarisation tasks in particular.

If it is possible to interpret the data collection from humans as an annotation task, the methodology of content analysis (Krippendorff, 1980) can be used to measure the amount of subjectivity present. This methodology assumes that an *annotation scheme* or *coding scheme* has been created. An annotation scheme is a set of *categories* or *classes* that annotators can choose from, together with descriptions of the semantics of each category.

The usual way to establish trust in such an annotation scheme is to conduct a reliability study. Krippendorff distinguishes three different properties that an annotation scheme must fulfill: *Stability* (or *intra-annotator agreement*) is the requirement that one and the same annotator should produce the same judgments when asked to complete the annotation of the same data again at a later point in time. *Reliability* (also *reproducibility* or *inter-annotator agreement*) quantifies the similarity of judgments produced by different annotators on the same data. Finally, *accuracy* or *validity* measures how well an annotator's judgments correspond to a known gold standard, if such a ground truth exists.

For categorial annotation tasks, standard agreement metrics exist. *Summary metrics* are single numbers that allow for quick assessment of agreement (Carletta, 1996). The most well-known such metric is Fleiss' κ (Fleiss, 1971), which takes into account chance agreement between annotators, as opposed to a simple percentage agreement; and Krippendorff's α, which additionally considers degrees of similarity between categories. Fleiss' κ is given as

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{4.6}$$

where $\bar{P} - \bar{P}_e$ is the degree of agreement actually attained in excess of chance, and $1 - \bar{P}_e$ is the degree of agreement attainable above what would be predicted by chance (Fleiss, 1971). Krippendorff's α is calculated as follows:

$$\alpha = 1 - \frac{D_o}{D_e} \tag{4.7}$$

where D_o refers to the observed disagreement among values chosen for any single data point, and D_e is the disagreement that one would expect if annotation were performed

randomly (taking into account the observed category distributions, as described above for Fleiss' κ). The observed disagreement D_o is calculated as

$$D_o = \frac{1}{n} \cdot \sum_c \sum_k o_{ck} \delta^2_{ck} \tag{4.8}$$

where o_{ok} is the number of data points for which two annotators differed in their annotation and chose classes o and k, respectively; n is the total number of pairs of annotations; and δ^2_{ck} is the (manually specified) importance of a misclassification involving classes c and k. For instance, in a setting where only two classes (a and b) and two annotators are used, the value of o_{ab} would be 3 if there are three data points for which one annotator has chosen class a and the other annotator has chosen class b.[2] Note that $o_{ck} = o_{kc}$, i.e. each frequency of a type of misclassifications is used twice for calculating α.

The additional difference function δ^2_{ck} in Equation 4.8, which can be given either in closed form or as an enumeration of all function values, allows for treating different types of misclassifications differently. For instance, where the class labels are numbers on an interval scale, a misclassification between classes 1 and 3 should result in a higher observed disagreement than one between classes 2 and 3. This can be achieved, for instance, by using a difference function that grows quadratically as the difference between the two labels increases (Krippendorff, 2007):

$$_{interval}\delta^2_{ck} = (c - k)^2 \tag{4.9}$$

The expected disagreement assuming a random assignment of class labels is calculated accordingly:

$$D_e = \frac{1}{n(n-1)} \cdot \sum_c \sum_k n_c n_k \delta^2_{ck} \tag{4.10}$$

where n_c refers to the total frequency of misclassifications between class c and any other class, and n_k is calculated analogously.

4.3.2 Subjectivity in summarisation tasks

Having discussed how subjectivity in human annotation can be quantified in principle, I now turn to subjectivity in summarisation tasks.

In general, human agreement on content selection for summarisation is known to be low (Rath et al., 1961; Salton et al., 1997; Marcu, Daniel, 1997; Lin and Hovy, 2002). This was demonstrated first in early work by Rath et al. (1961), who found that both intra- and inter-annotator agreement for their sentence selection task was poor, although better than with a random selection. They conclude that there is no single set of representative sentences that all humans can be expected to agree on, but many equally representative sets of sentences for a given article. Similarly, for the DUC evaluation scheme, it has been shown on the basis of percentage agreement that both intra- and inter-annotator agreement are low (Lin and Hovy, 2002).

One should therefore avoid constructing a gold standard for summarisation on the basis of a single human solution only, as the scores assigned to a system summary using

[2] An explanation for cases where higher number of annotators and/or classes are used can be found in Krippendorff (2007).

such a gold standard may vary considerably depending on which human was used. As a consequence, most recent evaluation methods aim to reduce the impact of subjectivity by using multiple human solutions. For instance, the ROUGE family of surface-based overlap metrics are commonly calculated using multiple reference summaries (cf. Section 4.2.2.2 on page 72). In particular, those ROUGE metrics that operate on n-grams score the system summary based on the combination of all n-grams found in any of the system summaries. As each gold-standard summary is expected to contain n-grams that do not occur in any of the other gold-standard summaries, the system summary will be scored based on a higher overall number of gold-standard n-grams compared to the case where only a single gold-standard summary is used. This property somewhat increases the probability that a system summary is adequately rewarded even if some gold-standard summaries use a different wording to express a fact mentioned in the system summary.

Since ROUGE is only surface-based, it will miss many true similarities that are expressed by different wordings. Lin and Hovy therefore test to what extent ROUGE scores correlate to human judgments, in particular to the judgments elicited in the DUC evaluation effort. However, it is questionable whether correlation to human judgments can demonstrate the reliability of an evaluation method if these human judgments cannot themselves be reliably elicited, as was demonstrated by the low intra- and inter-annotator agreement for the DUC annotation scheme found by Lin and Hovy (2002).

Multiple gold-standard summaries have also been used with deep evaluation methods. The pyramid method by Nenkova and Passonneau (2004) addresses human subjectivity in content selection by assigning higher weight to content units expressed by multiple summary writers.

Nenkova et al. (2007) investigate the human subjectivity present in their method in various ways. One approach is to evaluate a pair of held-out summaries[3] using pyramids of different sizes. Even if the scores of the two summaries differ considerably when evaluated using a single large pyramid (e.g. of size 9), some too small pyramids (e.g. of size 2) can result in the worse summary receiving a higher score (Nenkova et al., 2007), which is undesirable. Choosing the right pyramid size means finding the right balance between annotation effort and score reliability (Nenkova and Passonneau, 2004). In addition, Nenkova and Passoneau analyse the subjectivity present in other steps of their evaluation methodology. This includes the creation of SCUs from human summaries and system summaries respectively, and the matching of SCUs to each other.

4.4 Requirements of timeline generation evaluation

Each of the evaluation methods presented has important limitations. Shallow evaluation methodologies assume that a difference in wording always represents a difference in meaning, which is not true. Existing semantically-oriented evaluation methodologies, on the other hand, often require prohibitive annotation effort.

An evaluation methodology for single-document summarisation algorithms which does not suffer from these problems must fulfill the following requirements:

1. It should be “deep”, i.e. based on abstract semantic meaning units, like the pyramid method. However, it should not incur the quality problems observed in a surface-oriented automation of SCU candidate identification, such as in Harnly et al. (2005) and Passonneau et al. (2013).

[3] A similar analysis is performed with more than two held-out summaries, using correlation coefficients.

2. It should require as little annotation effort as possible. In particular, the annotation effort should be a one-time effort that is performed before system summaries are created and that is independent of the number of system summaries considered (as is the case for ROUGE). This has the advantage that an infinite supply of new system summaries (e.g. those created in day-to-day development) may be evaluated at no further cost. However, the one-time effort needed to construct the evaluation resource is still substantial, in particular with larger numbers of articles. The evaluation methodology must therefore provide detailed guidelines on how timeline creation should be performed, such that the risk of having to repeat the data collection is minimised.

In the next chapter, I will describe an evaluation methodology for the special summarisation task of timeline generation which fulfills these requirements. I will follow the intuitions of Nenkova and Passonneau (2004) and Teufel (2010) in that semantic units will be created from multiple timelines. Similar to Nenkova and Passoneau, I do not use subjective judgments of summary quality, as these scores can often not be reliably elicited (Lin and Hovy, 2002). While the specifics of my evaluation methodology are tailored towards the task of creating a history timeline, it can be applied, with suitable modifications, to single-document summarisation tasks more generally.

Chapter 5

Evaluation of timelines using semantic units

In the previous chapter, I discussed the merits and limitations of different methods for summarisation evaluation. Methods that use semantic units are robust in cases where the same content is expressed differently, but the need for manually matching these units makes such methods expensive, as this has to be done for each system summary in turn. On the other hand, surface-oriented methods such as ROUGE, which require less annotation effort, assume that a difference in wording always corresponds to a difference in meaning, which is not true. For this reason, such methods typically fail to detect the presence of important content in a system summary if that content is presented, for instance, in a more abstract form than in the source text.

In this chapter, I will describe an evaluation methodology for timeline generation which suffers from neither of these limitations.[1] It is similar to the pyramid method of Nenkova et al. in that semantic units are used. In particular, I will use a unit that covers exactly one event; this unit will be called "Historical Content Unit" (HCU). However, in contrast to the pyramid method, the methodology can be used for automatic evaluation of system timelines without the need for manual annotation of each timeline.

The chapter is structured as follows. In Section 5.1, I will describe the main principles of my evaluation methodology. In Section 5.2, the various steps needed to evaluate timelines using the methodology will be presented. Section 5.3 introduces an evaluation resource constructed using my methodology. In Section 5.4, I will describe experiments carried out in order to verify that central concepts of the methodology are meaningful. In particular, I will analyse whether the evaluation resource created is reproducible. I will also present an additional development resource which was created in a simplified manner (Section 5.5).

5.1 Principles of the evaluation methodology

In this section, I will introduce central concepts of my evaluation methodology.

[1] The work presented in this chapter has been published previously (Bauer and Teufel, 2015). The evaluation resource presented in Section 5.3 can be downloaded from `http://www.cl.cam.ac.uk/~smb89/form.html`.

5.1.1 Event definition

Let us first turn to the definition of an event. The event definition I adopt strikes a balance between the event model of TimeML and that of the Automatic Content Extraction effort, which were both introduced in Section 2.2. I define an event as follows:

1. An event is equivalent to an action (such as "France invades Algeria") or a state change ("Cameron becomes prime minister") in the real world. Approaches such as TimeML (described in Chapter 2) do not guarantee this equivalence. Examples of descriptions that are not events under my definition include cognitive acts, such as accounts of what people like, think, doubt, understand or remember, and descriptions of states (verbs like "remain", "be" etc.). Similarly, verbs denoting wishes and aspirations (such as "want" or "desire") do not constitute events.

2. The action or state change must hold true beyond doubt according to the text. Negated verbs, hypothetical statements or other constructions whose truth value strongly depends on someone's viewpoint are excluded.

3. It must be possible to anchor the action or state change in time and space. This criterion excludes actions and state changes that are independent of time (e.g. "volcanic activity occurs along these earth plate boundaries"), location (e.g. "the new year began"), or both (e.g. "trees use sunlight to convert water and carbon dioxide into sugar"). However, acceptable actions include those that are repeated over a limited period of time (e.g. "the king invaded various countries between 1413 and 1426") or in a limited number of locations (e.g. "the new faith spread to many surrounding cities").

An event in my approach is an abstract semantic unit, similar to ACE events. But as opposed to ACE, I do not use a finite list of fixed event types. All real-world actions and state changes that meet the above criteria can be annotated, independently of whether they fall in a pre-specified event schema or not. Similar to TimeML, I assume that the real-world action or state change expressed by an event can be represented by a single verb, nominalisation or event-like noun in the text.

5.1.2 HCUs

I now introduce the concept of a *Historical Content Unit* (HCU). An HCU is a content unit which expresses an event. This is in contrast to Semantic Content Units (SCUs) in the pyramid method of Nenkova et al. (2007), which often represent details mentioned in the surface text that do not themselves constitute an event, such as the number of casualties of a terror attack (as described in Section 4.2.3, cf. page 73).

HCUs are defined based on overlap between multiple human gold-standard timelines for a given source text, similar to SCUs. HCUs are also like SCUs in that they are associated with a pyramid weight (called *HCU weight*) and a manually created description. The HCU weight specifies how many timeline writers expressed an HCU in their timeline.

In addition, HCUs contain manually created links back into the source text; in particular, to TimeML events identified automatically in the text. A link is created whenever a TimeML event fully or partly expresses an HCU's semantic content. For each link, an *anchor weight* is additionally annotated. The anchor weight quantifies to what extent the TimeML event represents the HCU's semantic content.

HCU 16	
Action	Fatos Nano is elected Prime Minister
Agent	*not given*
Patient	Fatos Nano
Time	June 1997
Location	Albania

TimeML events	e18(1.0); e20(0.5)

Timeline entries	Fatos Nano was elected Prime Minister.	Timeline writer F
	Fatos Nano became Prime Minister.	Timeline writer C

Figure 5.1: Example of an annotated HCU.

Each HCU can be represented by a description similar to the one shown in Figure 5.1. The Action field contains a textual description of the real-world action or state change represented by the HCU. The field "Agent" describes who caused the real-world action or state change, while "Patient" shows who was affected by it. "Time" and "Location" describe how the HCU is anchored in time and space. The figure also shows the identifiers of two TimeML events ("e18" and "e20") in the source text that express the content of this HCU, together with their respective anchor weights ("1.0" and "0.5"); and two gold-standard timeline entries expressing the HCU which were written by two different timeline writers.

5.1.3 Overview of the evaluation methodology

I will now give an overview of my evaluation methodology. Figure 5.2 depicts the process of creating an evaluation resource. In a first step (represented by the arrow between the top and left boxes in the figure), timelines representing a history article on a given topic are elicited from human timeline writers.

In a second step (represented by the arrow between the left and right boxes at the bottom of the figure), a human experimenter familiar with the evaluation methodology (here: myself) identifies HCUs. HCUs are identified only based on the human-written timelines; system summaries are not considered at this stage. For instance, HCU 3 in the figure represents one particular event which has been expressed in two human-written timeline entries. It therefore receives an HCU weight of 2.

In a third step (represented by the arrow between the right and top boxes in the figure), the links between HCUs and TimeML events (including anchor weights) are annotated by a human experimenter. For instance, HCU 1 in Figure 5.2 can be represented by an event description involving "annexed" and another event description involving "invaded".

Once these steps are complete, the resulting resource can be used to evaluate an unlimited number of system timelines, as long as the system timeline has been created by selecting a number of TimeML events from the source text. No human annotation on these new timelines has to be performed. At evaluation time, the score of a system timeline is calculated based on the HCUs (including the anchor weights) previously annotated. A system that recalls many HCUs with a high HCU weight for a given timeline length will score better than a system that does not. As this evaluation metric is recall-oriented, a system will be punished for selecting redundant information.

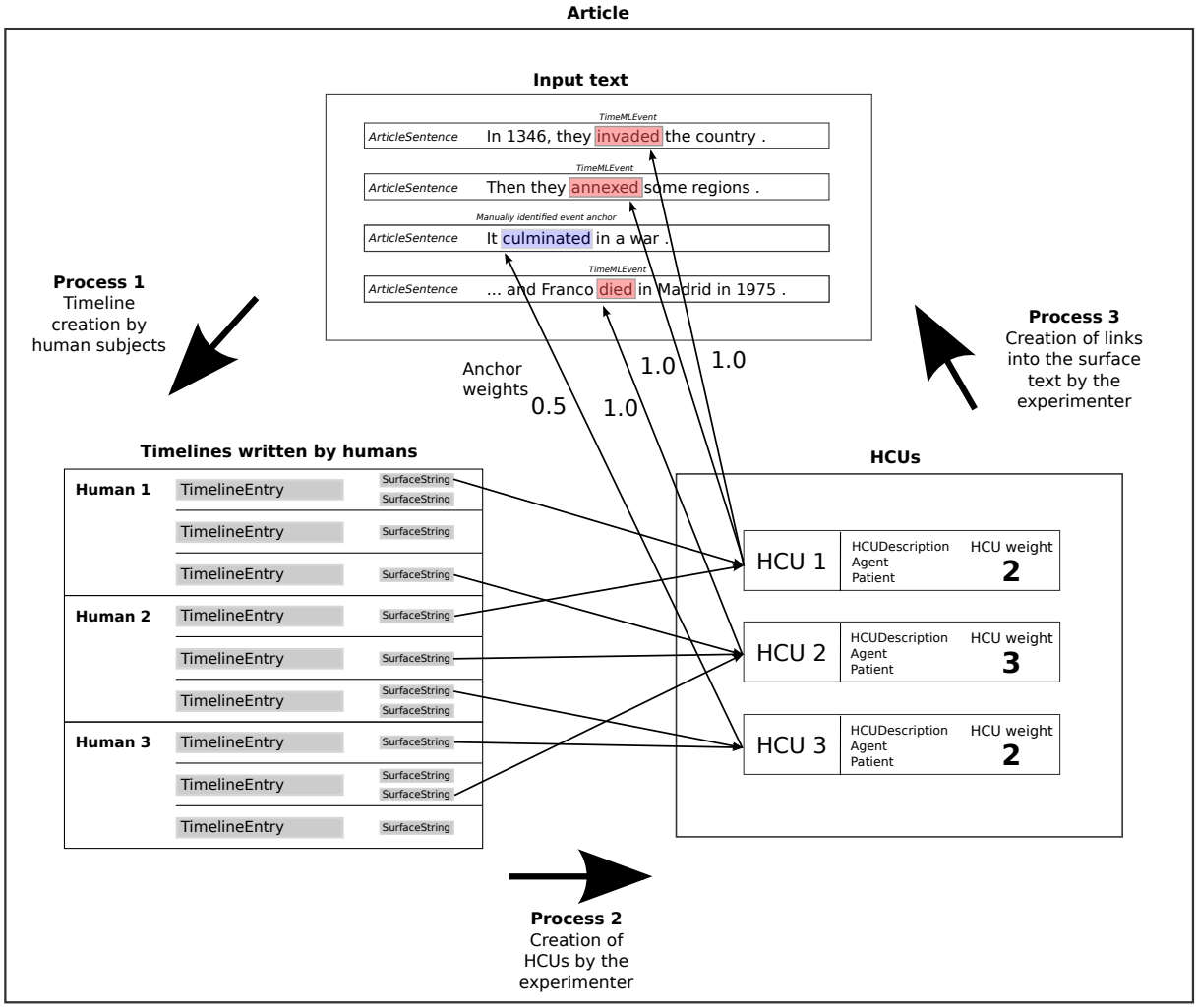


Figure 5.2: Overview of the evaluation methodology.

5.2 Design of the evaluation methodology

I will now describe the three central steps of my evaluation methodology in detail. Each step will be illustrated using examples drawn from the evaluation resource I created based on my methodology. This resource itself will be described in greater detail in Section 5.3.

5.2.1 Timeline elicitation

The first step in my methodology is to collect gold-standard timelines from human timeline writers. As described in the introduction, I define a timeline as a list of dated events or *timeline entries*. I assume that humans perform three main tasks when creating such a timeline for a given source article:

1. Real-world events are identified in the source text. Human timeline writers are free to decide what exactly constitutes an event in their mind. In particular, they may abstract away from the text by combining multiple individual actions described in the text into a single event. For instance, the annexation of several regions of a country within a longer time period may be summarised into a single event denoting the annexation of that country.

Your task is to create a **historical digest** for the following article describing the history of printing. A historical digest is a list of pairs of dates and important events, each on a single line:

date	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
date	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
date	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

You should create as many lines as you feel are appropriate, but no more than 25.
Each line should be understandable on its own, without context, and contain only one event.
If you feel that more than one important event is taking place at a particular point in time, then please create more than one line for that time.
In general, a good historical digest strikes a balance between mentioning all and only important events and still giving a complete account of the overall time period covered.
Ideally, all the events you select are important and it is possible to assign an exact date to each of them.
There are cases, however, such as time periods in which few events take place, or time periods that we know relatively little about (such as prehistory), where you may depart from these criteria. For example, you may choose an event that is less important in order to make sure a time period is covered; or you may use a vaguer date for an event. If there is an event that covers a longer time period, it is better to create a single entry with the entire time period as the date than to create individual entries for the beginning and end of the event.
If you happen to know additional facts about the topic of the text, please do not use them. When creating your historical digest, please also do not use any other sources (e.g. web pages, encyclopaedias).

Thank you very much for participating in this task. It will help develop algorithms that create historical digests automatically.

Figure 5.3: Guidelines for writing a timeline.

2. A subset of these events is selected for the timeline.

3. Each selected event must be expressed as a timeline entry, which consists of a date and a textual event description. Given that event descriptions are self-contained sentences or noun phrases, their surface form is often different from text in the source article. In particular, event descriptions should be independent of the document context, and also independent of each other.

Recall that a good timeline is expected to contain only salient events while covering the entire time period described by the article. These objectives are often conflicting: Ensuring coverage of the whole time period described sometimes necessitates the selection of less important events. A good timeline should therefore strike a balance between these conflicting objectives.

In order to elicit timelines that meet the aforementioned criteria, I use the guidelines shown in Figure 5.3. No further background knowledge, e.g. linguistic or subject matter expertise, was required to perform the task. The word "timeline" is not mentioned in the guidelines, as participants of the experiment should not feel reminded of any objects called timelines which they might have seen earlier, for instance graphical timelines. The term

"historical digest" was used instead. In line with the desirable properties of timelines, the instructions do not contain any statements about the linguistic realisation of timeline entries.

An important parameter in timeline elicitation is the length of the timeline created, i.e. the number of entries it contains (25 in the guidelines shown in Figure 5.3). Timelines that are too long are often not useful in practice, while short timelines may lack important content. It may also be difficult to achieve agreement between timeline writers if timelines are too short.

In general, the length ratio between a summary and the source text is referred to as the *compression factor*. I define the compression factor of a timeline as follows:

$$CF = \frac{\text{number of timeline entries}}{\text{number of verbs in the source text}} \tag{5.1}$$

The number of verbs is used since the task is to select events in the surface text, which are often represented by verbs.

Choosing the number of timeline entries as the length of the summary and the number of verbs as the length of the source text is in contrast to standard single-document summarisation tasks, where the length of both the summary and the source text is measured in words, as follows:

$$CF = \frac{\text{number of words in the summary}}{\text{number of words in the original text}} \tag{5.2}$$

I nonetheless use the number of timeline entries since I consider the number of words in a timeline entry to be irrelevant for a content selection task.

