Ruprecht-Karls-Universität Heidelberg
Institut für Informatik
Arbeitsgruppe Datenbanksysteme

Master Thesis

Time-Centric Content-Exploration in Large Document Collections

Name: Philip Hausner
Matrikelnummer: 3220550
Betreuer: Prof. Dr. Michael Gertz
Datum der Abgabe: 02.03.2020

Ich versichere, dass ich diese Master-Arbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe und die Grundsätze und Empfehlungen "Verantwortung in der Wissenschaft" der Universität Heidelberg beachtet wurden.

Abgabedatum: 02.03.2020

Zusammenfassung

Zeit ist eine natürliche Art um Informationen zu ordnen, und den Ablauf von Geschehnissen, z.B. mithilfe eines Zeitstrahls, zu strukturieren. Solche Darstellungen ermöglichen es Nutzern Informationen auf einen Blick zu erfassen, und können somit kompliziertere textuelle Daten ersetzen oder zumindest ergänzen. Allerdings kann sich die händische Konstruktion von Zeitstrahlen als mühselig und fehleranfällig erweisen. Zudem sind diese oftmals statische Representationen von gegebenen Informationen und erlauben es dem Nutzer nicht mit den zugrunde liegenden Daten zu interagieren, was den Benutzer auf eine passive Rolle beschränkt. Explorative Szenarien zielen darauf ab dem Nutzer eine aktive Rolle zu geben, in der er nicht nur Informationen aufnimmt, sondern auch die Präsentation beeinflussen kann. In dieser Thesis skizzieren wir inwieferen Graphenmodelle genutzt werden können um Dokumentensammlungen in zeitorientierter Art zu erkunden, und nutzen zeitliche Informationen um unstrukturierte textliche Daten automatisch zu strukturieren. Um das zu erreichen, wenden wir Techniken aus der Computerlinguistik wie Wortkookkurrenzen und named entity recognition an, wobei wir einen besonderen Fokus auf die Extraktion temporaler Informationen legen. In dieser Arbeit verwenden wir für diese Extraktion das domänensensitive Programm HeidelTime. Die extrahierten zeitlichen Ausdrücke werden dann genutzt, um ein Graphenmodell aufzubauen, das jedem zeitlichen Ausdruck ein Term-Kookkurrenz Netzwerk zuweist, sodass es für jeden in der Dokumentensammlung genannten Zeitpunkt genau einen Graphen gibt. Zusätzlich werden diese Netzwerke in verschiedene Granularitäten unterteilt, abhängig davon ob sich der zugehörige zeitliche Ausdruck auf ein Jahr, einen Monat oder einen Tag bezieht. Um die unterschiedliche Relevanz eines Terms in Bezug auf einen bestimmten Zeitpunkt zu berücksichtigen, führen wir eine termhäufigkeits-inverse Zeitstempelfrequenzmetrik ein, die Terme für ein gegebenes Netzwerk nach Relevanz ordnet. Darüber hinaus präsentieren wir mehrere Explorationsszenarien, in denen das vorgeschlagene Modell angewendet werden kann, und skizzieren in diesem Zusammenhang ein Gerüst, das es Nutzern erlaubt Dokumentensammlungen zu erkunden, und so Erkenntnisse über die Chronologie von Ereignissen zu bekommen. Um den Nutzen unserer Methode zu demonstrieren, wird das Modell auf zwei verschiedenen Datensätze aus einer juristischen und historischen Domäne angewandt, und die Ergebnisse diskutiert. Dabei zeigen wir, dass die automatisch generierten Netwerke Informationen aus den jeweiligen Datensätzen zuverlässig repräsentieren.

Abstract

Time is a natural way to order information, and structure the sequence of events, e.g., along a timeline. Such representations allow users to grasp information at-a-glance and can hence replace or at least complement more complicated textual data. The manual construction of timelines, however, can prove to be a tedious and error-prone task. Moreover, such timelines are often static representations of the information given in a data set, and do not allow users to interact with the underlying data, limiting the user to a passive role. Exploratory scenarios aim to give a user an active role, in which she not only absorbs knowledge, but also influences in which ways the information is presented to her.

In this thesis, we outline how graph models can be employed to explore document collections in a time-centric fashion, and use temporal information to automatically structure otherwise unstructured textual data. To achieve this, techniques from natural language processing like word co-occurrence extraction and named entity recognition are discussed and employed, focusing particularly on temporal information extraction, which in this thesis is done with the help of the domain-sensitive temporal tagger HeidelTime. Extracted temporal expressions are then used to build a graph model that assigns a term co-occurrence network to each temporal expression present in the document collection, s.t. there is exactly one graph for each point in time mentioned in the data set. Additionally, networks are subclassified depending on the type of time granularity they represent, i.e., if the associated temporal expression is of year, month or day granularity. To account for the varying relevance of a term or entity in regard to a timestamp, a tf-inverse timestamp frequency metric to rank terms for a given set of networks is introduced. Furthermore, we present multiple exploratory scenarios in which the proposed model can be applied, and in this context outline a framework that allows users to explore and gain insights about the chronology of events in a given document collection. To demonstrate the usefulness of the approach, the model is employed to two different data sets, one from a legal and one from a historical domain, and the respective results are discussed, showing that the automatically constructed networks accurately represent information given in the respective data set.

Contents

1 Introduction

1.1 Motivation

Every day a multitude of documents, containing even more sentences and words, are created and often published in electronic form. Wikipedia alone consists of more than 6 million articles to date[1], the New York Times published more than 13 million online available articles[2], and about 500 million tweets are sent each day[3]. This abundance and variety of textual data offers opportunities as well as challenges in the field of information extraction. Typical examples can also be found in the legal domain: In large document collections about complex court cases, journalists and officials have created lots of textual data: Statutes, court protocols, expert opinions, news stories, official governmental statements, and the like, can be analyzed to get insights into cases like the Enron scandal, the Panama papers, the CumEx-Files, or the National Socialist Underground (NSU) trial. For the Panama papers alone, nearly 400 journalists investigated[4], creating a lot of textual data in the meantime. However, manually reading all these documents and summarizing or highlighting the most important key points quickly becomes a tedious, costly and error-prone task. In this context, it needs to be distinguished between electronic and non-electronic data. For information stored in an electronic format, e-discovery and technology assisted review systems are developed, aiding users to analyze electronic data. With that in mind, information extraction aims to extract and organize information from various data collections with the goal of presenting relevant content to a certain audience, which, depending on the use case, can consist of only a few individuals or the broad public. Typical methods originating from *natural language processing* (*NLP*) like sentence segmentation enable us to process text and transfer the data from a pure textual representation to one that is more suitable for information extraction tasks. Moreover, techniques such as *Named Entity Recognition* already provide means to extract valuable information from textual data. In this context, temporal information certainly plays

[1] `https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia` ; accessed 3. Feb. 2020

[2] `https://archive.nytimes.com/www.nytimes.com/ref/membercenter/nytarchive.html` ; accessed 3. Feb. 2020

[3] `https://www.dsayce.com/social-media/tweets-day/` ; accessed 3. Feb. 2020

[4] `https://ethicaljournalismnetwork.org/resources/publications/ethics-in-the-news/panama-papers` ; accessed 3. Feb. 2020