The aim during timeline elicitation should be to achieve the same compression factor for all timelines. For each article, I therefore set a different upper limit on the number of timeline entries, based on the number of verbs contained in the source article.

I will now describe how a sensible value for CF can be chosen. In a first step, I write timelines for randomly chosen articles myself, without a pre-specified restriction on the number of entries. I then calculate their CF values. The average of these empirically determined values for CF is used for all remaining articles.

For practical reasons, I will use the inverse of the compression factor (called *inverse compression factor* or ICF) in the remainder of this chapter. Using the procedure described above, ICF was set to 23.19. For other types of source texts (e.g. articles describing a smaller number of historical events in greater detail), a different value for ICF may be more appropriate.

The target numbers of events for the remaining articles were calculated using the following formula:

$$\text{target number of timeline entries} = \frac{\text{number of verbs in the article}}{ICF} \tag{5.3}$$

As I suspected that timeline writers in general have a tendency towards writing longer timelines, the resulting values were multiplied by 1.2 and then rounded to the next multiple of 5. Statistics on the target numbers of timeline entries chosen for the articles in my own evaluation resource will be given in Section 5.3.

Figures 5.4 and 5.5 show two timelines on the history of the jet engine elicited from human timeline writers. These examples demonstrate that timelines on the same topic can be very different in nature. Overall, the first timeline writer produced shorter timeline

Date	Timeline entry
150 BCE	The aeolipile, a device that uses steam power to cause a sphere to spin, was invented.
13th century	The Chinese invented the rocket, which has the idea of jet propulsion.
1633	As a stunt, Ottoman Lagari Hasan Çelebi took off with a cone-shaped rocket.
1913	René Lorin came up with a more efficient form of jet engine, the subsonic pulsejet.
1791	The patent for a stationary turbine was granted to John Barber in England.
1903	Norwegian engineer Ægidius Elling built the first gas turbine to run self-sustaining.
1915	In Hungary, Alberto Fonó increased the range of artillery by uniting a gun-launched projectile with a ramjet propulsion unit.
1921	Frenchman Maxime Guillaume filed the first patent for using a gas turbine to power an aircraft.
1923	Edgar Buckingham of the US National Bureau of Standards expressed scepticism about jet propulsion.
16/01/1930	Frank Whittle submitted his patent for a two-stage axial compressor feeding a single-sided centrifugal compressor.
1926	A.A. Griffith published a seminal paper that makes practical axial compressors possible.
April 1937	Frank Whittle had his first engine running on liquid fuel.
September 1937	In Germany, Hans von Ohain had their first engine running on hydrogen supplied under external pressure.
1938-1942	The first turboprop, the Jendrassik Cs-1, designed by György Jendrassik, was produced in the Ganz factory.
15 May 1941	A flyable version of Whittle's engine was fitted to an airframe to carry out the first flight.
1944	Mass production of Junno 004 engine started as a powerplant for the first jet-fighter, the Messerschmitt Me 262.
1941	The Metrovick F.2, the UK's first axial-flow engine, ran.
1950s	The jet engine became almost universal in combat aircraft, with the exception of cargo.
1960s	All large civilian aircraft has become jet powered.
1970s	The innovation of high bypass jet engines achieved higher fuel efficiency than the best piston and propeller engines.

Figure 5.4: Human-written timeline (first example) on the history of the jet engine.

entries. Certain entities are mentioned using longer referring expressions (e.g. "Norwegian engineer Ægidius Elling" and "Frenchman Maxime Guillaume"). However, relatively few subordinate clauses are used. In contrast, the second timeline writer produced very long timeline entries which are rich in subordinate clauses (e.g. "Whittle produces first jet engine *that runs and that is liquid-fueled and includes a self-contained fuel pump.*") and prepositional phrases. In many cases, it appears that a single timeline entry contains multiple event and state descriptions which are interconnected in complex ways (for instance in the entry "Erich Warsitz flies an He-178 fitted with Hans von Ohain and Max Hahn's HeS 3 gasoline-fueled engine marking the world's first turbo-jet-powered flight."). Another difference is that the first timeline starts in Antiquity and also covers an event in the Middle Ages, while the first event mentioned in the second timeline took place as late

Date	Timeline entry
1791	Patent for a stationary gas turbine granted to John Barber in England
1903	Ægidius Elling, a Norwegian engineer, builds first gas turbine to successfully run self-sustaining.
1903	Manufacturing limitations affecting safety and reliability inhibit the exploitation of the gas turbine in viable engines
1921	First patent for using a gas turbine to power an aircraft filed in 1921 by Frenchman Maxime Guillaume.
1926	The fact that axial compressors became practical is attributed to ideas published by A.A. Griffith.
1928	Frank Whittle formally submits his ideas for a turbojet to his superiors.
1928	Albert Fonó applies for a German patent on aircraft powered by supersonic ramjets.
1932	Whittle's Jet-engine patent showing a two-stage axial feeding a single-sided centrifugal compressor is granted.
1932	Albert Fonó is awarded a German patent on aircraft powered by supersonic ramjets.
1935	Hans von Ohain starts working on a design similar to Whittle's without knowledge of his work.
April, 1937	Whittle produces first jet engine that runs and that is liquid-fueled and includes a self-contained fuel pump.
September, 1937	Hans von Ohain and Max Hahn produce first HeS 1 centrifugal engine using hydrogen supplied externally as fuel.
August 27, 1939	Erich Warsitz flies an He-178 fitted with Hans von Ohain and Max Hahn's HeS 3 gasoline-fueled engine marking the world's first turbo-jet-powered flight.
15/5/41	First flight of Whittle's engine, fitted to a Gloster E28/39 airframe is carried out at RAF Cranwell.
1943	First flight based on Metrovick F.2, UK's first operational axial-flow engine
1944	Anselm Franz's axial-flow compressor-based turbine jet engine design goes into mass production as the power-plant of the Messerschmitt Me 262 fighter.
Following end of WW II	Centrifugal-flow engines continue to be improved and due to their small size and robustness, when compared to axial flow, they are used in helicopters.
Following end of WW II, 1950s and 1960s	Originating from British designs, US and Russian axial-flow engines find application on fighters such as the MiG-15 and F-86 Sabre.
1960s	Relentless improvements in the turboprop amount to the piston engine being replaced in mainstream aviation by jet engines.
1970s and beyond	The invention of the high bypass engine highlights the fuel efficiency of the jet engine, obtained at high altitudes, gradually transforming aviation into safe, economic and fast travel.

Figure 5.5: Human-written timeline (second example) on the history of the jet engine.

as in the 18th century. It seems that the first timeline writer placed greater importance on covering the overall time period described by the article, while the second writer was of the impression that many events in the 20th century were too important to be omitted in favour of events in earlier time periods.

5.2.2 Creation of HCUs

Once human-written timelines have been elicited, a human experimenter creates an evaluation resource consisting of HCUs, as described in Section 5.1. The process of creating HCUs from human-written timelines consists of two stages: identifying candidates for HCUs in *individual* human-written timelines, and merging these HCU candidates into HCUs. I will now describe these two processes in turn.

5.2.2.1 Identification of HCU candidates

For each HCU candidate, I fill in a template which consists of the five fields shown at the top of Figure 5.1 on page 83 (Action, Agent, Patient, Time, Location). In normal operation, one HCU candidate is created for each human-written timeline entry, since each entry is expected to express exactly one historical event. However, this heuristic does not work in all cases.

The first potential problem is that some timeline entries may describe something that is not an event by my definition. I therefore discard a timeline entry if it does not express an event according to the definition in Section 5.1.1.

The identification of HCU candidates is also complicated when a single timeline entry is composed of two entirely separate event descriptions which are written as two separate sentences or joined by a comma or semicolon. Here, I create a separate HCU candidate for each event. The date of the original timeline entry is used for all resulting HCU candidates.

Another problem arises where a timeline entry is syntactically complex, i.e. where it is not simply a noun phrase ("Crowning of King Peter") or a main clause ("King Peter was crowned"). In particular, the timeline entry may contain additional syntactic modifiers, for instance a subordinate clause or a prepositional phrase. Where such a modification is essential to the semantics of the event, I consider it to be part of the HCU candidate's semantics[2]. Conversely, if it is redundant, I disregard it.

One example of syntactic modification that I expect to appear frequently is relative clauses. Restrictive relative clauses (RRC) serve to delimit the potential referents of the head, whereas non-restrictive relative clauses (NRRC) merely give an additional piece of information about an already identified entity. NRRCs are usually set off from the main clause by commas (Comrie, 1981).

These two types of relative clauses are treated differently. RRCs are always added to the HCU description, while for NRRCs this must be decided on a case-by-case basis. Consider the following example timeline entries:

1. "Thousands of people were killed in the 9/11 terrorist attacks, which remain a national trauma to this day."

2. "Louis issued a new constitution which provided for a parliament composed of an elected Chamber of Deputies."

[2]These modifications will be treated as "background information" later on (cf. page 98).

Timeline entry	Kingdom of Hadramaut was conquered by Himyarite king Shammar Yuhar'ish, unifying all of the South Arabian Kingdoms.
HCU candidate	Kingdom of Hadramaut conquered by Himyarite king Shammar Yuhar'ish

Timeline entry	Kingdom of Ma'in ended, one of the first to end, and the Minaic language died
HCU candidate 1	Kingdom of Ma'in ended
HCU candidate 2	the Minaic language died

Timeline entry	Justinian I sent fleet to Yemen to prevent Dhu Nunas massacre, turning West Yemen into vassal state.
HCU candidate	Justinian I sent a fleet to Yemen to prevent Dhu Nunas massacre

Timeline entry	Yemen became a province in the Islamic Empire, being ruled as part of Arab-Islamic caliphates.
HCU candidate	Yemen became a province in the Islamic Empire

Timeline entry	Mutawakkilite kingdom of Yemen formed in the north after Turkish forces withdraw.
HCU candidate	Mutawakkilite kingdom formed in the north of Yemen

Timeline entry	Royalist forces, supported by Saudi Arabia and Jordan, opposed, starting the North Yemen Civil War
HCU candidate	Royalist forces opposed and thereby started the North Yemen Civil War

Timeline entry	Republic of Yemen (ROY) was declared with Saleh as president and al-Baidh as vice president.
HCU candidate	The Republic of Yemen was declared

Figure 5.6: Examples of timeline entries and corresponding HCU candidates (Action fields only) where the creation of HCU candidates was non-trivial.

In the first example, the information contained in the NRRC "which remains a national trauma to this day" is not essential to the main event's semantics (the killing of many people). In contrast, the RRC "which provided for a parliament composed of an elected Chamber of Deputies" in the second sentence arguably contains an essential part of the main event's semantics.

Under no circumstances did I create separate HCUs for events described in a syntactic modification of another event, given that the timeline writer did not perceive them as important enough to be expressed in a separate timeline entry.

Figure 5.6 shows a number of timeline entries for which one of the aforementioned problems occurred, as well as the Action fields of the corresponding HCU candidates.

5.2.2.2 Merging HCU candidates

Once HCU candidates have been identified in a set of human-written timelines for a document, HCUs have to be created based on these candidates. In particular, any group of HCU candidates that represent the same real-world event have to be merged into a single HCU.

Establishing whether two HCU candidates refer to the same event can be difficult, as the same event may be expressed in many different ways. I apply two basic principles to test this. The first is the principle of unification of non-conflicting information. This principle means that when two or more HCU candidates appear to refer to the same real-world event, I merge them into a single HCU if they do not contain conflicting information. To verify whether there is conflicting information, the values of each field (Action, Agent etc.) are compared. If there is a conflict in any of the fields, the HCU candidates cannot be merged.

The second principle is that of atomicity. It means that separate HCUs are created when the real-world action or state change of one HCU candidate is not fully covered by the action or state change the other HCU candidate expresses. For example, if one HCU candidate expresses the invasion of two countries, but a second HCU candidate only mentions the invasion of one of the countries, two HCUs have to be created, one for each country.[3]

A similar procedure was used by van Halteren and Teufel (2003) for annotating factoids, and Nenkova et al. (2007) used the same intuition for creating Semantic Content Units (SCUs) in the pyramid method. However, splitting of existing content units tends to happen less frequently with HCUs than with SCUs, since no second HCU is created when two timeline entries merely describe different aspects of the same real-world event (e.g. a general statement about the impact of a natural disaster vs. the exact number of victims).

Figure 5.7 shows a screenshot of the database software used to create HCUs according to the schema in Figure 5.1 on page 83[4]. The example HCU shown merges two HCU candidates describing the same event (the invention of the first gas turbine) into a single HCU. Although the second HCU candidate additionally mentions that the gas-turbine **successfully** ran self-sustaining (a fact ommitted by the first HCU candidate), the two candidates can be merged into a single HCU, given that the additional information (the success) is non-conflicting information and does clearly not constitute a separate action or state change in the real world. Further examples of how HCU candidates are merged into HCUs are given in Figure 5.8 (see page 93).

5.2.3 Creation of links between HCUs and TimeML events

I will now describe how an HCU, once created, should be linked to TimeML events in the text which express the HCU's semantic content.

5.2.3.1 General principles

Anchor weights. I define anchor weights as numbers between 0 and 1. An anchor weight of 1.0 is used where a TimeML event fully expresses the semantic content of an HCU. This is the case, for instance, when the timeline writer has copied a text fragment from the source text. Many HCUs are only linked to a single TimeML event, and this one event is assigned an anchor weight of 1.0.

Anchor weights are normally assigned independently for each link to a TimeML event. This makes annotation fast and less complex. However, at evaluation time, this can lead

[3]Note that the HCU representing the invasion mentioned by two timeline writers will receive a higher HCU weight.

[4]The links from HCUs to TimeML events are not shown in this example.

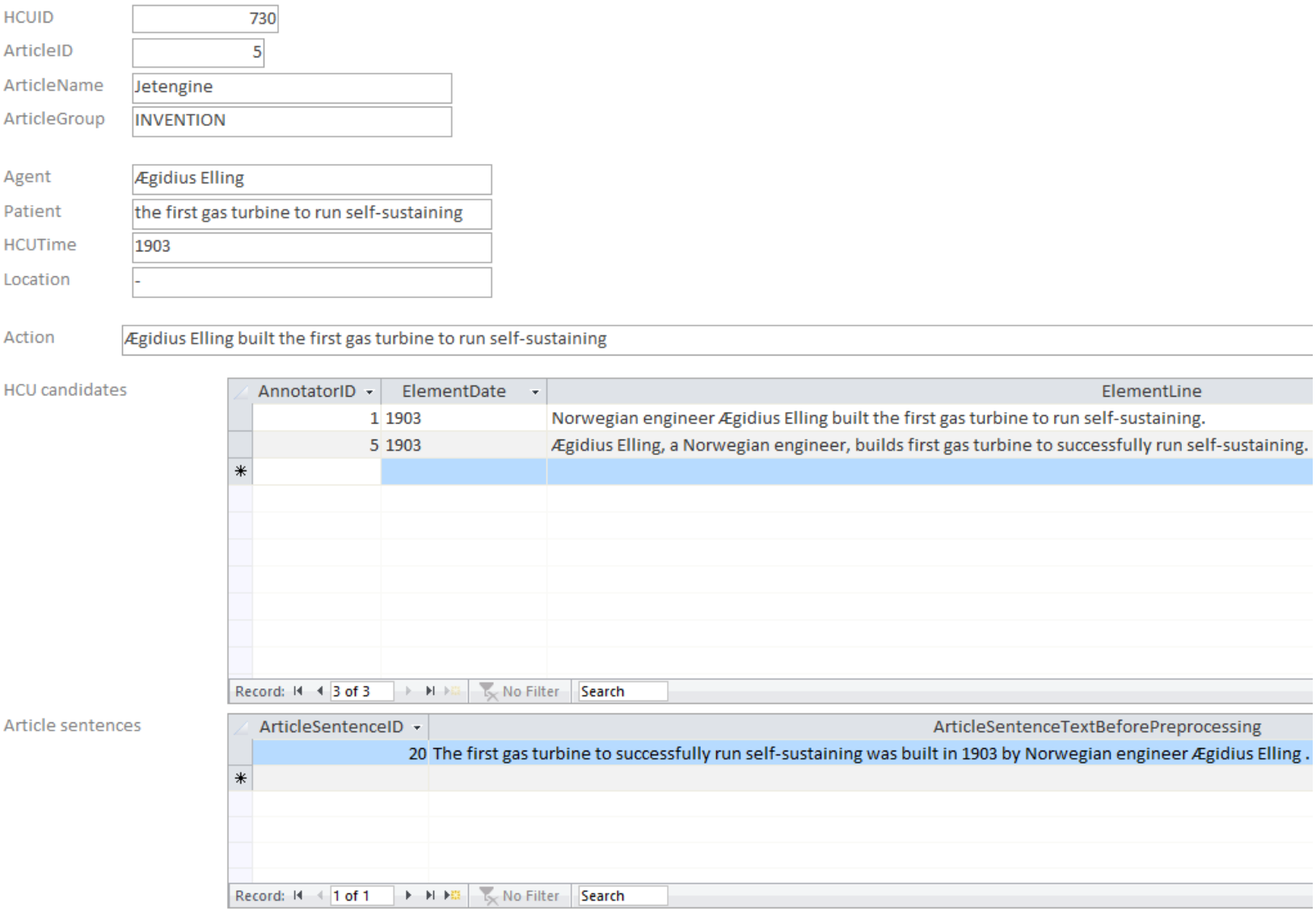


Figure 5.7: Example HCU for the article *History of the jet engine* created from two HCU candidates.

HCU	San'a' becomes the district capital
HCU candidate 1	Ottomans made San'a' the district capital
HCU candidate 2	San'a' was made Yemeni district capital
HCU candidate 3	San'a' becomes Yemeni district capital

HCU	A war between the Mutawakkilite kingdom of Yemen and the house of Saud breaks out
HCU candidate 1	The Mutawakkilite Kingdom of Yemen engaged in hostilities with the House of Saud
HCU candidate 2	War breaks out between Kingdom of Yemen and House of Saud
HCU candidate 3	Mutawakkilite kingdom – House of Saud war.

HCU	North and South Yemen make efforts toward unification
HCU candidate 1	Efforts were made for reunification of North and South Yemen.
HCU candidate 2	North and South Yemen agree to renew discussions of unification
HCU candidate 3	Real efforts towards unification

HCU	The government fights numerous insurgents
HCU candidate 1	The government fights numerous rebel groups.
HCU candidate 2	Government fights various insurgents

HCU	The Kingdom of Qataban is founded
HCU candidate 1	Qataban founded in Baihan Valley
HCU candidate 2	Establishment of Kingdom of Qataban (Baihan Valley)
HCU candidate 3	The Kingdom of Qataban was established

Figure 5.8: Examples of HCUs and the HCU candidates they were created from (Action fields only).

to a problem if the system timeline contains two TimeML events that express the same HCU, since the sum of the weights for all TimeML events related to a single HCU can be greater than one. This situation should be avoided, since the importance of an HCU during evaluation should not depend on how many TimeML events in the text express it. The reward that an algorithm can be given per HCU will therefore be capped to 1.0 during evaluation, as I will explain in Section 5.2.4 (cf. page 100).

Procedure. In a first step, for each HCU I manually identify all sentences that express its content fully or partly. I could have chosen to start by analysing individual TimeML events, but for human processing, the procedure is speeded up if the identification starts from sentences instead. Subsequently, each TimeML event in these selected sentences is analysed individually. A TimeML event is only assumed to represent an HCU if its context in the source text is consistent with the HCU's fields. However, it is not necessary for the content of each field to be explicitly mentioned in the text.

In cases of doubt, I use the intuition underlying Rappaport Hovav and Levin's definition of an *achieved state* to help me make the decision. Verbs of change of state lexicalise a particular achieved state in the real world and denote the bringing about of this state. For instance, the verb "cleaned" in "Peter cleaned the floor" entails a resulting change in the contacted surface (Rappaport Hovav and Levin, 1997). I assume that two events

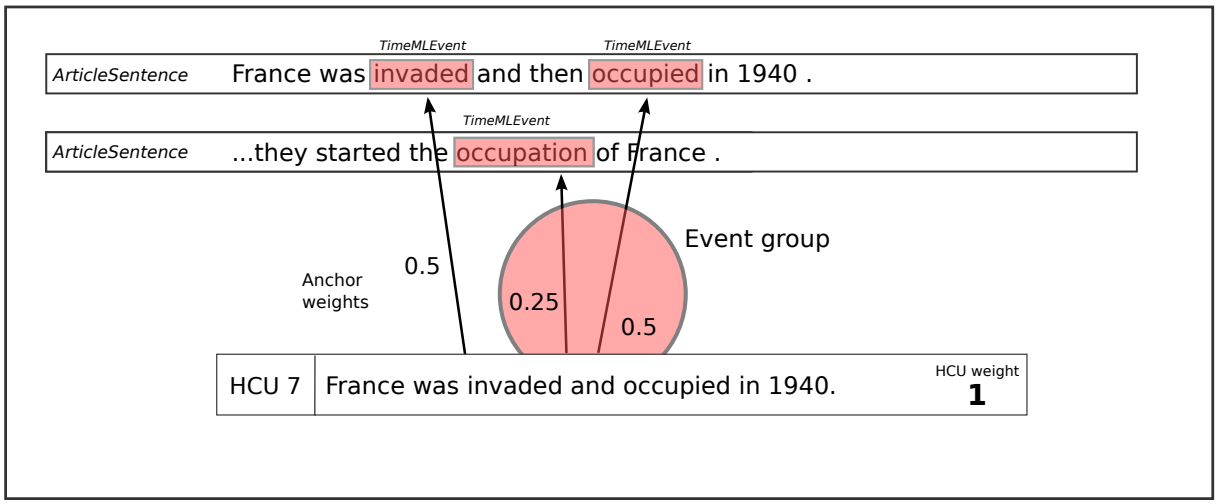


Figure 5.9: Event groups.

can describe the same achieved state in the real world even if they are lexically different. For instance, the two event descriptions "The Arabs conquered Spain" and "The Arabs established their rule in Spain" arguably entail the same achieved state, namely the beginning of Arab rule in Spain. When deciding whether an HCU is represented by a TimeML event, I compare the achieved state entailed by the TimeML event (taking into account its context) to that entailed by the HCU description and then decide whether they are the same.