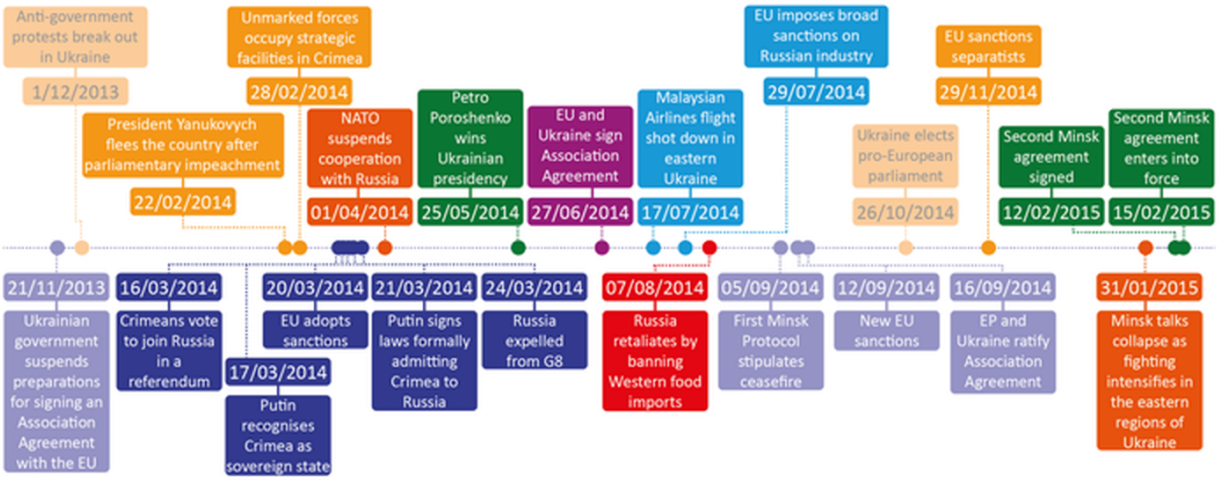


Figure 1.1: Manually created timeline for the Ukrainian crisis depicting major events between November, 2013, and February, 2015.[5]

an important role to organize such data, since it allows for the grouping of data around anchor points. Additionally, it enables us to order events in a chronological order, a concept that humans can intuitively grasp, and which can be visualized in the form of timelines, utilizing the eyes as probably the best information transmitter we have as a human being: Research suggests that the transmission rate of our eyes is about as fast as an Ethernet connection [27]. Figure 1.1 shows a manually created timeline for the Ukrainian crisis, which allows for observation of the major events of the crisis at-a-glance, instead of searching for this information potentially for hours. Based on this concept, time-based extraction methods to organize information present in large document collections are discussed, and results are depicted in the form of timelines. Furthermore, the concept of timelines is combined with co-occurrence graphs by associating each point on the timeline with a network representing the respective point in time, in which nodes represent various terms and entities related to the point in time, and edges describe relations between those terms and entities. By employing characteristics like different time granularities, tools can be built that not only present information to the user, but also allow her to interact with the data, e.g., by exploring the occurrence of specific entities with regard to varying granularity levels. Nonetheless, constructing such timelines comes also with its own set of problems: Simply exposing the user to a timeline with hundreds of timestamps, each containing a network with a multitude of nodes, is not desirable. Instead, we investigate ways to reduce the number of timestamps and nodes in a timeline to provide concise representations for users. By doing so, we aim to

[5]European Parliamentary Research Service, 13. Feb., 2015, `https://epthinktank.eu/2015/02/13/minsk-peace-agreement-still-to-be-consolidated-on-the-ground/eprs-briefing-548991-minsk-peace-summit-fig1-timeline/` ; accessed 28. Jan. 2020

enable a user to explore the content of a large document collection by herself without the need to read through the whole data set, ultimately providing meaningful insights about the chronology of the events in the data set as well as the occurring entities.

1.2 Goals of this Thesis

This thesis aims to provide a time-centric approach for textual data exploration, organizing information around temporal expressions, and presenting the results in the form of flexible timeline representations. To achieve this, we introduce a novel time-centric graph representation that not only allows to organize textual data employing co-occurrence information, but also enables us to order them chronologically. This representation facilitates a manner in which large document collections can be explored by a user without the need to read all of the text by herself. Moretti termed techniques like this Distant Reading [43], which aims to study and analyze massive amounts of text instead of single documents to gain insights about a broader spectrum of literature. Furthermore, several scenarios in which the proposed model can be employed for efficient data exploration are presented, which among others utilize an introduced time hierarchy. Finally, we propose a metric similar to the well-known tf-idf scheme to rank the importance of terms and entities with regard to a given timestamp. Specifically, the contributions of this work include:

- An introduction to time-centric data processing.
- Time-centric co-occurrence graphs as a means to organize terms around temporal expressions in the form of co-occurrence graphs.
- Several scenarios that enable users to explore document collections in an efficient manner are introduced, and different options to utilize the proposed methods considered.
- A timeline implementation employing time-centric co-occurrence graphs is provided.
- We assess the value of temporal information extracted from German texts by the multilingual domain-sensitive temporal tagger HeidelTime by analysing the extracted temporal expressions of a data set containing juridical protocols of the *National Socialist Underground* (NSU) trials.
- Finally, we show that the extracted time-centric co-occurrence graphs are representative of the document contents by applying the method to the above-mentioned data set about the NSU trials as well as to a data set concerning the Weimar Republic extracted from the German Wikipedia.

1.3 Outline

This thesis is structured in the following way: In Chapter 2, we present the fundamentals necessary for this work. This includes an introduction to graph theory, natural language processing, as well as the extraction of temporal information. Finally, an overview of related work and literature relevant to this thesis is provided. Chapter 3 introduces the underlying document model, establishes a concept of time and timelines, and presents several time-centric graph representations. Afterwards, time-centric co-occurrence graphs are proposed, and eventually, the chapter concludes by demonstrating several approaches for timeline exploration. In Chapter 4, the reader is provided with implementational details, and evaluate the performance of the temporal tagger HeidelTime. Moreover, we show the usefulness of the proposed approach by applying our methods to a given data set. Finally, we conclude this work with a brief summary and provide an outlook on potential future work in Chapter 5.

2 Fundamentals

This chapter introduces the fundamentals necessary for this work. Firstly, we establish a foundation in graph theory, including appropriate structures to store graphs. Secondly, several challenges in *Natural Language Processing* (NLP) are discussed, in particular with a focus on text processing. Thirdly, the concept of co-occurrences is introduced, and then the extraction of temporal information is covered. Hereby, a focus is placed on HeidelTime, a tool that is commonly used to extract temporal information from text documents. Finally, an overview of related work relevant to this thesis is given.

2.1 Graph Theory

This section discusses the fundamentals of *graph theory*, a field of mathematics which studies *graphs*. In this thesis, we often also refer to graphs as so-called *networks*. While the definition of graphs and networks is often different from one another, we do not differ between the two. Thus, when we later talk about networks, the term can always be replaced by the term graph. For the definitions given in this section, we mostly refer to Latora et al. [33] and van Steen [73].

2.1.1 Graphs

The most basic subjects of graph theory are undirected graphs. When not stated otherwise, we always refer to undirected graphs when discussing networks.

Definition 1 (Undirected Graph) *An **undirected graph**, or simply **graph**, $G = (N, L)$ consists of two sets, $N \neq \varnothing$ and L. The m elements of $N = \{n_1, n_2, ..., n_m\}$ are distinct, and are referred to as **nodes**, **points**, or **vertices**. The second set is $L = \{l_1, l_2, ..., l_k\} \subseteq N \times N$, s.t. L is represented by k unordered mutually distinct pairs of elements of N. We call elements of L **links**, **lines**, or **edges**, respectively.*

In the context of this work, the symbol $|\cdot|$ denotes the cardinality of a set, i.e., $|N|$ refers to the number of nodes, and $|L|$ to the number of links, respectively. Since links are defined by the nodes n_i and n_j they connect, we often do not refer to links by their label l_k, but

denote them as l_{ij}, (n_i, n_j), or simply (i, j). Additionally, for undirected graphs, $l_{ij} = l_{ji}$ holds, because the pairs L consists of are unordered. The nodes n_i and n_j are referred to as the end-nodes of link l_{ij}. A link that connects a node to itself, i.e., a link $l_{ii} = (n_i, n_i)$, is called a self-loop. Finally, if there exists more than one link connecting two nodes, we speak of a multi-link joining the two nodes.

Definition 2 (Subgraph) *Given a graph $G = (N, L)$, a **subgraph** of G is a graph $G' = (N', L')$ with $N' \subseteq N$ and $L' \subseteq L$. If G' consists of all links $l \in L$ that join two nodes in N', i.e., there exists no link $l_{ij} \in L : l_{ij} \notin L' \wedge n_i \in N' \wedge n_j \in N'$, then we call G' **induced** by N', and write $G' = G[N']$.*

Subgraphs allow us to zoom in on a part of the graph, and e.g., allow us to identify and isolate regions in a graph with a specific (un-)desirable property.

2.1.2 Directed and Weighted Graphs

In some cases, it might be useful to specify the direction of a link, since it can be a difference if node n_i is connected to node n_j via a link l_{ij}, or the other way around via a link l_{ji}. An example would be a citation network, in which we can differ between the person who cites and the one who is cited. To incorporate this into graph theory, *directed graphs* are introduced.

Definition 3 (Directed Graph) *A graph $G = (N, L)$ is called **directed** if the links contained in $L = \{l_1, l_2, ..., l_k\} \subseteq N \times N$ are ordered pairs of elements of N.*

Note that the only difference between Definition 1 and Definition 3 is that for directed graphs, L consists of *ordered* pairs of elements, thus, it no longer holds that $l_{ij} = l_{ji}$. In a directed graph, and given a link l_{ij}, we refer to n_i as the source node and n_j as the target node. For clarity, we sometimes call elements of L *directed links*, *lines*, or *edges*.