Dependencies between events. Dependencies between TimeML events need to be modelled explicitly only if there is a group of events that express only part of any given HCU's semantics, even if selected together. In such cases, I use the concept of *event groups*. An event group is defined as a set of events which together do not express more than a certain fraction of the HCU's semantics. This fraction is called the event group's anchor weight.

Consider the situation in Figure 5.9. There are three events "invaded", "occupied" and "occupation", which all express part of the semantics of the HCU's semantic content "France was invaded and occupied in 1940", namely the invasion and subsequent occupation of France. The events "occupied" and "occupation", however, are reformulations of the same information, whereas "invaded" expresses a different portion of the HCU's semantics. Therefore, the HCU's semantics is fully expressed only if the algorithm has selected the event "invaded" and at least one of the events "occupied" and "occupation". Consequently, I use an event group containing the events "occupied" and "occupation".

Event groups can be defined recursively: Some of the events in an event group can be grouped together in a further event group, if taken together they express only a specific fraction of the HCU's content.

5.2.3.2 Guidelines

There are cases where the general principles given previously are not sufficient, since the relationship between an HCU and one or more TimeML events is more involved. In those cases, I use the guidelines below. They cover two groups of frequently occurring cases: Situations where it is not clear whether a given text fragment expresses the semantic

content of an HCU at all, and cases where it is difficult to decide which one out of several TimeML events should be linked to the HCU.

1. When multiple event words in the same clause express the HCU's semantic content (e.g. the two underlined events in "the king waged war against Russia" for an HCU with the description "War against Russia"), I assign an anchor weight of 1.0 to each event.

2. No links are created for verbs which, although they are used in an idomatic expression expressing an event, do not themselves lexicalise the real-world action or state change expressed by that event. For instance, the event "saw" in "the next century saw the crowning of King Peter" would not receive a link, only the event "crowning".

3. If there are multiple separate mentions of the same event in the text, only the first mention receives an anchor weight of 1.0. Later mentions receive an anchor weight of 0.5 if they are the main focus of their containing sentence. Otherwise, no link is created.

   The reason for this weighting decision is that only the first mention is *discourse-new*; all further mentions are *discourse-old*. These concepts are taken from the theory of *information status* (Prince, 1992). Discourse-old events have already been evoked in the discourse, while discourse-new events have not. Discourse-old events are typically less suitable for creating a timeline entry, as they rely on the reader's knowledge of the document context.

   In the following example, the first instance of "conquered" is discourse-new and the second is is discourse-old. The second instance is the main focus of the second sentence and would thus receive an anchor weight of 0.5.

   ... and conquered France. (...) France was conquered by ...

   Conversely, in the following example, the second TimeML event ("conquest") is not the main focus of the sentence. Here, no link would be created.

   ... and conquered France. (...) After the conquest of France by (...)

4. No links are created for events that are attributed or presented as uncertain. The event "tortured" in the sentence "According to government sources, the president was tortured" is an example of such a case.

   The likely reason why the author used such a construction is that it is controversial whether the event mentioned indeed took place. My event model, however, requires that an action or state change must be true beyond doubt according to the text.

5. If an HCU description describes content mentioned in the source text using a synonymous expression, an anchor weight of 1.0 is awarded. I consider two descriptions as synonymous if they can be assumed to refer to the same real-world event in the given context, e.g. in the case of "allowed" vs. "permitted" or "ties the knot with" vs. "marries".

When I found it difficult to decide whether two events are synonymous, the similarity score between the two event words as calculated by word2vec (Mikolov et al., 2013) is used as a guide; for the examples in my corpus, a similarity score of 0.4 or higher seems to indicate synonymity.

6. Where an entity mentioned in the HCU description involves metonymy in the source text or vice versa, I award an anchor weight of 1.0. Metonymy is a phenomenon whereby a concrete concept refers to a more complex, abstract concept to which it is closely related (Gibbs Jr., 1994). For instance, a sentence in the source text might state that a scientist "contacted London". Here, the reference to the capital functions as a metonymy for the British government. The corresponding HCU description could mention that the scientist contacted the government.

7. Events which plausibly entail the action or state change described by the HCU are assigned an anchor weight of 1.0. The entailment must be immediately obvious to a reader regardless of background knowledge without the need for additional interpretation. For instance, the two descriptions "the last monarch of France was executed" and "the French monarchy ended" can be assumed to refer to the same real-world event. Although the execution of the last king of France and the end of the monarchy are technically distinct, the first statement directly implies the second and would thus receive an anchor weight of 1.0.

8. Where the text describes the beginning of a historical event, there is a potential problem because two verbs are present: a verb indicating the beginning (such as "start" or "begin") and the actual event verb. But of course this combination does not correspond to two separate events. My solution is to assign an anchor weight of 1.0 only to the actual event verb, while no link is created for the verb indicating the beginning, unless the HCU description places emphasis on the beginning of the event. If the emphasis is on the beginning of the event, an anchor weight of 1.0 is awarded to the verb indicating the beginning, while the main verb receives an anchor weight of 0.5.

9. When the source text mentions the end of an event described in the HCU, I award an anchor weight of 1.0 to both the description of the ending and the main event word. For example, both "annexation" and "completed" in the sentence "The annexation was completed by 1564" receive an anchor weight of 1.0 if the HCU describes the annexation. The reason for this rule is that an action and its completion result in the same achieved state.

10. If timeline writers generalise one or more concrete events in the source text, the anchor weight is distributed evenly across all these events. An example of such a case is where the article mentions a series of disputes in a royal family, and the human writer abstracts away from these individual happenings by creating the timeline entry "quarrels in the royal family".

11. Where the source text first describes a fact in general terms and then mentions a special case (such as "New machines were invented in the 19th century. This includes the steam engine."), but the HCU description only discusses the special case (here: "the steam engine was invented"), the TimeML event expressing the general fact (here: "invented") is linked to the HCU. I use this rule since the semantic

content expressed by the HCU is best represented by the general statement (here: an invention). TimeML events occurring in the context of the special case (here: "includes") are irrelevant.

12. If the HCU description contains a plural noun phrase for which a collective reading[5] applies (such as for "A and B married"), no link is constructed to a TimeML event whose context fails to mention the entire set of entities denoted by that phrase. The same applies in the reverse case, i.e. when there is a collective plural noun phrase in the source text and the HCU description omits some of the entities denoted by that noun phrase.

    Noun phrases for which a distributive reading applies are treated differently. If the context of a TimeML event omits some of the entities denoted by the corresponding noun phrase in the HCU description, the TimeML event is assigned an appropriately reduced anchor weight to reflect the fact that it does not express the HCU's semantic content fully. An anchor weight of 1.0 is awarded, however, where the missing entities are negligible. Consider the following example, where for an HCU with the description "Gutenberg and his colleagues invent movable type", the corresponding text in the article reads as follows:

    "It is traditionally summarized that Johannes Gutenberg, of the German city of Mainz, developed European movable type printing technology with the printing press around 1439 and in just over a decade, the European age of printing began. However, the details show a more complex evolutionary process spread over multiple locations. Also, Johann Fust and Peter Schöffer experimented with Gutenberg in Mainz."

    Here, despite the HCU description stating that Gutenberg made the invention in collaboration with colleagues, the involvement of colleagues is arguably a negligible detail.

    In the reverse case, i.e. where the HCU description omits some of the entities denoted by a plural noun phrase in the TimeML event's context, an anchor weight of 1.0 is assigned, since the TimeML event still expresses the HCU's semantic content fully.[6]

13. Where the text explicitly describes a cause-and-effect relationship between two historical events, no link is constructed for a TimeML event which expresses this relationship, unless that event is itself part of the HCU description.

    For example, for an HCU whose semantic content is "Italy was unified", no link is created for the verb "led" in the sentence "and this led to Italy being unified".

[5]Plural noun phrases can have a *distributive reading* or a *collective reading*. A distributive or non-collective reading means that what a plural noun phrase says about a set can be paraphrased exhaustively as a conjunction of predications of the individual members of this set; this is not possible with a collective reading (Kamp and Reyle, 1993). The sentence "The inhabitants built a town hall" has a collective reading: Every inhabitant made a contribution to the construction of the town hall, but only the synergetic effect of their concerted action allowed them to succeed (Kamp and Reyle, 1993).

[6]Note that my evaluation methodology is inherently limited where a TimeML event is syntactically linked to a plural noun phrase, but two distinct HCUs have been created for subsets of the entities denoted by that noun phrase. For instance, it is possible that the two invasions mentioned in "Nazi Germany invaded Poland and Denmark" in the source text are represented by different HCUs. As HCUs are linked to single event words (and not to their arguments), there is no way to take this fact into account during evaluation. I do not offer a special treatment for such cases as they occur very rarely.

This is an adequate choice since the semantic content of the HCU is the process of unification, not an action that preceded the unification.

14. When the syntactic construction containing the TimeML event is modified by a *control construction* or a *raising construction*[7], the assignment of anchor weights depends on whether that construction is mentioned in the HCU description.

    No links are created for introducing verbs which are not mentioned in the HCU description (e.g. the verb "happened" in the construction "the king happened to arrive early"), as a positive context arguably adds nothing substantial to the HCU semantics. Here, an anchor weight of 1.0 is assigned to the subordinated event.

    If the HCU description also mentions the introducing verb, an anchor weight of 1.0 is given to the introducing verb, and an anchor weight of 0.5 is assigned to the subordinated event.

    However, no link is constructed for the subordinated event if it is negated or invalidated by the introducing event. Examples of introducing verbal constructions that show this behaviour include "prevented from", "refuses to" and "demanded in vain that". This rule is necessary because syntactically dominating verbs can change the truth-conditional status of a sentence.

    One example is a sentence in the source text describing the fact that country A prevented country B from invading country C. Here, although the corresponding HCU description will mention the invasion and there is a TimeML event which describes that invasion, it is a truth-condition of the sentence that the invasion did **not** take place. It would therefore be wrong to construct a link for the event describing this negated invasion.

15. Some TimeML events express a syntactic modification which is part of an HCU description. I call such TimeML events "background information"[8]. In contrast, TimeML events in the text that express the main predicate of the HCU are called "foreground information". Events referring to background information receive an anchor weight of 0.5, while the corresponding foreground information receives an anchor weight of 1.0. For instance, there may be an HCU with the following description:

    "Territorial expansion was put to an end by the death of Suryavarman II"

[7] Control and raising constructions consist of an intransitive matrix clause with an infinitival complement (for instance, "Barnett seemed to understand the formula") (Davies and Dubinsky, 2008). Control and raising constructions have different thematic structures: In raising constructions such as "Barnett seemed to understand the formula", the matrix subject ("Barnett") has a thematic role only in the action of the inifinitival complement ("understand the formula"). The predicate "seem" does not assign a thematic role to "Barnett". They are called "raising constructions" because the subject of the understanding, "Barnett", is raised to the main clause (Carnie, 2002). On the other hand, in a sentence such as "Barnett tried to understand the formula", both "try" and "understand" assign a thematic role to "Barnett" (Davies and Dubinsky, 2008). Such constructions are called *control constructions*, because the subordinate clause ("to understand the formula") is assumed to contain a special null NP which is *controlled* by the subject of the main clause ("Barnett") (Carnie, 2002).

[8] As I explained on page 89, the existence of background material in a sentence is a side effect of my decision to bundle it into an HCU if it is a syntactic modification of a main event which is essential to the semantics of the event.

Here, the death of Suryavarman II is arguably an important part of the event's semantics, as the territorial expansion was stopped directly by his death. It was therefore made part of the HCU description during HCU creation. However, the territorial expansion is the core semantic content of this HCU.

16. Where the core semantic content of an HCU occurs as the subordinated verb of a periphrastic causative[9] (e.g. "do" in "make someone do something") or as the past participial complement of a causative verb (e.g. "done" in "have something done"), only the subordinated verb is given an anchor weight of 1.0. For instance, the HCU description may say that "King Mark resigned", but the text contains the periphrastic causative construction "his son had King Mark resign". In this case, no link is created for the causative "had", as the subordinated verb ("resign") expresses the actual semantic content (the resignation).

    If, however, the causative verb is part of the HCU description, an anchor weight of 1.0 is awarded to the causative verb, and an anchor weight of 0.5 is awarded to the subordinated verb. In the example above, this would be the case if the HCU description also mentioned the role of the son in King Mark's resignation ("his son had King Mark resign").

17. A historical event and its later discovery are assigned the same anchor weights. For instance, a TimeML event describing the discovery of a temple receives an anchor weight of 1.0 if the corresponding HCU represents the construction of that temple in Antiquity.

    This is a relatively common phenomenon, as articles tend to mention the recent discovery of historical evidence (e.g. the excavation of an ancient city in the 20th century). The discovery provides evidence that an earlier event took place. The importance of the discovery is directly tied to the importance of the original event.

5.2.3.3 Examples

Figure 5.10 gives two examples of HCUs for which some anchor weight assignments were non-trivial. I consider an assignment as trivial if the textual description in the HCU Action field literally corresponds to text in the source text, does not contain background information, and can be straightforwardly linked to a single TimeML event using an anchor weight of 1.0.

In the first example in Figure 5.10, the HCU Action description contains background information (the terrorist attack of 2001) in addition to the foreground information (the killing of 65 state residents). The TimeML event representing the background information ("attack") is therefore given an anchor weight of 0.5, while the event representing the killing ("killed") is assigned an anchor weight of 1.0.

In the second example, multiple concrete events (conquests and annexations of individual cities and cities etc.) were generalised into a single timeline entry "Carolingian conquest", which led to the creation of a single HCU. Here, since no TimeML event describes the entire conquest, the HCU's anchor weight is distributed across all TimeML events describing actions that are arguably part of the Carolingian conquest. As there are seven such events in this case, each of them receives an anchor weight of $\frac{1}{7}$.

[9] A periphrastic causative is a construction such as "they have someone do something" that consists of a noun phrase, a causative verb (such as "get", "have" or "make"), another noun phrase, and an infinitive (Hollmann, 2003).

HCU 113	
Action	65 state residents were killed in the 9/11 terrorist attack
Agent	–
Patient	65 state residents
Time	11/09/2001
Location	–
TimeML events	In the terrorist **attacks (0.5)** of September 11 , 2001 , 65 state residents were **killed (1.0)** .
Timeline entries	65 state residents were killed in the terrorist attack.
	65 state residents were killed in the terrorist attacks
	65 state residents were killed in the terrorist attacks on the World Trade Centre

HCU 385	
Action	The Carolingians conquer Catalonia
Agent	the Carolingians
Patient	Catalonia
Time	760-801
Location	Catalonia
TimeML events	The first county to be **conquered** ($\frac{1}{7}$) from the Moors was in the former area of Septimania that became Roussillon -LRB- with Vallespir -RRB- in around 760 .
	In 785 the county of Girona -LRB- with Besalú -RRB- on the south side of the Pyrenees was **taken** ($\frac{1}{7}$) .
	Ribagorza and Pallars were **linked** ($\frac{1}{7}$) to Toulouse and were **added** ($\frac{1}{7}$) to this county around 790 .
	Urgell and Cerdanya were **added** ($\frac{1}{7}$) in 798 .
	After a series of struggles , Charlemagne 's son Louis **took** ($\frac{1}{7}$) Barcelona from the Moorish emir in 801 and **set** ($\frac{1}{7}$) up the County of Barcelona .
Timeline entries	Carolingian conquest

Figure 5.10: Examples of anchor weight annotations for two HCUs.

5.2.4 Scoring system summaries

Once anchor weights have been assigned, HCUs can be used to score system summaries. I will now describe how this is done in practice.

Pyramid scores. System-generated timelines are scored using an adapted version of the pyramid score proposed by Nenkova et al. (2007). The difference is that the extent to which the timeline covers an HCU can now be calculated automatically from the anchor weights. In contrast, the original pyramid method requires a human expert to analyse which gold-standard SCUs are covered in each individual system summary.

My metric $P_d(T, nHCU)$ is a variation of the pyramid score in Nenkova et al. (2007) and defined as follows:

$$P_d(T, nHCU) = \frac{\sum_{h \in H_d} score(h, E_h, T)}{maxscore(d, nHCU)} \tag{5.4}$$

where d refers to an article, T is the set of all TimeML events in the system-generated

timeline, H_d is the set of all HCUs annotated for an article d, E_h is the set of all events for which a link to HCU h exists, and $nHCU$ is the number of HCUs that an ideal timeline is expected to express. I set $nHCU$ to the number of TimeML events n that the algorithm is allowed to select, given that very few TimeML events in the source text are linked to more than a single HCU, as I will show experimentally in Section 5.3.7 (cf. page 107).

The maximum score that an algorithm can receive for an HCU is calculated analogously to the original pyramid method, with HCUs taking the place of SCUs:

$$maxscore(d, nHCU) = \sum_{i=j+1}^{nt} i \cdot |T_i| + j \cdot (nHCU - \sum_{j+1}^{nt} |T_i|) \tag{5.5}$$

where

$$j = \max_i (\sum_{t=i}^{nt} |T_t| \geq nHCU) \tag{5.6}$$

and i is the HCU weight as defined in Section 5.1.2, i.e. the number of timeline writers who expressed an HCU in their timelines. The score that an algorithm obtains for a single HCU h is defined as follows:

$$score(h, E_h, T) = w_h \cdot Cov(h, E_h, T) \tag{5.7}$$

where w_h is the HCU weight of HCU h. The maximum weight possible is equal to the number of timeline writers.

The function $Cov(h, E_h, T)$ is central to algorithm scoring. I use it to calculate the extent to which the semantic content of HCU h is expressed by the events T chosen by the algorithm. For instance, an HCU h may have links to two TimeML events. Both these links have an anchor weight of 0.5. Assuming that a system timeline to be scored only contains one of those events, the function $Cov(h, E_h, T)$ will return a score of 0.5.

The basic version of $Cov(h, E_h, T)$ is defined as follows:

$$Cov(h, E_h, T) = min(1.0, \sum_{e \in E_h} v_{e_j} \cdot s(T, e_j)) \tag{5.8}$$

where v_{e_j} are the anchor weights between 0 and 1 that were previously annotated, and $s(T, e_j)$ is a helper function indicating whether the set of events T in a system timeline includes event e_j:

$$s(T, e_j) = \begin{cases} 1 & \text{if } e_j \in T \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

As described in Section 5.2.3.1, the anchor weights v_{e_j} are normally assigned separately for each TimeML event, but the total score a timeline can obtain for an HCU is capped to 1.0 (cf. Equation 5.8). This upper limit prevents a timeline from being rewarded for containing redundant information.

Recall that in many cases, there is only a single TimeML event which expresses the HCU's semantics, and its anchor weight is 1.0. In those cases, Equation 5.8 simplifies to:

$$Cov(h, E_h, T) = s(T, e_j) \tag{5.10}$$

where e_j is the one TimeML event that fully expresses the HCU's semantics.

The total score a system timeline receives is the sum of the individual scores calculated for each HCU, as in the original pyramid method.

Event groups. The score in Equation 5.8 is calculated by summing the anchor weights v_{e_j} of all events the algorithm has chosen. In some cases, however, there are dependencies between TimeML events that need to be modelled using event groups, as described in Section 5.2.3.1 (see page 94).

Recall that event groups are defined recursively. Formally, an event group E_k can be written as

$$E_k = \{e_1, e_2, ..., E_1, E_2, E_3, ...\} \tag{5.11}$$

where e_i refers to individual events and $E_1, E_2, E_3, ...$ are further event groups.

Due to the introduction of event groups, Equation 5.8 has to be revised. The score that a system timeline is awarded for a given HCU is calculated as follows:

$$Cov(h, E_h, T) = min(g_{E_h}, \sum_{E_{h_j} \in E_h} Cov(h, E_{h_j}, T) + \sum_{e \in E_h} v_{e_j} \cdot s(T, e_j)) \tag{5.12}$$

where g_{E_h} is the anchor weight of event group E_h, E_{h_j} are further event groups that are part of event group E_h, and E_h is now a special top-level event group with $g_{E_h} = 1$ that is implicitly defined for every HCU[10].

5.3 Construction of an evaluation resource

Using the design described in the aforementioned section, I constructed an evaluation resource from human-written timelines.[11] I will now describe the details of this process.

5.3.1 Selection of input texts

I will first describe how input articles for the resource were selected.

Selection principles. It is important to choose articles that are representative of all history articles for which timelines can be constructed. Only Wikipedia articles contained in the corpus presented in Chapter 3 were considered, for two reasons: First, these articles can be guaranteed to exclusively describe historical content, as their titles start with "History of". Second, the presence of a corresponding gold-standard timeline indicates that humans find a timeline about the article's topic relevant.

Articles were selected such that a variety of *subject areas* were covered; examples of subject areas are "Invention" or "Sport". The subject area of each article was identified manually.

I tried to approximate the distribution over subject areas in the timeline corpus presented in Chapter 3. This distribution was skewed, with more than 80% of all articles belonging to the subject area "Geo-political entity" (GPE). The next most frequent subject areas were "Field of science" and "Invention". For most subject areas, only one or two articles were available.

[10] If no event groups have been annotated, this top-level event group merely contains all events for which links to the relevant HCU exist.

[11] The evaluation resource can be downloaded from `http://www.cl.cam.ac.uk/~smb89/form.html`.

For each subject area, the average number of sentences per article was calculated. I then selected, for each subject area, those articles whose number of sentences was close to the average. Overly long articles take considerably longer to annotate, which is prohibitive given the already high annotation effort. Many very short articles skip entire time periods as they are incomplete.[12]

Exclusion criteria. Articles which exhibited one of the following problems were excluded:

- articles containing text that is already arranged in the form of a timeline;
- articles that contain empty sections, given that the presence of an empty section often entails that a particular time period is not covered by the text;
- non-chronological articles, where I declare an article as non-chronological if more than half of any section describes events that took place earlier in time than the events described before that section;
- articles very similar in terms of topic to an article already chosen.

In contrast, generally low text quality was not a reason for removing an article. There is no theory of text quality that can be automatically applied, and hence any such filtering would have been subjective.[13] More importantly, such a manipulation would have negatively impacted the general applicability of my algorithm to *any* history article in principle.

Resulting selection of articles. Timelines for 11 articles were collected, six describing the history of geo-political concepts (GPE), two on the history of inventions, one on the history of a field of science, and one article describing the history of a food item. Table 5.11 contains statistics in terms of sentences, words and verbs.

Number of timelines per article. Three timelines per article were elicited. This choice was a compromise which allowed me to collect pyramids of acceptable size for a reasonably large number of articles.

Choice of timeline lengths. The rightmost column in Figure 5.12 gives the hard upper limit on the number of entries for each article. Interim results in red color were calculated using Equation 5.3 (cf. the procedure described on page 86).

When performing the experiments in Chapter 6, I soon discovered that one of the most deciding factors for algorithm performance was whether or not an event was close to a date. This is a problem since it means that even a baseline which randomly selects events with an associated date can perform very well. For four articles in the corpus, the number of events with an associated date was so low that randomly selecting from them

[12]The probability that an article is incomplete is higher for short than for long articles. However, not every short article is necessarily incomplete. There are a high number of well-maintained articles that are shorter than others because the topic discussed is very specific.

[13]It would have been possible to filter out articles which the Wikipedia community has marked as low-quality. For instance, some articles contain a warning such as "This section is outdated. Please update this article to reflect recent events or newly available information". But ultimately, such classifications are subjective too.

ID	Topic	Subject area	Sentences	Words	Verbs
1	Yemen	GPE	209	5513	646
2	Connecticut	GPE	206	5395	605
3	Cambodia	GPE	205	5061	573
4	Paleontology	Field of science	178	5604	657
5	Jet engine	Invention	94	2655	352
6	Wine	Food	135	3741	440
7	Wales	GPE	183	5117	615
8	Melbourne	GPE	223	5593	599
9	Namibia	GPE	203	5512	637
10	Catalonia	GPE	196	6786	567
11	Printing	Invention	286	6835	956

Figure 5.11: Statistics for the articles in the test corpus.

ID	Topic	Sentences	Verbs	Target # events	Max. # events
1	Yemen	209	646	27.86	**35**
2	Connecticut	206	605	26.09	**35**
3	Cambodia	205	573	25	**30**
4	Paleontology	178	657	28.33	**35**
5	Jet engine	94	352	15	**20**
6	Wine	135	440	18.97	**25**
7	Wales	183	615	26.52	**35**
8	Melbourne	223	599	25.83	**35**
9	Namibia	203	637	27.47	**35**
10	Catalonia	196	567	24.45	**30**
11	Printing	286	956	41.22	**50**

Figure 5.12: Target and maximum numbers of timeline entries.

would have resulted in a performance close to the maximum score. For one article, the number of events with an associated date was even lower than the maximum number of events to be chosen for the timeline.

The problem was caused by the fact for those articles, my estimation of the maximum number of events was too high, and that consequently, timeline writers were asked to write too many timeline entries. The estimate would have been more accurate if I had based it on the number of dates rather than on the number of verbs.

I resolved the problem in the following way: The article where the number of events with an associated date was lower than the maximum number of events was excluded from the corpus. For the remaining three affected articles, I went back to the original timeline writers and asked them to remove a specified number of entries. Before doing so, I calculated a new upper limit $maxEvNew$ on the number of timeline entries for each article affected as follows:

$$maxEvNew = \bar{r} \cdot EvWithDate \tag{5.13}$$

where $EvWithDate$ is the number of events with an associated date, and $\bar{r}$ is the ratio between the maximum number of events and the number of dated events in the seven articles for which the problem did not occur. It is obtained as the average of a ratio r calculated separately for each of the seven articles, which is defined as

ID	Topic	Max. # events (original)	Max. # events (corrected)
1	Yemen	35	35
2	Connecticut	35	35
3	Cambodia	30	30
4	Paleontology	35	**22**
5	Jet engine	20	**12**
7	Wales	35	35
8	Melbourne	35	35
9	Namibia	35	35
10	Catalonia	30	30
11	Printing	50	**25**

Figure 5.13: Corrected maximum numbers of timeline entries.

$$r = \frac{maxEv}{EvWithDate} \tag{5.14}$$

where $maxEv$ is the maximum number of events for an article, and $EvWithDate$ is the number of events with an associated date in that article. Figure 5.13 shows the corrected maximum number of timeline entries in bold font for the three affected articles.

5.3.2 Participants

The data collection was performed using 30 human timeline writers, who agreed to write one or two timelines each. Participants included postdocs and graduate students of the University of Cambridge. Not all were native speakers of English, but they all had lived in English-speaking countries for a minimum of one year (most of them much longer) and were used to working with scientific English on a daily basis. While a majority of the participants were computational linguists, not all of them were.

5.3.3 Materials

Human participants were provided with hard copies of the following:

- the article to be summarised;
- the set of guidelines on how to complete the task (cf. Figure 5.3 on page 85);
- ruled paper on which to write the timeline entries (timeline writers were free to use them or use their own paper).
  Each line on the ruled paper contained a short field for the date of the event, and a long field spanning the rest of the line which was intended to be used for the textual event description.

The articles were pre-processed before the experiment. Introductory sections which appear before the table of contents were removed. These often contain a form of summary already, which should not be available to the timeline writers at the time of writing. I also removed hyperlinks and images. These could have distracted the reader and/or provided unwanted cues. For example, some readers might have considered an event to be less

important only because the mention of the event's agent (i.e. some historical figure) was not linked to the Wikipedia article describing that figure. I wanted the timeline writers' judgments to be based only on the running text.

5.3.4 Procedure

The data collection was carried out in December 2014 and January 2015. Each article was assigned to timeline writers who were unlikely to have prior knowledge about the history of the concept described in the article. I wanted to avoid a situation where the timeline writers' choices of events could have been influenced by their prior knowledge.

Timeline writers who were not present in Cambridge at the time of the data collection were sent the materials via email, and were asked to print the materials before starting to create timelines. They were free to work on the task in their own time.

Upon return of the created timeline(s), I asked the timeline writers to give an estimate of the total time required, and recorded any comments the timeline writers chose to voice to me. The creation of a timeline for a medium-sized article took about two hours on average (including the time required for reading the source article). Timeline writers seemed to apply different strategies when composing a timeline. Some writers first read the entire article and then selected events in a separate step. Other writers started by marking up a large number of events while reading the source text, and then removed some of them in further iterations, until the limit on the number of timeline entries was respected.

No written or oral instructions apart from the written guidelines were standardly given. A small number of timeline writers who asked for clarification of the guidelines were provided with explanations that were as close as possible to the written guidelines.

5.3.5 Characteristics of gold standard

Figure 5.14 lists the number of timeline entries that the participants wrote for each article. For most articles, the average number of lines written is slightly below the maximum allowed, but note also that some participants exceeded the maximum number of events. Since the deviation was only marginal, I chose to accept the slightly longer timelines without asking the participants to correct their timelines.

A further observation is that timeline entries are typically short in terms of words: The average entry in the human-written timelines is only 11 words long and only consists of a noun phrase or main clause.

5.3.6 Creation of HCUs

I will now give statistics for the creation of HCUs that I performed based on the human-written timelines. This step was carried out in January 2015, using a standard relational database. Creating HCUs for a single article required about three to six hours of processing time, depending on the length of the source article. This estimate includes the time required to digitise hand-written timeline entries.

Figure 5.15 lists the number of HCUs I created for each article. Longer articles tend to have a higher number of HCUs annotated, although the relationship is not strictly proportional. For instance, 82 HCUs have been annotated for the article on the history of Melbourne (which has a line limit of 35 entries), while only 25 HCUs exist for the

Article	Writers	Max. events	Number of lines written			Average no. of words
			min	avg	max	
Yemen	6	35	24	32.67	38	8.96
Connecticut	3	35	30	32.00	34	11.21
Cambodia	3	30	25	27.67	30	10.14
Paleontology	3	22	20	21.33	22	13.18
Jetengine	3	12	12	12.00	12	16.52
Wales	3	35	35	36.67	38	9.29
Melbourne	3	35	21	30.00	35	9.05
Namibia	3	35	21	27.00	31	11.87
Catalonia	3	30	23	26.67	30	9.47
Printing	3	25	25	25.33	26	10.69
Average						11.04

Figure 5.14: Statistics on the number of timeline entries (lines) written per article.

Article	Line limit	HCUs	Number of HCUs with		
			HCU weight 1	HCU weight 2	HCU weight 3
Yemen	35	76	46	22	8
Connecticut	35	72	43	23	6
Cambodia	30	73	48	19	6
Paleontology	22	46	31	9	6
Jetengine	12	25	14	10	1
Wales	35	72	31	27	14
Melbourne	35	82	59	18	5
Namibia	35	57	35	13	9
Catalonia	30	62	37	18	7
Printing	25	59	40	11	8

Figure 5.15: HCU statistics for each article in the corpus.

much shorter article on the history of the jet engine (where the line limit is 12 entries). Figure 5.15 also shows that using multiple timeline writers leads to the characteristic pyramid shape known from Nenkova et al.'s pyramid method (with HCUs instead of SCUs). Such a pyramid shape is obtained for all articles in the corpus, i.e. the number of HCUs per tier decreases from left to right in each line of Figure 5.15. This suggests that the final limits on the number of timeline entries were chosen sensibly.

5.3.7 Anchor weight annotation

I now provide statistics for the annotation of anchor weights performed. This step, which was completed in March 2015, took about one to two weeks for all articles. Note that this process would have required considerably more time if anchor weights could not be assigned independently of each other in most cases.

Figure 5.16 shows that the average number of TimeML events that an HCU is linked to is 1.04 (macro-averaged across articles). The number of TimeML events per HCU is lowest for the article "History of printing". This is due to the article's writing style. The text is written such that many important events, which timeline writers added to their timelines, are mentioned only implicitly. For instance, for a sentence starting with "A

Article	TimeML events per HCU
Yemen	1.13
Connecticut	1.08
Cambodia	1.15
Paleontology	1.20
Jetengine	0.88
Wales	1.04
Melbourne	1.02
Namibia	1.25
Catalonia	0.84
Printing	0.76
Average	1.04

Figure 5.16: Number of TimeML events per article.

Number of links to a TimeML event (per HCU)	Number of HCUs
0	121 (17.2 %)
1	468 (66.5 %)
2	97 (13.8 %)
3	14 (2.0 %)
4	2 (0.3 %)
7	2 (0.3 %)
Total	704

Figure 5.17: Number of TimeML events linked to per HCU.

dot matrix printer or impact matrix printer refers to a type...", timeline writers are likely to create a timeline entry describing the invention of the matrix printer, although the invention is never explicitly stated. In such cases, it is not possible to find a suitable TimeML event which expresses the HCU's content.

Figure 5.17 shows how many TimeML events HCUs are linked to. 82.8% of HCUs are linked to at least one TimeML event; 17.2% of HCUs are not linked to any event. The highest number of events that any HCU is linked to is 7. However, only a very small number of HCUs are linked to three or more TimeML events. An additional analysis (not shown in the table) revealed that for 57.0% of all HCUs without a linked event, there *is* in fact a suitable textual anchor somewhere in the text, but the word concerned was not recognised as a TimeML event.

Figure 5.18 shows how many HCUs TimeML events are linked to. 83.8% of all TimeML events are not linked to any HCU. The vast majority of linked events are linked to a single HCU; only 9 events are linked to two HCUs, and none to more than two. This suggests that an algorithm selecting k TimeML events is highly unlikely to express the semantic content of more than k HCUs.[14]

[14]This result reconfirms the assumption underlying the scoring mechanism introduced in Section 5.2.4 that an ideal selection of k TimeML events expresses exactly k HCUs.

Number of HCUs linked to (per TimeML event)	Number of TimeML events	
0	3707	(83.8 %)
1	708	(16.0 %)
2	9	(below 0.1 %)

Figure 5.18: Number of HCUs linked to per TimeML event.

5.4 Reliability of the resource

In Section 4.3, I mentioned that the evaluation of summarisation tasks is complicated by human subjectivity. This potential problem also applies to my evaluation methodology.

Any good evaluation methodology strives to prove that the underlying evaluation resource is replicable. I will now present my experiments investigating reliability.

5.4.1 Suitability of pyramids

Both the elicitation of gold-standard timelines and the subsequent creation of HCUs are subject to human subjectivity. It is theoretically possible that the task of timeline generation is so arbitrary that a completely different timeline would emerge if different timeline writers were used. Similarly, it could also be the case that my definition of an HCU would not enable other human experimenters than myself to create a reproducible set of HCUs.

I therefore analysed agreement between two independently created pyramids for one randomly selected article. To this end, a further three human timelines for this article were elicited, such that six timelines are available in total. There are six possible splits of the six timeline writers into two teams of three timeline writers each. For each split, a matrix like the one shown in Figure 5.19 and Figure 5.20 was created. The matrices summarise how many HCUs were chosen by m timeline writers in Team 1 (shown as rows) and by n writers in Team 2 (shown as columns), for all possible pairs of m and n.

Note that cell (0,0) is a special case. This cell refers to historical events in the source text that no writer has chosen to mention in their timeline. The number of such HCUs cannot be measured automatically, as I annotated (and indeed encountered material for) HCUs only for those historical events that at least one writer had expressed. I therefore estimated the value of this cell, called $avgEventFreq$, using the average number of TimeML events per HCU:

$$avgEventFreq = \frac{eHCU}{nHCU} = \frac{305}{100} = 3.05 \tag{5.15}$$

where $eHCU$ is the total number of TimeML events in sentences with at least one TimeML event for which a link to an HCU exists; and $nHCU$ is the number of HCUs created for the article. This calculation is based on the simplifying assumption that a sentence (and hence, any TimeML events contained in it) expresses the content of at most one HCU.

I now estimate the number of HCUs that could be constructed from the remaining

		Team 2			
		0	**1**	**2**	**3**
Team 1	**0**	87*	19	2	1
	1	18	15	10	1
	2	6	8	9	4
	3	1	0	3	3

Figure 5.19: Best split (with 94.1% of all HCUs falling into one of the grey cells).

		Team 2			
		0	**1**	**2**	**3**
Team 1	**0**	87*	17	6	0
	1	20	9	6	3
	2	8	14	5	1
	3	0	2	6	3

Figure 5.20: Worst split (with 89.8% of all HCUs falling into one of the grey cells).

sentences in the text:

$$\begin{aligned} approx &= \frac{eNoHCU}{avgEventFreq} \\ &= \frac{266}{3.05} \\ &\approx 87 \end{aligned} \tag{5.16}$$

where $eNoHCU$ is the number of TimeML events in sentences that do not express an annotated HCU. In Figure 5.19 and Figure 5.20, the value of cell (0,0) is marked with an asterisk to differentiate it from the cells whose values were directly observed.

Perfect agreement between the two teams would manifest itself in a diagonal matrix, i.e. for all HCUs, the number of timeline writers per team who chose a particular HCU would be the same in Teams 1 and 2, and all other cells would contain a zero. In reality, the weights are not fully identical, but only similar.

I now perform the following analysis in order to quantify agreement between the two teams. I assume that a positive case is where the two teams' frequencies differ at most by one; the cells which represent those cases are coloured grey in the two matrices. All other cases were assumed to reflect poor agreement. Figure 5.19 shows the split where the proportion of HCUs in the grey area is highest, while Figure 5.20 gives the split where it is lowest. On average across all splits, 91.9% of all HCUs fall into the grey area. This result suggests that the reproducibility of pyramids is acceptable to be used for evaluation of system summaries.

The above way of measuring agreement is more appropriate than more standard summary metrics such as Krippendorff's α or Fleiss' κ. What is being measured here is not inter-annotator agreement between a number of individual coders who annotate a set of coding units according to a provided coding scheme. Instead, agreement is measured between teams of fully independent timeline writers, each of whom writes a new timeline based on a textual input document.

However, to give the reader a rough idea of the agreement achieved if a summary metric were used, I calculated Krippendorff's α in the following non-standard way: Each team of three timeline writers was interpreted as a single coder; each HCU was treated

as a coding unit; and the value assumed to have been chosen by a team was the number of timeline writers in the team who expressed that HCU in their timelines. I assumed that HCU weights are values on an interval scale, and therefore used an interval difference function for α as proposed by Krippendorff (1980). The value for α achieved using this set-up was 0.530. This number, while not in the range of highest agreement, is an acceptable result given the high level of subjectivity. This number expresses the same facts as the figures calculated by simple agreement, but shows the effect of using much more rigorous summary metrics.

5.4.2 HCU weight judgement

I conducted a further small experiment in order to investigate whether a difference in HCU weight corresponds to a perceived difference in importance. I assume that this is the case if a person without knowledge of the annotated HCUs considers HCUs with a higher weight to be more important than HCUs with a lower weight. Knowledge of the source text is arguably not necessary for this task, as HCUs with a higher weight are expected to describe inherently more salient events.

Due to the subjective nature of HCU weighting, it is not adequate to test this hypothesis on individual HCUs, since two humans are fairly likely to perceive the importance of any single HCU differently. However, I expect these differences to average out when a human is presented multiple HCUs of the same weight simultaneously as a set.

I therefore construct two lists of HCUs for a given article. One list contains all HCUs of weight 1 and 3, while the other list is composed of all HCUs of weight 2 and 3. The HCUs with weight 3 were included in both lists in order to give the human judge information about the most important events in the history of the concept, so that they could relate the other less important events they were judging within the framework of the important events. Each HCU is represented by its date and its HCU Action description. In addition, the HCU weight is shown (as "(3)") if it is 3. The following is an extract of such a list:

...
1945 (3) World War II ended
1948 A new currency was introduced
...

The human judge (my supervisor Simone Teufel), who had not previously read the source texts, was given the two lists for an article simultaneously. For two randomly selected articles, she was able to determine correctly which of the two lists contained HCUs with a higher weight. This is a further reason to believe that HCU weight measures importance in a way that is sufficient to be usable for practical evaluation.

5.4.3 Inter-annotator agreement for anchor weight annotation

Another step in my evaluation methodology which is potentially affected by human subjectivity is the assignment of anchor weights. To confirm the usability of the guidelines presented in Section 5.2.3.2, a subset of anchor weight assignments was also performed by a second experimenter (my supervisor Simone Teufel), who was provided with the matching sentences for a subset of HCUs as identified by the first experimenter (myself). She independently assigned anchor weights to all pairs of HCUs and TimeML events in the

matching sentences (158 in total). In 87.9% of all cases, the same weight was assigned. The result of this experiment suggests that the guidelines designed by myself enabled acceptable reproducibility.

Outright mismatches (e.g., where one experimenter thought that an event represented an HCU, but the other did not) were mostly due to inference, as the decision whether one event entails another can be subjective.

I only give percentage agreement here. κ cannot be calculated because there is no fixed set of categories from which the annotator can choose (instead, a number between 0 and 1 is assigned). Note also that the two experimenters do not work under the same conditions. The first experimenter starts by selecting matching sentences and then assigns anchor weights to TimeML events in these sentences. In contrast, the second experimenter has to follow the first experimenter's choice of matching sentences.

Overall, the results obtained in the three experiments described in this section give reason to believe that the three central steps of my evaluation methodology can be reliably performed.

5.5 Construction of a development resource

The test corpus presented in Section 5.3 should not be used for day-to-day development of new algorithms, as systems must not be optimised on test data. I therefore constructed a separate corpus for the purpose of day-to-day development.

A radically simplified procedure was used to construct the development corpus, since the human effort with the full evaluation methodology followed up to now is considerable. With the original method, the following time-consuming steps have to be carried out:

1. Human timeline writers have to read and mentally process the source article.
2. Timeline entries need to be written by the timeline writers.
3. HCU candidates need to be created by an experimenter.
4. HCU candidates need to be merged into HCUs by an experimenter.
5. Links (including anchor weights) between HCUs and TimeML events need to be assigned by an experimenter.

It is not possible to reduce the time the human timeline writer spends reading the source text, as a good understanding of the article is the necessary preparation for writing a timeline. However, the other steps can be simplified at the price of lower quality. I made the following modifications to the original method:

- I performed the selection of important content as a single person instead of using multiple human timeline writers.
- I did not write textual timeline entries. Instead, I directly marked up important text spans in the source text.
- I did not create HCU candidates and HCUs using the procedure described earlier. Instead, a single HCU was directly constructed for each marked-up text span in the source article, without the prior creation of HCU templates as shown in Figure 5.1

on page 83.[15] In a second step, the HCU was linked to one or more TimeML events in that text span which expressed it. Anchor weights were assigned as in the original procedure.

The aforementioned simplifications lower the quality of the resource in the following ways:

- Due to the well-known subjectivity of content selection tasks, the simplified procedure is less reliable than the original method.
- It is possible that humans perform content selection differently when no textual timeline entries have to be produced.
- It is now no longer impossible by construction that two different HCUs refer to the same real-world event, as was guaranteed before. The main reason for this is that I no longer verify that two HCUs indeed refer to different events. In the full methodology, all HCU candidates that refer to the same event are merged into a single HCU. On the other hand, in my simplified procedure, HCUs are created independently of each other, one at a time. When there are two mentions of the same real-world event and both have been selected as important text spans, two HCUs will be created. A second reason is that HCUs are no longer created from human-written timelines. In the full methodology, it is extremely unlikely that a human timeline writer would create a timeline describing the same event twice.
- Due to the lack of textual timeline entries, a potential surface generation component of timeline generation algorithms cannot be evaluated. With the original method, this was a prospect for future work (although not performed here).

The simplified procedure however allowed me to create a development corpus with considerably more (30) articles than in the test corpus (10).

For selecting the articles, the same distribution over subject areas as in the test corpus was used. This results in 19 articles about geo-political entities, 3 articles about fields of science, 5 articles about inventions, and 3 articles about foods and drinks (as listed in Figure 5.21). The limits on the number of events to be selected were calculated in the same way as for the test corpus. Articles which met the exclusion criteria outlined in Section 5.3.1 (cf. page 103) were removed.

The development resource was used to tune the methods described in Chapter 6. In contrast, the test resource was held out during development and only used in the final evaluation.

5.6 Chapter summary

In this chapter, I introduced an evaluation methodology for timeline generation algorithms. This methodology combines the advantages of existing approaches in that it is based on semantic meaning units and, once an initial investment into evaluation resources has been made, enables free automatic evaluation of an unlimited number of system timelines.