Definition 4 (Weighted Graph) *Given a mapping $w : L \to \mathbb{R}$, a graph $G = (N, L)$ is called **weighted** if a numerical value is assigned to each edge with respect to w.*

The weight of an edge usually represents the strength of the connection between two nodes in a graph, and is generally a positive real number. For example, we may have a model where cities are modeled as nodes and streets between the cities as edges. We could then integrate the length of the streets by using them as weights for our model.

2.1.3 Bipartite and Multipartite Graphs

In a *bipartite graph* the nodes of the graph can be divided into two disjoint sets, such that only links exist that connect one node of the one set to a node of the other set, while no links connect nodes belonging to the same set. A typical example would be a *document/term graph* where there is a set of documents, a set of terms and there exists an edge (t_i, d_j) if term i appears in document j.

Definition 5 (Bipartite Graph) *A **bipartite graph** $G = (N, V, L)$ is a triple consisting of the disjunct sets $N = \{n_1, n_2, ..., n_m\}$, $V = \{v_1, v_2, ..., v_w\}$ and the links $L = \{l_1, l_2, ..., l_k\} \subseteq N \times V$, where N and V are non-empty. In accordance with previous graphs, N and V are called the **nodes**, **points**, or **vertices** of G. L consists of distinct unordered pairs of elements, where one element is from N and the other from V. We also call them **links**, **lines**, or **edges**, respectively.*

Naturally, we can extend this definition to directed and weighted bipartite graphs. This can be done analogously to Definition 3 and Definition 4.

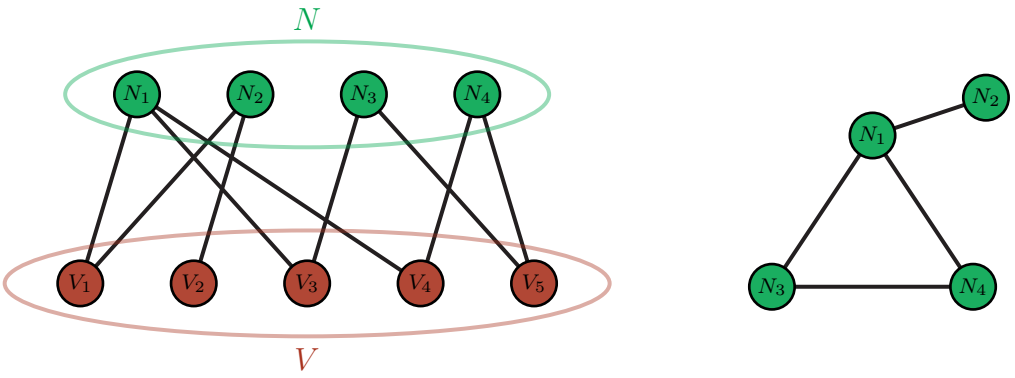
One-Mode Projection

Oftentimes, we are interested in the behaviour of one of the sets of a bipartite graph with regard to its links to the other set. *One-mode projection* enables us to achieve exactly that by taking into account only nodes from the set N or V, and establishes a link between two nodes if they have at least one common neighbour in the other set. More formally:

Definition 6 (One-Mode Projection) *Given a bipartite graph $G = (N, V, L)$ the **one-mode projection** of G onto N yields a graph $G' = (N, L')$ with*

$$L' = \{(n_1, n_2) : n_1 \in N \wedge n_2 \in N \wedge \exists v : (v \in V \wedge (n_1, v) \in L \wedge (n_2, v) \in L)\}. \tag{2.1}$$

Respectively, we can construct a one-mode projection of G onto V. Figure 2.1 illustrates the construction with an example. Given a document/term graph G, a one mode projection of documents onto G yields a graph that connects documents if they share a common term; a projection of terms onto G a graph of terms that are connected if they occur in the same document. Since this projection does not consider how many neighbours two nodes share, or how many links the common neighbour has in total, we can improve it by introducing an additional weighting scheme. A simple weighting function can map the number of neighbours two nodes share to the edge, however, more elaborate weighting functions are possible.