[15] Creating and merging HCU candidates is not necessary here, since content selection is only performed by a single timeline writer.

Topic	Sentences	Verbs	Max. events
GPE			
Serbia	354	968	55
East Timor	252	717	40
New South Wales	293	968	55
Martinique	232	566	30
Lithuania	563	1618	85
Lyon	159	393	25
Warsaw	333	881	50
Prague	213	529	30
Grenada	140	384	20
Queensland	187	455	25
Honduras	416	1363	75
Taiwan	307	1004	55
Burma	382	1136	60
Singapore	288	878	50
Venezuela	182	525	30
United States	566	1669	90
Slovakia	322	859	45
Croatia	222	666	35
Falkland Islands	423	1510	80
Invention			
Hearing aids	113	345	20
Paper	121	450	25
Rockets	178	575	30
The camera	149	488	30
Weapons	298	1054	55
Field of science			
Artificial intelligence	314	1074	60
Botany	283	985	55
Optics	166	577	30
Food and drink			
Alcoholic beverages	238	765	40
Sugar	153	421	25
The hamburger	334	1161	65

Figure 5.21: Statistics for the articles in the development corpus.

The initial investment is admittedly substantial. It is a process that requires three major steps: First, human gold-standard timelines for a number of source articles have to be elicited. These articles are used, in a second step, to create Historical Content Units (HCUs) which abstract away from different wordings of the same content. The final step is the annotation of anchor weights between HCUs and textual anchors in the source text. Crucially, with this system no annotation of system summaries whatsoever is required, which allows for the automatic free scoring of new system summaries in day-to-day development. I hope that this property will considerably simplify and accelerate the development of new timeline generation algorithms, which was hitherto hampered by the difficulties posed by evaluation.

I have also presented two evaluation resources created based on my methodology. The

first resource was constructed by 30 human participants not familiar with the evaluation methodology. It can be used as a test corpus and has been made available to the research community. I used this resource to perform three experiments concerning the reliability of my methodology, and confirmed that there is reason to believe that the method is reliable.

The second resource can be used for development and tuning. It contains a much higher number of articles, but is not of the same high quality and suffers from a number of other methodological drawbacks.

In the next chapter, I will use the evaluation methodology presented here to compare more than 15 methods for timeline generation. With my methodology, it was very easy to evaluate such a high number of methods, since evaluation was now completely free, yet guaranteed to be meaningful and semantically-oriented.

Chapter 6

Algorithms for timeline generation

In this chapter, I investigate various methods that perform timeline generation, and evaluate them using the evaluation methodology and resources described in Chapter 5.[1] The chapter is structured as follows: In Section 6.1, I present simple, uninformed methods that exploit explicit document structure and the presence of dates. Section 6.2 describes informed methods that rely on external knowledge. In particular, I adapt the existing supervised approach by Chasin et al. (2014), which was introduced in Section 2.6. I also present unsupervised methods that combine subject-area-specific lexical tendencies, syntactic dependencies between a verb and a date, as well as co-selection constraints between multiple events. The sources of income are combined using integer linear programming. All methods are evaluated in Section 6.4. Although the supervised methods can exploit the additional development resource described in Section 5.5 for training, they are outperformed by a combination of three unsupervised methods.

6.1 Uninformed methods

I now present methods for selecting events which I call uninformed, as they are not dependent on linguistic analysis of the article text, machine learning approaches, or external background knowledge. In contrast, more complicated methods typically require training data or some form of external knowledge.

The particular uninformed methods used here exploit document structure and the presence of dates. I analyse to which degree these properties are predictors for the relevance of particular events for the timeline. The advantage is that such cues are easily obtainable for any input text, including for texts from niche domains, for which additional background information might not exist.

6.1.1 Section structure

History articles, similar to many other types of documents, are organised into sections. It seems reasonable to assume that timeline writers start a new section in the article once a drastic shift in history occurs. My first method tests the improvement which can be achieved by assuring a balanced selection of content from all sections. The timeline must necessarily contain events from all parts of the source article, as it is supposed to retrace the evolution of history throughout the time period described in the article.

[1] The work presented in this chapter has been published previously (Bauer and Teufel, 2016).

In particular, I use a round-robin method (henceforth called RANDOM RR) that chooses TimeML events randomly from each section in turn, until the maximum number of events n to be selected is reached. Once sections run out, the algorithm moves back to the first section.

However, aside from covering all sections of a text in the timeline, it is also important to establish where exactly in each section important content is likely to occur. One hypothesis is that sections tend to start with key events that should be added to the timeline. To test this, I use a modified version (FIRST RR) of the round-robin method introduced above which selects the first TimeML event from each section in turn. When no more sections are available, the second event of each section is selected, and so forth.

There is an analogy with the well-known first-n-words baseline used for summarising news articles. A news article typically starts with a description of the main event reported in the article. The rest of the article often contains further details and explanations of that event. For this reason, selecting the first n words (where n is the intended summary length) has proven to be a strong baseline for news summarisation. In an evaluation of 31 systems and 6 baselines performed by Nenkova (2005), only a single system outperforms this approach.

Alternatively, one could assume that important events tend to occur at the end of a section rather than at the beginning. This is also plausible as the last sentence of a section typically describes the result state of a longer series of events, such as the final settlement of a territorial dispute. Method LAST RR selects the last event in each section in a round-robin fashion.

I also test variants (FIRST SEN RR, LAST SEN RR) of the methods above that select events sentence by sentence in a round-robin fashion, instead of selecting *individual* events. In other words, the program would first select all events in the first sentence of the first section, then select all events in the first sentence of the second section, and so on. This reflects the intuition that the first sentence of each section should contain at least one important event. If all events from this sentence are selected at once, this important event would be guaranteed to be selected.

The aforementioned approaches are compared to simple methods that do not take section structure into account. Method FIRST GLOBAL is an adapted version of the first-n-words summariser from news summarisation. Instead of the first n words, I select the first n events in the article. Method LAST GLOBAL is a variation which selects the final n events in the article. Method RANDOM GLOBAL selects events completely at random. My hypothesis is that methods that use section structure should perform substantially better than these very simple methods.

6.1.2 Presence of dates

Another hypothesis is to assume that article writers tend to furnish important events with an explicit date. I therefore construct a further method (called SIMPLE DATES) which randomly selects events from sentences that contain at least one *absolute date*. This definition excludes relative dates such as "the year before", as well as other temporal expressions (e.g. "every other year").

There are reasons why even some important events may not be explicitly dated. If the date of an event (whether important or not) can be straightforwardly inferred because one or more dates occur in preceding sentences, the writer may choose not to use an explicit date to avoid unnatural-sounding redundant text. Another possible factor is the

communication technology of the era considered. Events in earlier time periods were often passed on orally, and the exact dates were not recorded. It is also possible that an event can only be inferred retrospectively in the first place and that no exact date can be given.

Despite these other possible factors, I expect that important events are more likely to co-occur with a date than less important events, and that it is often left to the reader to infer the dates of less central events from dates in the event's context.

In my method, date mentions are identified using the state-of-the-art temporal information extraction toolkit HeidelTime (Strötgen and Gertz, 2010), which is rule-based and analyses each sentence independently (cf. Section 2.3.2 on page 28).

6.2 Informed methods

In this section, I describe methods for timeline generation that require access to information beyond the input text itself, such as an annotated corpus of training timelines or a background corpus of historical texts.

6.2.1 Supervised approach by Chasin et al.

I first describe the modified version of the approach proposed by Chasin et al. (2014) (cf. Section 2.6) which I use here. Like the original method, I use an SVM model to perform a binary classification of TimeML events into important and unimportant events. In my case, a timeline is constructed by selecting the n events with the highest scores assigned by the system.

6.2.1.1 Features

I re-implementated all features proposed by Chasin et al. These include structural, linguistic and semantic features as well as named entity weights and the importance of the sentence containing the event as calculated using TextRank. However, I adapted most features since the original method assigns the same feature vector to all events in a sentence, which is undesirable. In the modified Chasin method, I define features on the event level rather than on the sentence level whenever logically possible. Only features that are inherently tied to the sentence as a whole (such as the TextRank score) are left unchanged.

There are some other differences. Chasin et al. did not clarify what keywords their final system uses for calculating the maximum WordNet similarity. They considered the following alternatives: (a) the first two nouns in the article, (b) the first noun and the first verb, and (c) the first noun and the next word that is either a noun or a verb. I chose to use the first two nouns in the article as keywords, since the first verb in my history articles was almost always a general verb such as "to be".[2] Typical keywords are "territory" and "settlements" for geo-political entities, and "production" and "beverage" for food articles. Another difference to the original implementation is the choice of event recogniser. While the work of Chasin et al. was based on Evita (Saurí et al., 2005), I use the publicly available state-of-the-art system TipSem-B (Llorens et al., 2010), which performs better than Evita (cf. the discussion in Section 2.3.2 on page 28).

The features I use are summarised in Figure 6.1. Bold font indicates a modification.

[2]The article's introductions, which are otherwise excluded from processing in my evaluation methodology (see Section 5.3.3 on page 105) were retained for the purpose of obtaining keywords.

Structural	Digit presence in the sentence
	Position of the sentence in the article normalized by the number of sentences in the article
	Length of the event word
	Number of events in the sentence
	Number of "to be" verbs in the sentence
Linguistic	**Aspect of the event word (perfective vs. others)**
	Percentage of events in the sentence that have been assigned class "occurrence"
	Negation of the event word
	Part-of-speech of the event word (verb vs. non-verb)
	Tense of the event word (past tense vs. other tenses)
Named entities	Sum of the named entity weights in the sentence
Semantic	**Maximum WordNet similarity of the event word to the first two nouns in the article**
TextRank	TextRank rank of the sentence in the article, normalised by the number of sentences in the article

Figure 6.1: Modified set of features used in method Chasin.

6.2.1.2 Training

25 randomly chosen articles in the development corpus were used for training, while the remaining 5 articles were used for development. The amount of training data available here is higher than in Chasin et al.'s original method, where only 13 articles were available in total. Many of the articles in their corpus were also considerably shorter than my articles, given that they discussed only a single war or battle rather than the entire history of a concept. For training the SVM models, I used the LibLINEAR software (Fan et al., 2008) package, in the same way as for the experiment in Chapter 3. The C and g parameters of the model were tuned using standard grid search and 5-fold cross validation over the 25 articles. The SVM model was used to produce a probability estimate for each TimeML event, following the method by Platt (2000). These estimates were used as the scores assigned to events.

6.2.2 Unsupervised approaches

In this section, I propose several unsupervised approaches to timeline generation. I use the following three ideas: First, I expect information about the subject matter of an article to be a good predictor of the events that should be included in the timeline, as certain types of events (depending on the subject area) are well-known to have a decisive impact on history. For instance, the history of a country is often shaped by the proclamation of a new constitution or a severe military conflict. I will describe this idea further in Section 6.2.2.1.

The second idea (Section 6.2.2.3) is that events which are syntactically attached to a date are likely more important than events merely co-occurring with a date. I use this additional criterion since there are ways of using a date that intuitively do not indicate that a given event is important (e.g. dates which describe a less important detail of an event that is mentioned in a subordinate clause; dates in brackets etc.). I also investigate whether there are certain years in world history that are more likely to be mentioned in a timeline regardless of the article's topic, since events of exceptional importance took

place in those years (Section 6.2.2.2).

The third idea is that certain events should not be selected concurrently (Sections 6.2.2.4 to 6.2.2.6). For example, it is likely that a timeline should not contain many events occurring in the same year.

All three ideas will be put to the test in Section 6.4, separately and jointly.

6.2.2.1 Typicality of events

Multi-document summarisers typically exploit redundancy in a large corpus, as described in Section 2.5. Conversely, single-document timeline generation algorithms have very little text to work with and no redundancy inside their own text base.

I therefore propose a method which acquires information about events typical for a given subject area. I expect that prior knowledge about the text's subject area can help identify important events, as, depending on the subject area, certain types of events are a priori more relevant for a timeline than others, as described above.

For instance, one can see that words such as "develop" and "professor" are overrepresented in articles on fields of science, while words such as "election" and "constitution" are highly frequent in articles on geo-political entities, for example.

In order to identify events typical of a given subject area automatically, I contrast the textual content of all Wikipedia history pages that belong to that subject area to the textual content in the rest of Wikipedia. To identify such pages, I use the following procedure: For geo-political entities, I use an exhaustive list of pages describing the history of countries, states and cities (e.g. "History of Austria", "History of Texas", ...). For all other subject areas, I exploit the fact that Wikipedia is organised as a hierarchy of *categories*. The category system of Wikipedia corresponds to a directed acyclic graph, given that each article can belong to multiple categories. I use categories to compile lists of articles that are relevant to a given subject area. The Wikipedia category is chosen manually (e.g. "History of food and drink" for the subject area "Food and drink") here, but this process could be automated.

Crucially, this approach has the advantage that it does not require individual events to be mentioned as being important in the Web corpus. Events which are globally important are not necessarily considered important by the author of a given history article, and vice versa. For example, the description of the history of a field of science in the 18th century might present a particular invention as being important for the subsequent development of the field, although this event may not be mentioned at all in a general corpus of history articles.

My method starts by calculating for each event in the source text how typical it is of the article's subject area. Typicality of an event can be estimated by comparison to all other subject areas as follows:

$$typ(e) = \frac{\sum_{w \in R(e)} \frac{fr(A, l(w))}{1 + dist(w_e, w)}}{C} \tag{6.1}$$

where $R(e)$ is a fixed-size window of words (not crossing sentence boundaries) around the event word in question, $l(w)$ is the lemma of word w, A is the set of all articles that fall into the article's subject area, $fr(A, l(w))$ is a frequency ratio (henceforth called *typicality score*) indicating how typical $l(w)$ is of subject area A, $dist(w_e, w)$ is the absolute distance in words between word w and the event word w_e, and C is a normalisation constant.

The typicality score for a single token w and a subject area A is calculated as

GPE	INVENTION	FOOD
absolutism (35.1)	gas-works (7623.2)	yerba (2638.5)
protectorate (33.8)	reverse-angle (5366.8)	hamburger (1992.5)
serfdom (33.0)	flashback (1571.1)	saffron (1958.0)
club (0.02)	season (0.1)	play (0.1)
game (0.02)	team (0.1)	member (0.3)
season (0.02)	school (0.1)	bear (0.3)

Figure 6.2: Words with high and low typicality scores for three subject areas.

$$fr(A, l(w)) = \frac{wordscore(A, l(w))}{wordscore(H, l(w))} \tag{6.2}$$

where H refers to the set of all Wikipedia articles whose title starts with "History of", and *wordscore* is a function that calculates the relative frequency of a word lemma in a number of articles:

$$wordscore(G, l(w)) = \frac{freq(G, l(w))}{freq(G, *)} \tag{6.3}$$

where $freq(G, l(w))$ is the number of times word lemma l appeared in the set of articles G, and $freq(G, *)$ is the total number of words in the set of articles. The normalisation constant C is defined as:

$$C = \sum_{w \in R(e)} \frac{1}{1 + dist(w_e, w)} \tag{6.4}$$

Normalisation is necessary since the number of words in $R(e)$ is not always constant, despite the fixed window size. This is because with a window size n, there are not always n words before and after the event word (e.g. where the event word is the first or the last word in the sentence). Without a suitable normalisation, an event at the beginning or end of a sentence would wrongly receive lower scores.

Figure 6.2 shows for three sample subject areas words with high and low typicality scores (calculated using Equation 6.2). As expected, words that are intuitively related to the subject area receive high scores (such as *absolutism* for geo-political entities and *hamburger* for food). Words which are unrelated to the subject area (such as *season* and *team* for inventions) receive low importance scores.

The method based on typicality scores (called Typicality in Section 6.4) selects the n events with the highest typicality $typ(e)$, where n is the desired number of events in the timeline.

6.2.2.2 Global importance of dates

Although the Web corpus cannot be expected to contain information about each and every event in the source text, it is possible that events from certain years are more likely to be mentioned in timelines generally, since events of exceptional global importance took place in those years. For instance, the end of World War II influenced the history of many countries and fields of science alike. It can therefore be expected that the year 1945 is more likely to be present in timelines than other years.

I therefore use a method which considers the frequency $glob(y(e))$ of the event's year in all sentences in Wikipedia, where e is the event in question, and $y(e)$ is the year in which event e took place. Each event is linked to a date; events taking place in years that are frequently mentioned are expected to be more important for the timeline.

Each event is linked to its date of occurrence using the following heuristic. As dates, I consider all absolute dates identified by HeidelTime[3]:

1. If the year in which the event took place is stated explicitly ("Charlemagne was crowned in 800"), I select this date. In order to identify these cases, I examine whether the event word is linked to a date via a dependency path in which the intermediate word is the preposition "in". If there are multiple such paths, I select the shortest one.

2. Otherwise, if the event word occurs in a subordinate clause and a syntactically connected event word in the corresponding main clause has an explicitly annotated date, that event's date is selected, as the two events are likely to have taken place at the same date. For instance, in a sentence such as "The Indian campaign of Alexander the Great began in 326 BC, when he conquered the Achaemenid Empire of Persia", the event word "conquered" would be assigned the year 326 BC because it syntactically depends on the word "began", which has an explicitly annotated date.

3. Otherwise, I use the closest date to the left of the event (crossing sentence boundaries); if there is no event to the left (i.e. at the beginning of the article), I use the closest event to the right.

Method TYP. + GLOBAL DATES combines this global frequency with the typicality of each event (from method TYPICALITY). It starts by calculating the combined importance $comb(e)$ of typicality and global frequency:

$$comb(e) = typ(e) \cdot glob(y(e)) \tag{6.5}$$

where $comb(e)$ is the combined importance assigned to event e, and $typ(e)$ is the event's typicality as described in Section 6.2.2.1. It then chooses the n events with the highest combined importance $comb(e)$, similar to method TYPICALITY.

As method TYPICALITY, this method does not require information about specific events from the article under consideration to be present in the Web corpus.

6.2.2.3 Syntactic links between an event and a date

The informed methods described here use a syntactic parser to identify events that are syntactically connected to an absolute date. This is in contrast to method SIMPLE DATES (presented in Section 6.1.2), which simply considers whether the event co-occurs with an absolute date in the same sentence.

I base two methods on this idea. The first method (SYNTACTIC DATES) selects randomly from all events that are syntactically linked to a date. The second method (TYP. + SYNTACTIC DATES) is a combination of methods TYPICALITY and SYNTACTIC DATES.

[3]There is no existing system that links all TimeML events to their date of occurrence, unless the event and the date occur in the same sentence.

As in method TYPICALITY, the n most highly ranked events are selected for the timeline. However, only events that are syntactically connected to an absolute date are considered.

Absolute dates are identified using the HeidelTime package, as before; all other date expressions are discarded. Syntactic dependencies between tokens are determined using the C&C dependency parser (version 1.0) (Clark and Curran, 2007). I use them to construct a directed dependency graph for the sentence.

For each TimeML event, I then calculate the minimal distance t in terms of the number of dependency arcs to any token that is found to be part of a date. For instance, the dependency path from "designed" to "1810" in the sentence "With a patent in 1810, Koenig designed a steam press much like a hand press connected to a steam engine" has four arcs. Only events that are connected to a date via a dependency path up to T arcs long are considered to be syntactically linked to a date. A sensible minimum value for T is 2. This is due to constructions such as "was developed before 1398", where the dependency path between date and the event verb is of length 2. Experimental tuning of T on the development resource (described in Section 5.5) confirmed that this minimum of $T = 2$ is indeed optimal.

6.2.2.4 Discouragement of event pairs

A good timeline does not contain redundant events. A simple way of reducing redundancy is to discourage the selection of two or more events from the same year. Many pairs of events that happened in the same year are related to each other, and are therefore likely to be redundant with respect to each other. However, there are also counterexamples where two unrelated important events happen to occur in the same year. For instance, both the inauguration of Winston Churchill and the bombings of British cities took place in 1940; arguably, both these events should be mentioned in a timeline of British history.

I therefore impose a soft constraint which makes it less likely, but not impossible, that two events from the same year are included in the timeline. In order to achieve this, I use an integer linear program. Integer Linear Programming (ILP) is a way of maximising (or minimising) a linear objective function subject to a number of linear inequality constraints. Schrijver (1986) gives one common formulation of the *integer linear programming problem* as: Given a matrix A, and vectors b and c, determine $max\{cx|x \geq 0; Ax = b; x \text{ integral}\}$.

In a strict sense, ILP constraints are always *hard constraints*. For instance, one may specify that the sum of two variables may not exceed 5. Many practical implementations, however, also use *soft constraints*. Soft constraints are part of the program's objective function, whereas hard constraints are not. A soft constraint is a penalty score designed to encourage or discourage certain solutions of the problem. I will use a soft constraint to discourage the selection of events from the same year, and several hard constraints. These ensure, for instance, that the number of events selected does not exceed a given maximum.

I maximise the following objective function using the SCIP solver by Achterberg (2009):

$$\sum_{e_i \in E} x_i \cdot typ(e_i) - \sum_{e_i \in E} \sum_{e_j \in E \setminus e_i} b_{ij} \cdot sameyear(e_i, e_j) \tag{6.6}$$

where i and j are running numbers of TimeML events, E is the set of all TimeML events, each variable x_i indicates whether a corresponding event with number i has been chosen for inclusion in the timeline, b_{ij} is a variable indicating that both events i and j

have been selected for inclusion in the timeline, $typ(e_i)$ is the typicality of event i, and $sameyear(e_i, e_j)$ is an indicator function which returns 1 if events i and j happened in the same calendar year according to the date assignment created using the heuristic described in Section 6.2.2.2, and 0 otherwise. In other words, whenever two selected events e_i and e_j have the same date, the function $sameyear(e_i, e_j)$ applies a penalty to the value of the objective function.

This function is optimised subject to hard constraints which are partly inspired by the well-known multi-document summariser proposed by McDonald (2007).

The first two constraints ensure that the variables x_i (indicating that event i is chosen) and b (indicating that events i and j are chosen) are binary, i.e. their only possible values are 0 and 1:

$$x_i \in \{0, 1\} \quad \forall i \tag{6.7}$$

$$b_{ij} \in \{0, 1\} \quad \forall i, j \tag{6.8}$$

A third constraint ensures that b_{ij} is always 1 if both the variables x_i and x_j are set to 1:

$$x_i + x_j - b_{ij} \leq 1 \quad \forall i, j \tag{6.9}$$

A further set of constraints ensures that the variable b_{ij} can only be 1 if the corresponding variables x_i and x_j for the individual events are 1 also:

$$b_{ij} \leq x_i \quad \forall i, j \tag{6.10}$$

$$b_{ij} \leq x_j \quad \forall i, j \tag{6.11}$$

The final hard constraint guarantees that the total number of selected events is equal to the maximum timeline length L_{max}:

$$\sum_{e_i \in E} x_i = L_{max} \tag{6.12}$$

The method called Same-year penalty combines this criterion with method Syntactic dates, i.e. the integer linear program described here does not consider events that are not syntactically linked to an absolute date.

6.2.2.5 Section balance criterion

The next method (sec. bal.) discourages the selection of too many events from any single part of the document. To this end, I again use an integer linear program, and impose a series of hard constraints in order to ensure that at least one event from each section in the document is selected:

$$\sum_{e_i \in E_S} x_i > 0 \quad \forall S \tag{6.13}$$

where S is a section, and E_S is the set of all events in section S.

System sec. bal. is otherwise identical to System Same-year penalty, except for the objective function, for which I only use the sum of the event's typicalities:

$$\sum_{e_i \in E} x_i \cdot typ(e_i) \tag{6.14}$$

6.2.2.6 Maximising the distance between selected events

Another method assumes that timeline writers try to maximise the temporal distance between any two events in the timeline, while selecting only important events. This is one way of ensuring that the timeline covers the entire timespan described by the article.

It is not practical to use an integer linear program to test this hypothesis, as one would have to create a variable for each pair of events in the text. Such a program would take prohibivitely long to solve. I therefore use a graph-based approach whose nodes reflect events at a given position in the timeline. Here, finding the optimal selection of events involves finding the shortest path of length tl_{max} through the graph. The cost of taking a particular path is determined by the typicality of the selected events and the time distance between two selected events. A path has a low cost if it involves important events that are far apart in time.

The structure of the graph can be visualised as a lattice of nodes with height $|E|$ and width tl_{max} (see Figure 6.3), where $|E|$ is the number of events, and tl_{max} is the permissible number of events in the timeline.

For each event e_i, there are tl_{max} nodes, one for each possible position in the timeline. Node $e_{i,k}$ represents a choice of event e_i in the article as the kth event in the timeline. For each node, I create edges to all logically possible nodes for the next timeline position, namely those that represent an event taking place later in time than the event represented by the current node. For instance, for the node $e_{141,1}$, I create edges to all nodes $e_{i,2}$ for timeline position 2 representing events that took place later in time than event e_{141}. In this way, I ensure that the timeline created advances in time.

The weight of an edge between two nodes n_1 and n_2 represents the cost for adding the two events represented by these nodes to the timeline. This cost C is calculated as

$$C(n_1, n_2) = \frac{1}{typ(e(n_2)) \cdot distyears(e(n_1), e(n_2))} \tag{6.15}$$

where $e(n_i)$ is a function that returns the event represented by node i, $typ(e(n_2))$ is the typicality of event e_2 as defined in Section 6.2.2.1, and $distyears(e(n_1), e(n_2))$ is the absolute time distance (in years) between events $e(n_1)$ and $e(n_2)$. Absolute dates are assigned to events as before.

For finding the shortest path through the graph, I use Dijkstra (1959)'s shortest-path algorithm. In order to facilitate implementation, I add designated start and end nodes e_s and e_e to the graph (not shown in Figure 6.3), which are connected to all nodes $e_{i,1}$ and $e_{i,tl_{max}}$, respectively. The shortest path is then calculated from node e_s to e_e.

My method Maximal Distance combines this criterion with method Syntactic dates: The graph is constructed only using events that are syntactically linked to a date; all other events are disregarded.

6.2.3 Combination of the unsupervised method with the approach by Chasin et al.

A further method (+Chasin) combines two of my unsupervised methods (Typicality and Syntactic dates) with the supervised approach by Chasin et al. (2014). In particular, the typicality of an event is used as a further feature in method Chasin. In addition, events that are not syntactically connected to a date are discarded.

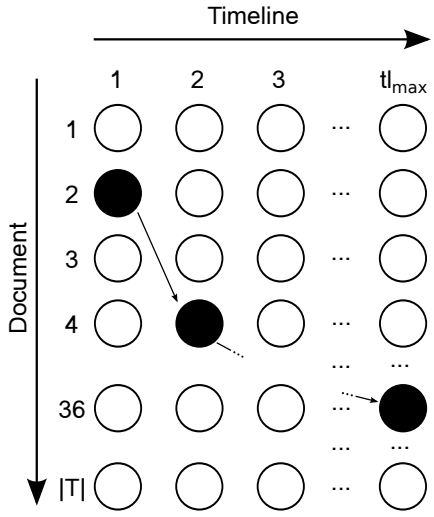


Figure 6.3: Structure of the graph.

6.3 Example output

Before proceeding to the evaluation, I will now show some example output of one of my systems. Figure 6.4 lists a number of sentences in the Wikipedia article "History of Wales" from which the algorithm has selected a TimeML event for the timeline. The selected events are printed in bold font. Selected events which are linked to an HCU in the annotated test resource presented in Chapter 5 (i.e. at least one human timeline writer has expressed these HCUs in their timeline) are printed in red color. The dates of the selected events, which were extracted automatically, are also shown.

Many of the events in red color as well as their context are intuitively typical of the history of geo-political entities, e.g. in the case of "assassinate", "revolted" and "elected". It is reasonable to expect that revolts and elections are more likely to be contained in timelines on countries than in other texts. Such subject-area-specific preferences are modelled by the typicality scores I use.

However, there are of course also events that have not been selected by human annotators although they appear to be typical, such as the passing of a set of laws in 1402. This result is to be expected, since the salience of an event is likely signalled by many different cues in the text. In the example shown, the passing of the Penal Laws in 1402 is presented in the context of a rebellion which is mentioned at the beginning of the sentence. In such a case, the human timeline writer could, for instance, perceive the rebellion itself to be more salient than the passing of a set of laws which is the result of that rebellion. Examples like this one show that it could be worthwhile to investigate discourse effects which let events in the text appear more or less salient than they would be perceived in isolation. This and similar ideas for future work will be discussed in Section 7.3.

One can also see that the example output shown in Figure 6.4 only contains events that co-occur with a date. In fact, each of these events is syntactically connected to its date of occurrence, because this output was generated using a method that removes all other events.

1241	War **broke** out in 1241 and then again in 1245 , and the issue was still in the balance when Dafydd died suddenly at Abergwyngregyn , without leaving an heir in early 1246 .
1244	Llywelyn the Great 's other son , Gruffudd had been killed trying to **escape** from the Tower of London in 1244 .
1301	After passing the Statute of Rhuddlan which restricted Welsh laws , King Edward 's ring of impressive stone castles assisted the domination of Wales , and he crowned his conquest by **giving** the title Prince of Wales to his son and heir in 1301 .
1378	The English government responded to the threat by sending an agent to **assassinate** Owain in Poitou in 1378 .
1400	In 1400 , a Welsh nobleman , Owain Glyndŵr (or Owen Glendower) , **revolted** against King Henry IV of England .
1402	As a response to Glyndŵr 's rebellion , the English parliament **passed** the Penal Laws in 1402 .
1455	In the Wars of the Roses which **began** in 1455 both sides made considerable use of Welsh troops .
1900	The first Labour MP , Keir Hardie , was **elected** as junior member for the Welsh constituency of Merthyr Tydfil and Aberdare in 1900 .

Figure 6.4: Subset of selected TimeML events (in bold font) for "History of Wales".

6.4 Evaluation

I will now evaluate the performance of all methods using the evaluation methodology presented in Chapter 5.

6.4.1 Method

The performance of all 18 methods (of which 9 are uninformed and 9 are informed) is evaluated using pyramid scores averaged across the 10 articles in the annotated test resource presented in Section 5.3. For each article, a pyramid score is calculated using Equation 5.4 (cf. page 100), with $nHCU$ set to the upper limit on the number of timeline entries used for eliciting the respective human-written timelines.

6.4.2 Results and Discussion

In Figure 6.5, the results of uninformed and informed methods are listed separately.

6.4.2.1 Uninformed Methods

I first describe the results of all uninformed methods. A completely random selection of events (Random global) results in a score of 0.10. As expected, selecting events from the beginning (First global) or end (Last global) of the article does not result in a clear improvement (0.12 and 0.09 respectively). However, selecting events from the beginning of each section in turn (First RR) leads to a much higher score (0.21) compared to methods that do not take section structure into account.

Constructing a timeline using the final events in each section (Last RR) has no effect (0.10). It makes no difference to the result whether the events at the beginning or end of a section are selected one at a time (First RR and Last RR) or in blocks, sentence

	Method	Description	Result
UNINFORMED METHODS	RANDOM GLOBAL	Random selection of events in the article	0.10
	LAST GLOBAL	Last events in the article	0.09
	FIRST GLOBAL	First events in the article	0.12
	FIRST RR	First events per section (round-robin)	0.21
	LAST RR	Last events per section (round-robin)	0.10
	RANDOM RR	Random selection per section (round-robin)	0.12
	FIRST SEN RR	All events in first sentences per section (round-robin)	0.19
	LAST SEN RR	All events in final sentences per section (round-robin)	0.10
	SIMPLE DATES	Random selection from events with a date in the same sentence	0.16
INFORMED METHODS	SYNTACTIC DATES	Random selection from events syntactically connected to a date	0.25
	CHASIN	Modified version of supervised method by Chasin et al. (2014)	0.12
	TYPICALITY	events with highest typicality	0.17
	TYP. + GLOBAL DATES	Method TYPICALITY combined with information about globally important dates	0.14
	TYP. + SYNTACTIC DATES	Method TYPICALITY with events that are not syntactically connected to a date removed	0.29
	SAME-YEAR PENALTY	ILP-based method penalising pairs of events from the same year	0.30
	SEC. BAL.	ILP-based method ensuring that content from each section is selected	0.29
	MAXIMAL DISTANCE	Graph-based method maximising typicality and time distance between any two events	0.29
	+CHASIN	Combination of Chasin et al.'s supervised approach with unsupervised features	0.28

Figure 6.5: Average pyramid scores for all methods on the test set.

by sentence (FIRST SEN RR and LAST SEN RR, the scores of which are 0.19 and 0.10 respectively).

These results suggest that in history articles, important events tend to be placed at the beginning of each section, not at the end. For timeline generation, methods FIRST RR and FIRST SEN RR seem to be the equivalent of the first-n-words approach for news summarisation, which has been shown to result in superior performance. But the difference lies in the fact that for timelines, it is not the beginning of the text where all important events are located. Instead, the nature of a timeline implies that all parts of the input article must be covered. A simple round-robin selection seems to achieve this to a satisfactory degree.

However, this class of methods might not work as well with articles for which my selection criteria do not hold. Remember that all articles in the evaluation resource are chronologically structured. While for such articles it is likely that each section contains at least some important event, this cannot be assumed, for instance, for articles whose document structure is different.

Let us now consider the last method in this section, SIMPLE DATES. This method is distinct from the other uninformed methods in that it uses not document structure, but

the presence of dates. Here, an improvement (0.16) over absolute random choice (0.10) occurs. The pair of results from methods SIMPLE DATES and SYNTACTIC DATES show that dates are a very successful cue for the presence of a salient event. SIMPLE DATES at 0.16 is better than its respective uninformed baseline despite only applying a very crude method of finding dates. The implementation effort involved in identifying dates in a more sophisticated way using a dependency parse is worth the while, as the improved result of 0.25 shows. I therefore chose this method as the baseline of informed methods, and it proves surprisingly high.

6.4.2.2 Informed Methods

I now turn to describing the results obtained with the remainder of the informed methods.

Chasin et al.'s (2014) approach does not do any better than the uninformed methods and performs much worse than SYNTACTIC DATES. This is a vindication of my hypothesis that Chasin et al.'s method is too domain-specific for my corpus. Their method is designed to work on articles describing individual battles and wars, at which it performs well (Chasin et al., 2014). The low performance of this method can also not be explained by the pure size of the training data used, as this is larger than what they originally used. Of course, this method might well perform a lot better with a corpus orders of magnitude larger, and we also know that the development corpus is of lower quality and was created differently from the test corpus. But nevertheless, the large gap between this system's performance and the baseline for informed methods (0.12 vs. 0.25) is surprising.

The score of the method that selects the events with the highest typicality alone (TYPICALITY) is lower at 0.17 than that of method SYNTACTIC DATES. In a way, this is disappointing because the information contained in typicality contains semantic trends about the nature of events, whereas dates can be harvested only by using a syntactic parser. I must conclude that timeline writers naturally use explicit dates in the source text as a signal of importance which is easibly detectable in a superficial manner. The semantic trends of typicality of events may require more data or they are a weaker signal when used in isolation. What happens when we combine the two signals? With method TYP. + SYNTACTIC DATES, which excludes events without an attached date but is otherwise identical to method TYPICALITY, the result improves (0.29), although it is still broadly in the same range of pyramid scores.

The highest result is observed using method SAME-YEAR-PENALTY, which also includes typicality and syntactic dates. The method combines semantic trends via typicality with one method of respecting inter-relationships between events and also profits from the presence of syntactically linked dates. Note that it is better than method FIRST RR although it does not depend on document structure. It will work equally well whether a text from the Internet is organised in sections or not, whereas FIRST RR entirely relies on section structure. SAME-YEAR PENALTY is thus the more general method.

Given that SAME-YEAR PENALTY does not use section structure, one could ask whether it could profit from having additional access to this information. Unfortunately, I did not know which features would work best in combination, so I did not test the combination of the method SAME-YEAR PENALTY with section information. However, I have results of a comparable method, SEC. BAL., in which the constraints about pairs of events are replaced by the section balance criterion. This method performs identically to TYPICALITY + SYNTACTIC DATES. This is somewhat surprising given that a section criterion is useful with uninformed methods such as FIRST RR. It seems that the improvement of method FIRST RR over method RANDOM GLOBAL has less to do with the balanced selection of

material from all article sections, and instead with the location of that material at the beginning of each section.

What about the other methods encoding constraints between events? It turns out that they do not result in further performance improvement compared to the methods presented thus far.

In particular, information about globally important dates (method TYP. + GLOBAL DATES) is not beneficial at all (0.14). Note that this is the case although all articles in the test resource discuss rather general topics such as the entire history of a country or field of science. One could expect that years such as 1945 are relevant in many history articles discussing the history of a country or field of science. The poor result obtained, however, suggest that the opposite is the case. This may indicate that mentions of globally important events are often mentioned in passing in an article, which makes global dates a feature not specific enough for a particular article. Although it would be possible to explore different ways of using background information, I see this as another argument in favour of using only or mostly local cues in the single article at hand in order to identify important events.

Another approach that does not lead to an improvement is method MAXIMAL DISTANCE. This result suggests that the expectation that a good timeline should cover the entire time period considered by the article does not necessarily result in a selection of equidistant (in terms of time) events. Our timeline writers seemed to be accepting of larger gaps despite the fact that the guidelines explicitly asked them to balance coverage with salience. This can potentially result in larger gaps between events in the timeline, in particular if a high number of important events happen in a short period of time. For instance, timeline authors might find it acceptable to skip earlier time periods in German history in order to be able to add multiple events about the Nazi era and Second World War to the timeline.

The final system evaluated here (+CHASIN) considers whether Chasin et al.'s method (CHASIN) can be improved by combining it with TYPICALITY and SYNTACTIC DATES, which is not the case (0.28). Given that their supervised approach performed poorly on its own, this result is not surprising. However, it is not unthinkable that better ways of combining Chasin et al.'s method with my unsupervised methods exist, but I have not explored them here.

6.4.2.3 Final remarks

In this evaluation, I have compared many unsupervised methods with an existing supervised approach, the only one that existed when I started working on this topic. This existing approach was the one that is most similar in remit to the objectives of this thesis, as it also frames timeline generation as a selection of events from a single article. My methods beat this supervised approach on the test resource presented in Chapter 5.

However, as it turns out, the task that Chasin et al. set themselves and my task are not so similar. My method is independent of domain, the time period discussed, and background knowledge about individual important events. In contrast, Chasin et al. restrict themselves to articles which discuss a single battle or war. This difference is important to me, because I set out to create a general system. In such an environment, totally unsupervised approaches often work better, and my achievement is to compare a wide range of such unsupervised methods. More sophisticated supervised methods may well fare a lot better. However, my guess is that they will require substantially more training data than is provided here.

Chasin et al. have shown that with a relatively small training corpus, supervised methods can do well for domain-specific treatments of one type of events (battles and wars). The task approached here is much more ambitious. In particular, training a supervised system on less domain-specific history articles is much harder, as this would require the ability to generalise across domains. More sophisticated supervised methods, better machine learning, intermediate processing or any other better algorithms could in principle be developed. It is also possible that a training corpus one or two orders of magnitude larger would help. However, what I have shown here is that my task is unlikely to be solvable with relatively simple features such as Chasin et al.'s and a small training corpus. This is the case even though I have treated Chasin et al.'s system fairly by giving it access to at least comparable training data size and even better event extraction (TIPSem-B) than originally used (Evita). In addition, my implementation of Chasin et al.'s system had access to introduction sections for extracting keywords, unlike all other methods considered.

6.5 Qualitative analysis

Finally, I analyse the created system output qualitatively, in order to give the reader an impression of the quality of timelines at different recall levels. Figures 6.6 (see page 133), 6.7 (see page 134) and 6.8 (see page 135) show extracts of three timelines for the article "History of Connecticut" produced by different systems. In particular, Figure 6.6 shows part of a timeline produced using method LAST RR, which achieved a pyramid score of 0.14; Figure 6.7 gives entries of a timeline created using method FIRST RR, achieving a pyramid score of 0.26; and Figure 6.8 shows part of a timeline produced using method TYP. + SYNTACTIC DATES, which obtained a pyramid score of 0.37. For each timeline, the first 10 entries in chronological order are listed. The leftmost column shows the reward given to the respective algorithm for selecting a single event, as calculated using Equation 5.7 (cf. page 101).

Figures 6.6 and 6.7 make it obvious that methods which select events purely based on their position in the document (e.g. from the beginning or the end of a section) have two main drawbacks. First, it is very likely that events which took place in the same year are co-selected, which is clearly undesirable. For instance, the timeline in Figure 6.7 contains four events from the same year (1650). In many cases, the co-selected TimeML events in fact refer to the same real-world event, such as with the two events taking place in 1689 shown in Figure 6.6. A second disadvantage is that TimeML events which do not express real-world events according to the definition in Section 5.1.1 (such as "according" and "stood" in Figure 6.7) are arguably more likely to be selected by uninformed methods (Figures 6.6 and 6.7) than by informed methods (Figure 6.8). This is due to the fact that uninformed methods do not consider the presence of dates or the typicality of an event and its context with respect to the article's subject area. On the other hand, a method such as FIRST RR is apparently more likely to select sentences such as the first one in Figure 6.7, which contain a summary statement of the ensuing historical era and are often considered important.

Although the extract of a timeline produced by method TYP. + SYNTACTIC DATES (shown in Figure 6.8) also contains a case where two closely related TimeML events have been co-selected (see the first two entries shown), the fact that events are selected from across the text reduces the risk of this happening considerably. Important events now have to be identified on the basis of cues such as the presence of a syntactically attached

Cont.	Date	Timeline entry
	1687	The Connecticut court **met** and voted on May 9 , 1689 to restore the old charter .
	1687	When word arrived that the Glorious Revolution had placed William and Mary on the throne , the citizens of Boston arrested Andros and **sent** him back to England in chains .
	1689	The Connecticut court met and **voted** on May 9 , 1689 to restore the old charter .
2.0	1689	The Connecticut court met and voted on May 9 , 1689 to **restore** the old charter .
1.0	1698	They also **reelected** Robert Treat as governor each year until 1698 .
	1781	The French General the Comte de Rochambeau celebrated the first Catholic Mass in Connecticut at Lebanon in summer 1781 while marching through the state from Rhode Island to **rendezvous** with General George Washington in Dobbs Ferry , New York .
	1781	The French General the Comte de Rochambeau celebrated the first Catholic Mass in Connecticut at Lebanon in summer 1781 while **marching** through the state from Rhode Island to rendezvous with General George Washington in Dobbs Ferry , New York .
1.0	1781	The French General the Comte de Rochambeau **celebrated** the first Catholic Mass in Connecticut at Lebanon in summer 1781 while marching through the state from Rhode Island to rendezvous with General George Washington in Dobbs Ferry , New York .
	1781	New London and Groton Heights were **raided** in September 1781 by Connecticut native and turncoat Benedict Arnold .
	1792	An area 25 miles -LRB- 40 km -RRB- wide at the western end of the Western Reserve , set aside by Connecticut in 1792 to compensate those from Danbury , New Haven , Fairfield , Norwalk , and New London who had suffered heavy losses when they were burnt out by fires **set** by British raids during the War of Independence , became known as the Firelands .

Figure 6.6: First 10 entries of a timeline for the article *History of Connecticut* produced using method LAST RR (achieving a pyramid score of 0.14).

date or an event whose context is typical of the article's subject area. One example visible in Figure 6.8 is the event "negotiated" in year 1683, which was not selected by the two uninformed models that created the timelines in Figures 6.6 and 6.7.

To conclude, in some cases even a small collection of timeline entries provide obvious visual cues as to why one timeline obtains a higher pyramid score than another. This is not generally the case, however. Due to the high subjectivity of content selection tasks, an independent experimenter trying to analyse the strengths and weaknesses of a given algorithm will not necessarily share the timeline writers' intuitions. A reduction of this inherent subjectivity can arguably only be achieved by increasing the number of timeline writers used to construct the gold standard.