(a) (b)

Figure 2.1: (a) Example bipartite network G with two node sets N and V, and (b) one-mode projection of G onto N.

Taking into account this reasoning and given a one-mode projection of $G = (N, V, L)$ onto N, Newman [45] proposed a weighting scheme as follows:

$$w_{ij} = \sum_{k \in V} \frac{\delta_i^k \delta_j^k}{n_k - 1}. \tag{2.2}$$

Here n_k denotes the number of neighbours of node v_k and

$$\delta_i^k = \begin{cases} 1, & \text{if } (n_i, v_k) \in L, \\ 0, & \text{else.} \end{cases} \tag{2.3}$$

Multipartite Graphs

While bipartite graphs divide nodes into two sets, *multipartite graphs* partition nodes into an arbitrary number of disjoint sets. Similar to bipartite graphs, edges can only exist between two nodes of different sets, but not between nodes of the same set. Formally:

Definition 7 (Multipartite Graphs) *A **multipartite graph** with m partitions $G = (N_1, ..., N_m, L)$ is a $(m+1)$-tuple consisting of the nonempty sets N_1 to N_m, with $N_1 \cap N_2 \cap ... \cap N_m = \varnothing$, and the links $L = \{l_1, l_2, ..., l_k\} \subseteq N_i \times N_j$ with $i \neq j \wedge 1 \leq i, j \leq m$.*

Note that a multipartite graph with two partitions is equal to a bipartite graph, and a graph with three partitions is called a tripartite graph. For multipartite graphs, different projections similar to one-mode projection are possible.

2.1.4 Matrix Representations

In this section, we introduce two different possibilities to represent graphs. Firstly, we present *adjacency matrices*, and secondly, *edge lists* as a representation for sparse graphs.

Adjacency matrices offer a simple and convenient way to store graphs. In an adjacency matrix, rows and columns represent the vertices of the graph while the entries indicate links between the vertices. A formal definition is given by:

Definition 8 (Adjacency Matrix) *Given a graph $G = (N, L)$, the **adjacency matrix** $A \in M^{|N|\times|N|}$ of G with entries a_{ij} is given by*

$$a_{ij} \neq 0 \iff l_{ij} \in L. \tag{2.4}$$

For unweighted graphs, a_{ij} is usually set to 1 if a link between the nodes i and j exists. For weighted graphs, the entry a_{ij} is usually determined by the weight of the link between nodes i and j, i.e., $a_{ij} = w_{ij}$. For directed and undirected graphs, the adjacency matrix is always squared, for undirected graphs even symmetric. However for bipartite graphs, the definition is slightly altered:

Definition 9 (Adjacency Matrix for Bipartite Graphs) *Given a graph $G = (N, V, L)$, the **adjacency matrix** $A \in M^{|N|\times|V|}$ of G with entries a_{ij} is given by*

$$a_{ij} \neq 0 \iff l_{ij} \in L. \tag{2.5}$$

Note that the main difference to Definition 8 is that A is no longer squared. Rows represent nodes of set N, and columns nodes of set V.

Adjacency matrices provide an intuitive and effective way to represent graphs. However, for sparse networks, i.e., graphs with only few links in comparison to the number of nodes, the adjacency matrix is also sparse. In this cases, it is often beneficial to resort to another representation, which enables us to store only non-zero elements. For that reason, so-called *edge lists* are introduced.

Definition 10 (Edge List) *Given a graph $G = (N, L)$ and its associated adjacency matrix A, the **edge list** E of G is a set defined by*

$$E = \{(i, j) \mid a_{ij} \neq 0\}. \tag{2.6}$$

In other words, an edge list stores the pairs of indices i and j, where $A_{ij} \neq 0$. Note that, strictly speaking, in the case of undirected graphs, each edge is stored twice, since (i, j) as well as (j, i) are contained in E. However, in practice, one would only store one of the tuples without any loss of information. Definition 10 can also be extended to weighted and bipartite graphs, but since this is a rather straightforward procedure, we will skip a formal definition.