6.6 Chapter summary

In this chapter, I investigated different methods for the content selection stage of timeline generation. The approaches were evaluated using the evaluation methodology described

Cont.	Date	Timeline entry
2.0	1614	Various Algonquian tribes **inhabited** the area prior to European settlement .
	1614	In 1614 Adriaen Block **explored** the coast of Long Island Sound , and sailed up the Connecticut River at least as far as the confluence of the Park River , site of modern Hartford , Connecticut .
	1614	In 1614 Adriaen Block explored the coast of Long Island Sound , and **sailed** up the Connecticut River at least as far as the confluence of the Park River , site of modern Hartford , Connecticut .
	1623	By 1623 , the new Dutch West India Company regularly traded for furs there and ten years later they **fortified** it for protection from the Pequot Indians as well as from the expanding English colonies .
1.0	1633	By 1623 , the new Dutch West India Company regularly **traded** for furs there and ten years later they fortified it for protection from the Pequot Indians as well as from the expanding English colonies .
	1650	**According** to the 1650 Treaty of Hartford with the Dutch , the western boundary of Connecticut ran north from the west side of Greenwich Bay " provided the said line come not within 10 miles -LRB- 16 km -RRB- of Hudson River . "
	1650	According to the 1650 Treaty of Hartford with the Dutch , the western boundary of Connecticut **ran** north from the west side of Greenwich Bay " provided the said line come not within 10 miles -LRB- 16 km -RRB- of Hudson River . "
	1650	According to the 1650 Treaty of Hartford with the Dutch , the western boundary of Connecticut ran north from the west side of Greenwich Bay " **provided** the said line come not within 10 miles -LRB- 16 km -RRB- of Hudson River . "
	1650	According to the 1650 Treaty of Hartford with the Dutch , the western boundary of Connecticut ran north from the west side of Greenwich Bay " provided the said line **come** not within 10 miles -LRB- 16 km -RRB- of Hudson River . "
	1662	Up until this time , Connecticut had adhered to the 1662 Charter , and with the independence of the American colonies over forty years prior , much of what the Charter **stood** for was no longer relevant .

Figure 6.7: First 10 entries of a timeline for the article *History of Connecticut* produced using method FIRST RR (achieving a pyramid score of 0.26).

in Chapter 5.

I first introduced a number of uninformed methods, which do not rely on external background information. I then presented a range of informed methods, in particular an improved version of a supervised system from the literature, and various novel unsupervised methods.

The results show that an unsupervised method that considers subject-area-specific information, syntactic links and the presence of events from the same year achieves the best result, outperforming a method that is restricted to syntactic links, other unsupervised methods, and a supervised approach from the literature that has access to an annotated training corpus of 30 articles.

Another result was that methods that select events from the beginning of each section ("First RR") perform better than a method that performs a random selection. This

Cont.	Date	Timeline entry
1.5	1662	On April 22 , 1662 , the Connecticut Colony **succeeded** in gaining a Royal Charter that embodied and confirmed the self-government that they had created with the Fundamental Orders .
	1662	On April 22 , 1662 , the Connecticut Colony succeeded in gaining a Royal Charter that embodied and **confirmed** the self-government that they had created with the Fundamental Orders .
2.0	1683	Finally , on November 28 , 1683 , the states **negotiated** a new agreement establishing the border as 20 miles -LRB- 32 km -RRB- east of the Hudson River , north to Massachusetts .
1.0	1789	Connecticut 's government continued unchanged even after the revolution , until the United States Constitution was **adopted** in 1789 .
	1795	Connecticut owned this territory until **selling** it to the Connecticut Land Company in 1795 for $ 1,200,000 , which resold parcels of land to settlers .
	1796	In 1796 , the first settlers , led by Moses Cleaveland , **began** a community which was to become Cleveland , Ohio ; in a short time , the area became known as " New Connecticut " .
	1796	In 1796 , the first settlers , led by Moses Cleaveland , began a community which was to become Cleveland , Ohio ; in a short time , the area **became** known as " New Connecticut " .
	1806	In 1806 , the state leadership **sent** town leaders instructions for the forthcoming elections .
1.0	1817	The failure of the Hartford Convention in 1814 wounded the Federalists , who were finally **upended** by the Republicans in 1817 .
1.0	1818	In 1818 , a new constitution was adopted that was the first piece of written legislation to separate church and state in Connecticut , and **give** equality all religions .

Figure 6.8: First 10 entries of a timeline for the article *History of Connecticut* produced using method TYP. + SYNTACTIC DATES (achieving a pyramid score of 0.37).

suggests that exploiting structural information about the input article can – where it is available – improve the performance of a timeline generation algorithm.

Chapter 7

Conclusion

7.1 Thesis overview

In this thesis, I have introduced the new research task of generating a timeline from a single history article. I assume that the process of creating a timeline involves selecting a subset of historical events from the source article. Due to the novelty of this task, evaluation frameworks were not readily available. A substantial part of this thesis has therefore discussed the design of suitable evaluation methodologies and the construction of adequate evaluation corpora. These resources were then used to evaluate a wide range of timeline generation algorithms.

The creation of timelines from textual resources has been an active research topic for about 15 years. In Chapter 2, I reviewed existing work. The term "timeline generation" has been used to describe a number of very different tasks, e.g. the identification of *all* events in a given text and the subsequent creation of a temporal ordering, or the identification of emerging topics in a stream of news stories.

Many existing works are not applicable to my task as they are in the tradition of multi-document summarisation. They assume the presence of a large electronic corpus of documents (mostly news articles) with annotated document creation times. As such information is only available for the most recent periods in history, existing works in this field have been evaluated on collections of news articles covering, for instance, the 1990s or the 2000s. One could even go so far as renaming this task "news timeline generation". Methods that assume the presence of a parallel corpus cannot be applied to my task, as my remit includes texts that describe events in the non-recent past, either exclusively or in combination with events from the recent past.

In my setting, content mentioned in a single input document has to be scored using mostly local cues. This task, however, is arguably harder than judging importance based on overlap between multiple documents. As an example of such an approach, I reviewed the work by Chasin et al. (2014), which classifies TimeML events in a single history article into important and unimportant events. The approach is evaluated using human judgments on the importance of individual TimeML events. Such an evaluation is of limited usefulness in practice, as it is difficult for a human annotator to decide whether an individual event word is important or not. This problem arises because TimeML events do not provide a clear one-to-one mapping to real-world events.

In Chapter 3, I described an initial experiment in the context of timeline generation. I introduced a supervised approach for the problem of identifying important historical figures that should be contained in a timeline on a given topic. The classifier uses both local

features from the entity's surroundings and semantic features about the entity's importance. My experiments show that this machine-learning approach results in performance superior to that of a frequency baseline, which is competitive in this setting. The results were achieved using a corpus specifically constructed by myself using existing timelines harvested from the web and history articles on the same topics, as well as a heuristic for aligning different mentions of the same person. I also discussed the limits of my approach, and explained my decision not to use the aforementioned corpus for the task of timeline generation, which was investigated in the remainder of the thesis.

The approach to timeline generation which I described in Chapters 4 to 6 is based on an interpretation of the task as an instance of single-document summarisation. In search of an appropriate evaluation methodology, I reviewed existing approaches to summarisation evaluation (Chapter 4). Evaluation methodologies based on semantic units, such as the pyramid method by Nenkova et al. (2007), assess whether a system summary expresses the same content as a number of gold-standard summaries written by humans. This approach has one crucial advantage over the alternatives: It is able to abstract away from the wording used to express the content. However, this robustness to reformulation comes at a cost: The need for human annotation of system summaries makes such methods prohibitively expensive.

The only alternative is surface-oriented evaluation, which can be performed automatically once a number of system summaries have been collected. But such methods, which rely on string overlap, cannot guarantee that a summary indeed expresses the semantic content that most human judges perceived as important.

Based on this comparison, I outlined the requirements for an evaluation methodology for the task of timeline generation which strikes a balance between these two families of approaches. The methodology should be based on semantic meaning units, yet ideally require no human annotation of system summaries whatsoever. In addition, the creation of semantic units from the gold-standard summaries should be reproducible.

Note that these considerations are valid for summarisation tasks more widely. Evaluation methodologies which make it possible to use semantically-oriented, "deep" evaluation in day-to-day algorithm development (and still not require boundless human annotation) are a key prerequisite for making significant advances in summarisation research in the future.

Chapter 5 presents a novel evaluation methodology which fulfills the requirements outlined in Chapter 4. I first introduced the concept of Historical Content Units (HCU) and described the event model used. I then explained the main principles of the proposed evaluation methodology. Next, I described how HCUs are created on the basis of the human-written timelines. A large part of the chapter was concerned with the identification of textual anchors for HCUs in the source text, in a move to automate the HCU detection. To this end, I introduced anchor weights, which are a way of quantifying the extent to which a TimeML event represents the semantic content of an HCU. I described how a human expert assigns such anchor weights, starting with general principles, and presented a detailed set of guidelines describing how HCUs and TimeML events are related in the many grey-area cases one encounters in practice. I then showed how an existing evaluation resource can be used to score system timelines.

Next, I described the creation of an evaluation corpus consisting of gold-standard timelines for a set of history articles. I explained how the source articles were chosen and pre-processed, and gave statistics for the resulting corpus. I also presented a second corpus consisting of annotated history articles. This corpus, which was created in a

drastically simplified manner in order to reduce the human effort required, can be used for development and tuning of new timeline generation algorithms.

I then showed that my way of creating HCUs results in the desired pyramid shape and that the concept of HCU weights is meaningful. I also gave statistics for the annotation of anchor weights performed, and analysed whether a consistent annotation can be achieved using my set of guidelines.

In Chapter 6, I compared the performance of various methods for timeline generation. I first described a number of uninformed methods which do not rely on external knowledge or linguistic processing, instead making use of section structure and the presence of dates. I then discussed a number of informed methods, starting with an improved version of Chasin et al.'s supervised approach that was trained on the additional development corpus described in Chapter 5. A second group of informed methods are unsupervised. Events are selected on the basis of an article's subject area or syntactic links between an event and a date. I also experimented with date-based co-selection constraints and important dates harvested from Wikipedia. Finally, all methods were evaluated using the evaluation methodology described in Chapter 5. The numerically best-performing method was a combination of three methods which consider the article's subject area and syntactic dates, while discouraging the co-selection of events from the same year. This is slightly surprising as all three methods are unsupervised.

7.2 Contributions

The main contributions of the thesis can be summarised as follows:

- I proposed an evaluation methodology for timeline generation that allows for evaluation of timelines using deep meaning units, which I call Historical Content Units (HCU). Creating HCUs requires multiple human-written timelines and extensive human annotation for each text. However, thanks to the anchoring of HCUs in the source text, no human annotation of system summaries is required. (Chapter 5)

- I investigated the performance of various informed and uninformed approaches to timeline generation. Informed approaches included supervised as well as unsupervised methods. I found that a combination of three unsupervised methods outperforms both other uninformed methods and a supervised approach. (Chapter 6)

- I showed that a supervised approach which uses structural, linguistic and semantic features outperforms a strong frequency baseline on the task of identifying important historical figures that should be mentioned in a timeline on a given topic. (Chapter 3)

- I created the following evaluation resources:

  - a manually annotated corpus (consisting of human-written timelines, HCUs and anchor weights for pairs of HCUs and TimeML events) that can be used to evaluate timeline generation algorithms according to the methodology presented in Chapter 5;

  - a separate development corpus (being of lower quality but comprising a larger number of articles than in the test corpus) that can be used for tuning timeline generation algorithms;

– a corpus of several hundred gold-standard timelines and accompanying history articles that can be used to train and evaluate systems which select important entities for timelines.

7.3 Directions for further research

Based on the insights obtained through my experimental work, I will now outline possible avenues for extending the methods presented in this thesis.

Surface realisation of important historical events. This thesis has not investigated surface realisation of timeline entries, but for any deployable system, one would need a way to present the content selected for a timeline entry.

For this purpose, one could either use natural language generation (NLG) techniques that create a new sentence from scratch, or manipulate the original surface text in suitable ways. Constructing timeline sentences directly from original article text is likely to involve operations such as sentence compression or sentence fusion. Often information from previous sentences has to be added in order to make a timeline sentence understandable. For instance, where a pronoun refers to an entity earlier on in the text, it must be resolved. The original sentence may also contain superfluous subordinate clauses that should be removed.

There are cases where the original source text cannot be transformed into a timeline entry. Here, entirely new timeline sentences need to be generated. This task would likely require deeper understanding of the event's meaning. For instance, events could be mapped to a matching frame in FrameNet (Baker et al., 1998), which would then be used as the starting point for creating a short timeline sentence.

Before research in this area can be performed, my evaluation methodology for timeline generation would have to be extended by a comprehensive strategy for measuring the quality of surface realisation. The resource I created already contains the textual timeline entries needed for this task. However, it is not known whether humans agree on how a given historical event should be verbalised in a timeline. It would therefore make sense to collect additional human-written timeline entries for a set of given historical events. The annotation effort for this corpus construction would be considerably lower than for the task of content selection, as human annotators need not read and process an entire history article before writing timeline entries.

Evaluation of other single-document summarisation tasks. My new evaluation methodology could, with suitable modifications, be used with other single-document summarisation tasks also. Its central advantage is that deep evaluation is possible without costly annotation of each and every system timeline. Being able to test the performance of new algorithms rapidly and reliably would be desirable in summarisation evaluation more generally. In particular, an approach that can easily evaluate multiple system summaries would, for the first time, allow for the automatic evaluation of a large number of single-document summarisers using an evaluation method based on semantic units.

My evaluation methodology would have to be adapted in a suitable way before it could be used with other single-document summarisation tasks. A main difference between my task and general single-document summarisation is that timeline entries always describe

events. This led to the definition of HCUs as meaning units. With other forms of single-document summarisation, these units would have to be defined differently. For instance, one could identify lexical cues in the gold standard summary which signal that the summary writer intended to start a new meaning unit. The guidelines for linking meaning units to anchors in the source text would also have to be redesigned in this case.

Topical changes in the article text. The methods for timeline generation presented in this thesis could be extended. For instance, it is likely that discourse phenomena in the source text play an important role when timeline writers decide on the events that should be added to the timeline.

One option is to use methods that detect where in the article text a topic shift occurs, for instance lexical chains (Barzilay and Elhadad, 1997) or topic models such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003). The article would first be segmented into text blocks discussing a single topic each. These text blocks could be used as an additional feature in any supervised or unsupervised timeline generation algorithm. It seems worthwhile to investigate this hypothesis since there is not necessarily a one-to-one correspondence between section and paragraph breaks and the beginning of a new topic.

Exploiting other discourse phenomena. Other discourse phenomena could be explored as well. For example, it is unlikely for a side event that is merely presented as evidence of another event to be mentioned in the timeline. In order to identify instances of such relations, one might use an existing discourse parser, for instance one which is based on Rhetorical Structure Theory (RST). The output of such a parser could be used to define co-selection constraints such as the ones I used in Chapter 6.

Timeline generation with regard to a user query. Further work may also address the related problem of timeline generation with regard to a user query. Timeline generation in this thesis refers to the creation of *generic* timelines, which are not tailored to any particular information need the user may have. However, some users may be more interested in learning about a country's economic history than about scientific advances, for instance. An algorithm that is able to take such preferences into account has to balance two potentially conflicting objectives: creating a timeline that is of high quality in its own right (which requires identifying salient events and covering the entire time period discussed in the article), and ensuring that events added to the timeline are relevant to the query. One way of operationalising query relevance would be to use query expansion techniques from information retrieval in order to assess for each sentence in the article to which degree its vocabulary is related to the query topic. This query relevance score could then be combined with a query-independent typicality score akin to the one used in my experiments.

Methods for timeline generation using texts from other genres. It would be useful to study the performance of different timeline generation algorithms with input texts other than history articles in an encyclopedia. With history books, textbooks or historical texts, it may be necessary to use different or adapted timeline generation algorithms, since texts from these genres could present historical information in a different way compared to history articles. For instance, textbooks often aim to interpret and explain historical events instead of simply enumerating them. Identifying the actual historical events could be a harder task with such texts than with relatively concise Wikipedia articles.

Recent events vs. events in the distant past. One could also investigate whether the best of way of constructing a timeline depends on how distant in the past the historical events in the input text are. For instance, it seems likely that a revolt that took place in 2013 is perceived as important by many human readers, while a similar uprising in the more distant past is omitted from the timeline. One possible reason is that humans find events that are similar to events which they have experienced personally more important. Timeline writers are also likely to have more background knowledge about history in the 20th and 21st century than about earlier periods of history. Knowledge about such preferences could, for instance, be used to fine-tune the algorithms presented in this thesis depending on the time period considered.

More sophisticated supervised methods. An obvious strategy for improving the performance of all supervised methods presented in this thesis is to use more powerful machine learning methods. For instance, it seems likely that neural networks, which have been shown to outperform other methods in many natural language processing tasks, could be used to learn a more robust model of what is an important entity or event that should be added to a timeline. The higher complexity of neural networks, however, means that training is not possible without a substantial amount of labelled data. To perform timeline generation using neural networks, an evaluation resource several orders of magnitude larger than the one presented in Chapter 5 would have to be constructed upfront.

An alternative would be to use active learning methods, e.g. for improving the performance of models based on support vector machines (Tong and Koller, 2002). Such approaches can help reduce the effort required for labelling training instances.

Identifying important named entities of other types. My preliminary experiment in Chapter 3 started from the assumption that a good timeline can be constructed by identifying important historical figures in a source article. A possible extension is to investigate which named entities of other types (such as locations or organisations) should be added to the timeline. An analysis of this kind would require the construction of a mapping between such entities in an article and co-referring entities in a gold-standard timeline (similar to the mapping for person names presented in Chapter 3).

Construction of textual timelines based on important historical figures. Further research should also address the construction of a full textual timeline based on a set of salient historical figures. While knowledge about important historical figures is useful for constructing a timeline, a concatenation of all sentences that mention an important person is unlikely to result in a good timeline. In particular, problems arise because in many cases, only a small number of the mentions of an important historical figure contain salient content that humans would add to the corresponding timeline. Even clearly important persons such as Winston Churchill may be mentioned in the context of unimportant side events. My supervised method therefore assigns a separate importance score to each mention of a historical figure. I do not yet use this score for the next step, the creation of timeline entries. To perform this step, one would have to investigate which mentions of an important figure should be used as the basis for constructing timeline entries. Ideally, those mentions that have been assigned the highest importance scores by my algorithm work best.

7.4 Outlook

It is my hope that timeline generation for single history articles will continue to be an attractive task. There is an increasing public interest in acquiring a better understanding of the recent as well as the distant past. Timelines are ideal for providing this knowledge, due to their visual and conceptual simplicity. Possible applications range from the use in educational institutions to interactive history browsers on the web.

This thesis has defined single-document timeline generation as a research task for the first time. I have here introduced a range of supervised and unsupervised approaches to timeline generation, but this is only the beginning of a new line of investigation. The evaluation methodology I presented will hopefully allow the research community to evaluate new algorithms in a more meaningful way than hitherto possible, and at a considerably reduced annotation cost.

Bibliography

Achterberg, T. (2009). SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41. `http://mpc.zib.de/index.php/MPC/article/view/4`.

Allan, J., editor (2002). *Topic Detection and Tracking: Event-based Information Organization*. Kluwer Academic Publishers, Norwell, MA, USA.

Allan, J., Carbonell, J., Doddington, G., Yamron, J., and Yang, Y. (1998). Topic Detection and Tracking Pilot Study: Final Report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, Lansdowne, VA, USA.

Allan, J., Gupta, R., and Khandelwal, V. (2001). Temporal Summaries of New Topics. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 10–18, New York, NY, USA. ACM.

Bach, E. (1981). On time, tense, and aspect: An essay in English metaphysics. *Radical Pragmatics*, pages 63–81.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet Project. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, pages 86–90, Montreal, Québec, Canada. Association for Computational Linguistics.

Barzilay, R. and Elhadad, M. (1997). Using Lexical Chains for Text Summarization. In *Proceedings of the ACL/EACL 1997 Workshop on Intelligent Scalable Text Summarization*, pages 10–17, Madrid, Spain.

Barzilay, R. and Lapata, M. (2008). Modeling Local Coherence: An Entity-based Approach. *Comput. Linguist.*, 34(1):1–34.

Bauer, S., Clark, S., and Graepel, T. (2014). Learning to Identify Historical Figures for Timeline Creation from Wikipedia Articles. In Aiello, L. M. and McFarland, D. A., editors, *Social Informatics - SocInfo 2014 International Workshops, Barcelona, Spain, November 11, 2014, Revised Selected Papers*, volume 8852 of *Lecture Notes in Computer Science*, pages 234–243. Springer.

Bauer, S. and Teufel, S. (2015). A Methodology for Evaluating Timeline Generation Algorithms based on Deep Semantic Units. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 834–839. The Association for Computer Linguistics.

Bauer, S. and Teufel, S. (2016). Unsupervised Timeline Generation for Wikipedia History Articles. In Su, J., Carreras, X., and Duh, K., editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2343–2349. The Association for Computational Linguistics.

Belz, A., Kow, E., and Viethen, J. (2009). The GREC Named Entity Generation Challenge 2009: Overview and Evaluation Results. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation*, UCNLG+Sum '09, pages 88–98, Suntec, Singapore. Association for Computational Linguistics.

Bethard, S. (2013). ClearTK-TimeML: A minimalist approach to TempEval 2013. In *Second Joint Conference on Lexical and Computational Semantics ('*'SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 10–14, Atlanta, Georgia, USA. Association for Computational Linguistics.

Bethard, S., Kolomiyets, O., and Moens, M. (2012). Annotating Story Timelines as Temporal Dependency Structures. In *LREC*, pages 2721–2726. European Language Resources Association (ELRA).

Bethard, S., Martin, J. H., and Klingenstein, S. (2007). Finding Temporal Structure in Text: Machine Learning of Syntactic Temporal Relations. *Int. J. Semantic Computing*, 1(4):441–457.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Brin, S. and Page, L. (1998). The Anatomy of a Large-scale Hypertextual Web Search Engine. In *Proceedings of the Seventh International Conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands. Elsevier Science Publishers B. V.

Carbonell, J. and Goldstein, J. (1998). The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336, New York, NY, USA. ACM.

Carletta, J. (1996). Assessing Agreement on Classification Tasks: The Kappa Statistic. *Comput. Linguist.*, 22(2):249–254.

Carlson, L., Conroy, J. M., Marcu, D., O'Leary, D. P., Okurowski, M. E., Taylor, A., and Wong, W. (2001). An Empirical Study of the Relation between Abstracts, Extracts, and the Discourse Structure of Text. In *Proceedings of the DUC-2001 Workshop on Text Summarization*, New Orleans, Louisiana, USA.

Carmel, D., Chang, M., Gabrilovich, E., Hsu, B. P., and Wang, K., editors (2014). *ERD'14, Proceedings of the First ACM International Workshop on Entity Recognition & Disambiguation, July 11, 2014, Gold Coast, Queensland, Australia*. ACM.

Carnie, A. (2002). *Syntax: A Generative Introduction*. Wiley Desktop Editions. Wiley.

Caselli, T., Fokkens, A., Morante, R., and Vossen, P. (2015). SPINOZA_VU: An NLP Pipeline for Cross Document TimeLines. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 787–791, Denver, Colorado, USA. Association for Computational Linguistics.

Caselli, T., Lenzi, V. B., Sprugnoli, R., Pianta, E., and Prodanof, I. (2011). Annotating Events, Temporal Expressions and Relations in Italian: The It-TimeML Experience for the Ita-TimeBank. In *Proceedings of the 5th Linguistic Annotation Workshop*, LAW V '11, pages 143–151, Portland, Oregon, USA. Association for Computational Linguistics.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chasin, R., Woodward, D., Witmer, J., and Kalita, J. (2014). Extracting and Displaying Temporal and Geospatial Entities from Articles on Historical Events. *Comput. J.*, 57(3):403–426.

Chierchia, G. (1995). *Individual-Level Predicates as Inherent Generics*, pages 176–223. Chicago University Press, Carlson & Pelletier edition.

Chieu, H. L. and Lee, Y. K. (2004). Query Based Event Extraction Along a Timeline. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 425–432, New York, NY, USA. ACM.

Cieri, C., Graff, D., Liberman, M., Martey, N., and Strassel, S. (1999). The TDT-2 Text And Speech Corpus. In *Proceedings of DARPA Broadcast News Workshop*, pages 57–60, Herndon, Virginia, USA. Morgan Kaufmann.

Clark, S. and Curran, J. R. (2007). Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Comput. Linguist.*, 33(4):493–552.

Comrie, B. (1981). *Language Universals and Linguistic Typology: Syntax and Morphology*. University of Chicago Press.

Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Mach. Learn.*, 20(3):273–297.

Cremmins, E. (1996). *The Art of Abstracting*. Information Resources Press.

Dale, R. (1992). Generating Referring Expressions Constructing Descriptions in a Domain of Objects and Processes. In *ACL-MIT press series in natural language processing*.

Davidson, D. (1967). The Logical Form of Action Sentences. In Rescher, N., editor, *The Logic of Decision and Action*. University of Pittsburgh Press.

Davies, W. and Dubinsky, S. (2008). *The Grammar of Raising and Control: A Course in Syntactic Argumentation*. Wiley.

Diab, M. T., Baldwin, T., and Baroni, M., editors (2013). *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*. The Association for Computer Linguistics.

Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. *Numer. Math.*, 1(1):269–271.

Edmundson, H. P. (1969). New Methods in Automatic Extracting. *J. ACM*, 16(2):264–285.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*, 9:1871–1874.

Filannino, M., Brown, G., and Nenadic, G. (2013). Mantime: Temporal expression identification and normalization in the tempeval-3 challenge. In Diab et al. (2013), pages 53–57.

Filatova, E. and Hovy, E. (2001). Assigning Time-stamps to Event-clauses. In *Proceedings of the Workshop on Temporal and Spatial Information Processing - Volume 13*, TASIP '01, pages 13:1–13:8, Toulouse, France. Association for Computational Linguistics.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Ann Arbor, Michigan, USA. Association for Computational Linguistics.

Fleiss, J. L. (1971). Measuring Nominal Scale Agreement Among Many Raters. *Psychological Bulletin*, 76(5):378–382.

Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741.

Genest, P.-E. and Lapalme, G. (2012). Fully Abstractive Approach to Guided Summarization. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 354–358, Jeju Island, Korea. Association for Computational Linguistics.

Gibbs Jr., R. W. (1994). Figurative thought and figurative language. In *Handbook of Psycholinguistics*, pages 411–446, San Diego, CA, US. Academic Press.

Goldstein, J., Mittal, V., Carbonell, J., and Kantrowitz, M. (2000). Multi-document Summarization by Sentence Extraction. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic Summarization - Volume 4*, NAACL-ANLP-AutoSum '00, pages 40–48, Seattle, Washington, USA. Association for Computational Linguistics.

Grishman, R., Westbrook, D., and Meyers, A. (2005). NYU's English ACE 2005 System Description. Technical report, Department of Computer Science, New York University.

Grosz, B. J., Weinstein, S., and Joshi, A. K. (1995). Centering: A Framework for Modeling the Local Coherence of Discourse. *Comput. Linguist.*, 21(2):203–225.

Gundel, J., Hedberg, N., and Zacharski, R. (1993). Cognitive Status and the Form of Referring Expressions in Discourse. *Language*, 69:274–307.

Hahn, U. and Mani, I. (2000). The Challenges of Automatic Summarization. *Computer*, 33(11):29–36.

Harman, D. (2002). The Development and Evolution of TREC and DUC. In Oyama, K., Ishida, E., and Kando, N., editors, *Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering, NTCIR-3, Tokyo, Japan, October 8-10, 2002*. National Institute of Informatics (NII).

Harnly, A., Nenkova, A., Passonneau, R., and Rambow, O. (2005). Automation of Summary Evaluation by the Pyramid Method. In *Proceedings of the Conference of Recent Advances in Natural Language Processing (RANLP) 2005*, Borovets, Bulgaria.

Hassel, M. (2004). *Evaluation of Automatic Text Summarization - A practical implementation*. Licentiate thesis, Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Higginbotham, J. (1985). On Semantics. *Linguistic Inquiry*, 16:547–593.

Higginbotham, J. (2000). On Events in Linguistic Semantics. In Higginbotham, J., Pianesi, F., and Varzi, A., editors, *Speaking of Events*. Oxford University Press.

Hockenmaier, J. and Steedman, M. (2007). CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Comput. Linguist.*, 33(3):355–396.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust Disambiguation of Named Entities in Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 782–792, Edinburgh, Scotland, United Kingdom. Association for Computational Linguistics.

Hollmann, W. B. (2003). *Synchrony and Diachrony of English Periphrastic Causatives: A Cognitive Perspective*. University of Manchester, School of English and linguistics.

Hou, Y., Markert, K., and Strube, M. (2013). Global Inference for Bridging Anaphora Resolution. In Vanderwende, L., III, H. D., and Kirchhoff, K., editors, *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 907–917. The Association for Computational Linguistics.

Hou, Y., Markert, K., and Strube, M. (2014). A Rule-Based System for Unrestricted Bridging Resolution: Recognizing Bridging Anaphora and Finding Links to Antecedents. In Moschitti, A., Pang, B., and Daelemans, W., editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 2082–2093. ACL.

Hovy, E. and Lin, C. Y. (1999). Automated Text Summarization in SUMMARIST. In Mani, I. and Maybury, M. T., editors, *Advances in Automatic Text Summarization*. MIT Press.

Hovy, E. and Marcu, D. (1998). Automated Text Summarization Tutorial – COLING/ACL'98.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2003). A Practical Guide to Support Vector Classification. Technical report, Department of Computer Science, National Taiwan University.

Jing, H., Barzilay, R., Mckeown, K., and Elhadad, M. (1998). Summarisation Evaluation Methods: Experiments and Analysis. In *Intelligent Text Summarization. Papers from the AAAI Spring Symposium*, Stanford, California, USA.

Jones, K. S. and Galliers, J. R. (1996). *Evaluating Natural Language Processing Systems: An Analysis and Review*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, volume 42 of *Studies in Linguistics and Philosophy*. Springer, Dordrecht, The Netherlands.

Kasami, T. (1965). An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages. Technical report, Air Force Cambridge Research Lab, Bedford, Massachusetts, USA.

Kessler, R., Tannier, X., Hagège, C., Moriceau, V., and Bittar, A. (2012). Finding Salient Dates for Building Thematic Timelines. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 730–739, Jeju Island, Korea. Association for Computational Linguistics.

Kolomiyets, O., Bethard, S., and Moens, M. (2012). Extracting narrative timelines as temporal dependency structures. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 88–97. The Association for Computer Linguistics.

Kratzer, A. (1995). Stage-Level and Individual-Level Predicates. In *The Generic Book*, pages 125–175. Chicago University Press, Gregory N. Carlson and Francis J. Pelletier edition.

Krippendorff, K. (1980). *Content analysis: An introduction to its methodology*. Sage Publications, Beverly Hills, California, USA.

Krippendorff, K. (2007). Computing Krippendorff's Alpha Reliability. Technical report, University of Pennsylvania, Annenberg School for Communication.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Laokulrat, N., Miwa, M., Tsuruoka, Y., and Chikayama, T. (2013). UTTime: Temporal Relation Classification using Deep Syntactic Features. In Diab et al. (2013), pages 88–92.

Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., and Jurafsky, D. (2013). Deterministic Coreference Resolution Based on Entity-centric, Precision-ranked Rules. *Comput. Linguist.*, 39(4):885–916.

Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In Marie-Francine Moens, S. S., editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Lin, C.-Y. and Hovy, E. (2002). Manual and Automatic Evaluation of Summaries. In *Proceedings of the ACL-02 Workshop on Automatic Summarization - Volume 4*, AS '02, pages 45–51, Phildadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Linguistic Data Consortium (2005). *ACE (Automatic Content Extraction) English Annotation Guidelines for Events*, 5.4.3 2005.07.01 edition.

Llorens, H., Chambers, N., UzZaman, N., Mostafazadeh, N., Allen, J., and Pustejovsky, J. (2015). SemEval-2015 Task 5: QA TempEval - Evaluating Temporal Information Understanding with Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 792–800, Denver, Colorado, USA. Association for Computational Linguistics.

Llorens, H., Saquete, E., and Navarro, B. (2010). TIPSem (English and Spanish): Evaluating CRFs and Semantic Roles in TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 284–291, Uppsala, Sweden. Association for Computational Linguistics.

Luhn, H. P. (1958). The Automatic Creation of Literature Abstracts. *IBM J. Res. Dev.*, 2(2):159–165.

Maienborn, C. (2005). On the limits of the Davidsonian approach: The case of copula sentences. *Theoretical Linguistics*, 31(3):275–316.

Mani, I. (2001). Summarization Evaluation: An Overview. In *Proceedings of the Third Second Workshop Meeting on Evaluation of Chinese & Japanese Text Retrieval and Text Summarization, NTCIR-2, Tokyo, Japan, March 7-9, 2001*. National Institute of Informatics (NII).

Mani, I., House, D., Klein, G., Hirschman, L., Firmin, T., and Sundheim, B. (1999). The Tipster Summac Text Summarization Evaluation. In *EACL 1999, 9th Conference of the European Chapter of the Association for Computational Linguistics, June 8-12, 1999, University of Bergen, Bergen, Norway*, pages 77–85. The Association for Computer Linguistics.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pages 55–60. The Association for Computer Linguistics.

Marcu, Daniel (1997). From Discourse Structures to Text Summaries. In *Proceedings of ACL Workshop on Intelligent Scalable Text Summarisation*, pages 82–88, Madrid, Spain.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.*, 19(2):313–330.

Markert, K., Hou, Y., and Strube, M. (2012). Collective Classification for Fine-grained Information Status. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 795–804, Jeju Island, Korea. Association for Computational Linguistics.

McDonald, R. (2007). A Study of Global Inference Algorithms in Multi-document Summarization. In *Proceedings of the 29th European Conference on IR Research*, ECIR'07, pages 557–564, Berlin, Heidelberg. Springer-Verlag.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005). Non-projective Dependency Parsing Using Spanning Tree Algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 404–411. ACL.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., Bottou, L., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, USA*, pages 3111–3119.

Minard, A.-L., Speranza, M., Agirre, E., Aldabe, I., van Erp, M., Magnini, B., Rigau, G., and Urizar, R. (2015). SemEval-2015 Task 4: TimeLine: Cross-Document Event Ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 778–786, Denver, Colorado, USA. Association for Computational Linguistics.

Moens, M. and Steedman, M. (1988). Temporal Ontology and Temporal Reference. *Comput. Linguist.*, 14(2):15–28.

Morris, A. H., Kasper, G. M., and Adams, D. A. (1992). The Effects and Limitations of Automated Text Condensing on Reading Comprehension Performance. *Information Systems Research*, 3:17–35.

Navarro, B. and Saquete, E. (2015). GPLSIUA: Combining Temporal Information and Topic Modeling for Cross-Document Event Ordering. In Cer, D. M., Jurgens, D., Nakov, P., and Zesch, T., editors, *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 820–824. The Association for Computer Linguistics.

Nenkova, A. (2005). Automatic Text Summarization of Newswire: Lessons Learned from the Document Understanding Conference. In Veloso, M. M. and Kambhampati, S., editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and*

the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA, pages 1436–1441. AAAI Press / The MIT Press.

Nenkova, A., Passonneau, R., and McKeown, K. (2007). The Pyramid Method: Incorporating Human Content Selection Variation in Summarization Evaluation. *ACM Trans. Speech Lang. Process.*, 4(2).

Nenkova, A. and Passonneau, R. J. (2004). Evaluating Content Selection in Summarization: The Pyramid Method. In Hirschberg, J., Dumais, S. T., Marcu, D., and Roukos, S., editors, *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2004, Boston, Massachusetts, USA, May 2-7, 2004*, pages 145–152. The Association for Computational Linguistics.

Nguyen, K., Tannier, X., and Moriceau, V. (2014). Ranking Multidocument Event Descriptions for Building Thematic Timelines. In Hajic, J. and Tsujii, J., editors, *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 1208–1217. ACL.

Nissim, M. (2006). Learning Information Status of Discourse Entities. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 94–102, Sydney, Australia. Association for Computational Linguistics.

Nissim, M., Dingare, S., Carletta, J., and Steedman, M. (2004). An Annotation Scheme for Information Status in Dialogue. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*. European Language Resources Association.

Nivre, J. (2008). Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34(4):513–553.

Otterbacher, J. C., Radev, D. R., and Luo, A. (2002). Revisions That Improve Cohesion in Multi-document Summaries: A Preliminary Study. In *Proceedings of the ACL-02 Workshop on Automatic Summarization - Volume 4*, AS '02, pages 27–36, Phildadelphia, Pennsylvania, USA.

Paice, C. D. (1990). Constructing Literature Abstracts by Computer: Techniques and Prospects. *Inf. Process. Manage.*, 26(1):171–186.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Parsons, T. (1990). *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press.

Parsons, T. (2000). Underlying States and Time Travel. In Varzi, A., Higginbotham, J., and Pianesi, F., editors, *Speaking of Events*. Oxford University Press.

Passonneau, R. J., Chen, E., Guo, W., and Perin, D. (2013). Automated Pyramid Scoring of Summaries using Distributional Semantics. In *ACL (2)*, pages 143–147. The Association for Computer Linguistics.

Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). WordNet::Similarity: Measuring the Relatedness of Concepts. In *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL–Demonstrations '04, pages 38–41, Boston, Massachusetts, USA. Association for Computational Linguistics.

Platt, J. (2000). Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In Smola, A., Bartlett, P., Schoelkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 61–74.

Prince, E. F. (1992). The ZPG Letter: Subjects, Definiteness, and Information-status. In *Discourse Description: Diverse Analyses of a Fund-raising Text*, pages 295–325, Philadelphia, Pennsylvania, USA. John Benjamins.

Pustejovsky, J., Castaño, J. M., Ingria, R., Saurí, R., Gaizauskas, R. J., Setzer, A., Katz, G., and Radev, D. R. (2003a). TimeML: Robust Specification of Event and Temporal Expressions in Text. In Maybury, M. T., editor, *New Directions in Question Answering, Papers from 2003 AAAI Spring Symposium, Stanford University, Stanford, CA, USA*, pages 28–34. AAAI Press.

Pustejovsky, J., Hanks, P., Sauri, R., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D., Ferro, L., and Lazo, M. (2003b). The TIMEBANK corpus. In *Proceedings of Corpus Linguistics 2003*, pages 647–656, Lancaster, United Kingdom.

Radev, D. R. and McKeown, K. R. (1998). Generating Natural Language Summaries from Multiple On-line Sources. *Comput. Linguist.*, 24(3):470–500.

Rahman, A. and Ng, V. (2011). Learning the Information Status of Noun Phrases in Spoken Dialogues. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1069–1080, Edinburgh, Scotland, United Kingdom. Association for Computational Linguistics.

Rappaport Hovav, M. and Levin, B. (1997). Building Verb Meanings. In Butt, M. and Geuder, W., editors, *The Projection of Arguments: Lexical and Compositional Factors*, pages 97–134. CSLI Publications, Stanford.

Rath, G. J., Resnick, A., and Savage, T. R. (1961). The Formation of Abstracts by the Selection of Sentences. Part 1. Sentence Selection By Men and Machines. *American Documentation*, 12(2):139–141.

Rösiger, I. and Teufel, S. (2014). Resolving Coreferent and Associative Noun Phrases in Scientific Text. In Bouma, G. and Parmentier, Y., editors, *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 45–55. The Association for Computer Linguistics.

Rowley, J. (1982). *Abstracting and Indexing*. Outlines of modern librarianship. C. Bingley.

Salton, G., Singhal, A., Mitra, M., and Buckley, C. (1997). Automatic Text Structuring and Summarization. *Inf. Process. Manage.*, 33(2):193–207.

Saurí, R., Knippen, R., Verhagen, M., and Pustejovsky, J. (2005). Evita: A robust event recognizer for QA systems. In *HLT/EMNLP*, pages 700–707. The Association for Computational Linguistics.

Saurí, R., Littman, J., Gaizauskas, R., Setzer, A., and Pustejovsky, J. (2006). TimeML Annotation Guidelines, Version 1.2.1.

Schilder, F. and Habel, C. (2001). 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, pages 9:1–9:8, Toulouse, France. Association for Computational Linguistics.

Schrijver, A. (1986). *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA.

Setzer, A. (2001). *Temporal Information in Newswire Articles: An Annotation Scheme and Corpus Study*. PhD thesis, The University of Sheffield, Sheffield, United Kingdom.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell system technical journal*, 27.

Shen, W., Wang, J., and Han, J. (2015). Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. *Transactions on Knowledge & Data Engineering*, 27(2):443–460.

Siddharthan, A., Nenkova, A., and McKeown, K. (2011). Information Status Distinctions and Referring Expressions: An Empirical Study of References to People in News Summaries. *Comput. Linguist.*, 37(4):811–842.

Siegel, S. and Castellan, N. (1988). *Nonparametric statistics for the behavioral sciences*. McGraw–Hill, Inc., second edition.

Smith, D. A. (2002). Detecting Events with Date and Place Information in Unstructured Text. In *Proceedings of the 2Nd ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '02, pages 191–196, New York, NY, USA. ACM.

Steinberger, J. and Jezek, K. (2009). Evaluation Measures for Text Summarization. *Computing and Informatics*, 28(2):251–275.

Strötgen, J. and Gertz, M. (2010). HeidelTime: High Quality Rule-Based Extraction and Normalization of Temporal Expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324, Uppsala, Sweden. Association for Computational Linguistics.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA. ACM.

Swan, R. and Allan, J. (2000). Automatic Generation of Overview Timelines. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 49–56, New York, NY, USA. ACM.

Teufel, S. (2010). *The Structure of Scientific Articles: Applications to Citation Indexing and Summarization*. CSLI Studies in Computational Linguistics. Center for the Study of Language and Information.

Tong, S. and Koller, D. (2002). Support Vector Machine Active Learning with Applications to Text Classification. *J. Mach. Learn. Res.*, 2:45–66.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Edmonton, Canada. Association for Computational Linguistics.

Tran, G. B., Alrifai, M., and Quoc Nguyen, D. (2013). Predicting Relevant News Events for Timeline Summaries. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 91–92, New York, NY, USA. ACM.

Tran, G. B., Herder, E., and Markert, K. (2015). Joint Graphical Models for Date Selection in Timeline Summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1598–1607. The Association for Computer Linguistics.

UzZaman, N. and 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.

UzZaman, N., Llorens, H., Derczynski, L., Allen, J., Verhagen, M., and Pustejovsky, J. (2013). SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA. Association for Computational Linguistics.

van Halteren, H. and Teufel, S. (2003). Examining the Consensus Between Human Summaries: Initial Experiments with Factoid Analysis. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*, HLT-NAACL-DUC '03, pages 57–64, Edmonton, Canada. Association for Computational Linguistics.

Vendler, Z. (1967). *Linguistics in philosophy* . Cornell University Press Ithaca, N.Y., USA.

Verhagen, M., Gaizauskas, R., Schilder, F., Hepple, M., Katz, G., and Pustejovsky, J. (2007). SemEval-2007 Task 15: TempEval Temporal Relation Identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 75–80, Prague, Czech Republic. Association for Computational Linguistics.

Verhagen, M., Saurí, R., Caselli, T., and Pustejovsky, J. (2010). SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 57–62, Los Angeles, California, USA. Association for Computational Linguistics.

White, M., Korelsky, T., Cardie, C., Ng, V., Pierce, D., and Wagstaff, K. (2001). Multidocument Summarization via Information Extraction. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–7, San Diego, California, USA. Association for Computational Linguistics.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83.

Yan, R., Kong, L., Huang, C., Wan, X., Li, X., and Zhang, Y. (2011a). Timeline Generation Through Evolutionary Trans-temporal Summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 433–443, Edinburgh, Scotland, United Kingdom. Association for Computational Linguistics.

Yan, R., Wan, X., Otterbacher, J., Kong, L., Li, X., and Zhang, Y. (2011b). Evolutionary Timeline Summarization: A Balanced Optimization Framework via Iterative Substitution. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 745–754, New York, NY, USA. ACM.

Zavarella, V. and Tanev, H. (2013). FSS-TimEx for TempEval-3: Extracting Temporal Information from Text. In *SemEval@NAACL-HLT*, pages 58–63, Atlanta, Georgia, USA. The Association for Computer Linguistics.