2.2 Natural Language Processing

In this section, we introduce the reader to the most relevant concepts of *natural language processing* (NLP) in the context of this work. Since this can by no means be an exhaustive introduction to NLP, we also refer the interested reader to Manning and Schütze [37], and the more recent work of Martin and Jurafsky [40], two of the most influential books in computational linguistics. While NLP includes all tasks that enable computers to understand human natural language, we limit ourselves here to written text, which we can find, e.g., in news articles. Additionally, we assume that the given documents are already preprocessed, e.g., certain characters, that differ from our desired character encoding, are removed or altered, and noise like HTML tags are discarded. During this thesis, the focus is placed on the English and German language, ignoring challenges that may arise in other languages like Chinese that exhibit major differences to their European counterparts.

2.2.1 Natural Language Processing Pipelines

When processing large documents, usually a so-called *natural language processing pipeline* is constructed. This enables for splitting up the complex task of NLP into numerous subtasks that are processed separately. Those subtasks can either be completely independent from each other or build upon each other; normally, it is a combination of both. This approach has several advantages compared to a static solution: Firstly, it offers more flexibility, since it allows to exchange single subtasks without having to redesign the whole pipeline, as long as some agreed upon data stream format is followed. Secondly, parts of the pipeline can be reused for different projects, and combined in diverse ways to solve future problems. In the following, we discuss some of the subtasks usually present in a typical pipeline. For the sake of completeness, it has to be mentioned that most NLP pipelines include *part-of-speech tagging* as well as *dependency parsing* upon which some of the functionalities presented build upon, however, these topics are excluded here, since they are highly complex and beyond the scope of this thesis.

2.2.2 Sentence Segmentation

Sentence segmentation, *sentence boundary disambiguation* or *sentence splitting* enables for breaking text apart into sentences, which further subdivides the large problem of understanding a whole text into smaller pieces. The most important hint for this task is punctuation, e.g., periods or exclamation marks. While in most cases segmenting sentences in regard to punctuation is already enough, punctuation can also be highly ambiguous. Consider a word like *"Mr. Smith"*: Using a naive approach, some models would at this point split the text into two sentences, however, a human can clearly distinguish between this use of a point and a sentence boundary marker. Although it is helpful to make use of an abbreviation dictionary for these cases, it cannot resolve the problem by itself. For example, it is not unusual that the expression *"Inc."* is placed at the end of a sentence, while *Mr.* is normally not; and thus, from knowledge about a word being an abbreviation, we cannot infer if the period is an abbreviation point or a sentence boundary marker. The problem becomes even more difficult in cases where emoticons, decimal points, slang, or other non-boundary uses of points have to be expected.
Even though it is possible to develop purely rule-based methods for sentence segmentation, state-of-the-art methods most often use a machine learning approach to tackle the problem [49, p. 18].

2.2.3 Word Tokenization

Another crucial step in NLP is *word tokenization*, i.e., the segmentation of text into separate words. A good indicator for word separation is the white space, however, as with sentence segmentation, we can identify some problems with simply splitting text at a space. For example, it might be useful to further split expressions like "you're" to yield the two tokens *"you"* and *"are"* as a result. Another obstacle, that limits simple space splitting, are compound nouns that can be either written as a single noun or two separate words, like *"ice box"* and *"icebox"*, or *"white board"* and *"whiteboard"*. Oftentimes, one of the first steps in NLP is detecting such compounds, and either splitting them up, or marking them as one word. This is similar to fixed multiword expressions like *New York*, in which a word is composed of multiple words, however, we usually want those to be detected as a single token. Therefore, this step is largely tied to *Named Entity Recognition*, which is discussed later in Section 2.2.6. Lastly, punctuation can be critical: A dot like in *"Ph.D."* should be kept, whereas commas in an enumeration should be dropped, and thus, plainly removing special characters is not possible. As can be observed from the points above, word tokenization is a complex task in itself. The described obstacles are only some of the most important in word tokenization, and in other languages completely different challenges may arise.

2.2.4 Word Normalization

When processing texts, it is often beneficial if two different forms of the same word, are identified identically. For example, looking at the words *"apartment"* and *"apartments"*, it is usually enough to represent the word in both cases as *"apartment"*. While on the surface they appear to be different, their root is the same; in both cases the information is mostly identical. In the worst case, ignoring this would lead to cases where basically identical words would be treated completely different. Reducing words to their base form is called *lemmatization*. This is often achieved by morphological parsing, which determines the morphemes, i.e., smallest meaningful units in a language, of a word. This is achieved by dividing a word into its *stem*, which mainly defines its meaning, and *affixes*, i.e., any additional information. One of the simplest methods to perform this is so-called *stemming*, which mainly removes the word-final affixes. However, stemming does not always generate desirable results. For example, observing the word *"caring"* would yield the base word *"care"* if correctly lemmatized, while stemming would produce the base word *"car"* since it would just remove the *"ing"* part of the expression. Obviously, the meaning of the term *car* largely differs from the meaning of *care*. Projects like *WordNet*[1] offer more sophisticated approaches for lemmatization, which yield more desirable results.

2.2.5 Stop Words

Some words contain less information than others. Typically, the words that carry the least amount of semantic information are filler words that are repeated frequently. In English this includes word like *"a"*, *"and"* and *"the"*. Stop words are usually filtered out, because not only are they disadvantageous for statistical analysis, but in general they also account for the most occurring words in a corpus. Hence, by removing them, we can improve our results, and shorten computation times simultaneously. For this task many pre-defined stop word lists exist for various languages, and thus, the only challenge here is to pick the most suitable for the task at hand.

2.2.6 Named Entity Recognition

Named entity recognition (NER) is one of the crucial tasks in *information extraction*, which turns unstructured information present in text into semantically meaningful and structured data. NER aims to detect and label all occurrences of named entities in a given text. Hereby, a named entity is an object that can be denoted with a proper name like *"Angela Merkel"*, *"Paris"*, or *"Microsoft"*. This includes persons, organizations, places, and, depending on

[1] `https://wordnet.princeton.edu/` ; accessed: 5. Oct., 2019

the task, further predefined categories. Naturally, a named entity can consist of more than one token. Since NER is often developed with one task in mind, results are not easy to transfer to different domains: A model, that aims to find names of drugs in pharmaceutical documents, will yield no satisfying outcomes in legal texts. Oftentimes, NER also includes time, and we come back to that specific topic in Section 2.4 where we will discuss *temporal information extraction* in more detail. Recognizing named entities is useful in many cases[2], e.g.,

1. Classification of news articles: By recognizing the key entities in an article, we can infer its general topic, and assign them a certain category like sports or politics.

2. Search algorithms: Querying large information databases like Wikipedia can be sped up if we only have to match the query with named entities occurring in each article, in contrast to scanning whole articles.

3. Recommender systems: News websites often show users further reading at the end of an article. This can be enabled by linking to documents that mention similar named entities as the one the user is reading at the time.

4. Research: Getting a grasp of all relevant papers regarding a certain field can be difficult. By organizing them with respect to certain categories, e.g., author, general topic, etc., this can help scientists keep track of their field.

5. Question answering: Reducing a question and its possible answers to their named entities, can help to single out the desirable answers, and thus, the process can be accelerated significantly [42].

All of these applications have in common that they help users to discover and explore content which would otherwise be obscured by too much noise around the important clues in a document, and hence, single out meaningful from irrelevant information.
However, NER is also challenging, since, as with other NLP tasks, it has to deal with ambiguity and boundary detection. For example, *"London"* can among other things refer to the capital of England, or to a person like *"Fritz London"*, both of which are valid named entities. Regarding boundary detection, a useful NER model has to consider cases in which an entity consists of multiple words, like *"USS George Washington"*. *"Washington"* alone can refer to many different subjects, e.g., a state of the U.S. or its capital, while *"George Washington"* most probably hints to the first president of the U.S., or some other person. In this special case, however, the complete expression is referring to a vessel of the U.S.

[2] `https://towardsdatascience.com/named-entity-recognition-applications-and-use-cases-acdbf57d595e` ; accessed 5. Oct., 2019

Navy. This illustrates that for employing NER, it is not sufficient to establish a dictionary containing desired entities, but also the context is of crucial importance. Algorithms for NER range from rule-based approaches to more recent advances using neural nets, particularly LSTMs (*long short-term memory*), a *recurrent neural network* architecture [31].

2.3 Word Co-Occurrences

Many methods in NLP, e.g., a lot of recent word embedding models, rely on the extraction of so-called *word co-occurrences*, or short *co-occurrences*. We say, two terms co-occur if they appear together in a certain distance, e.g., in a fixed window defined by a number of words or sentences. This window is not necessarily symmetrical, e.g., one could only be interested in co-occurrences with words that appear before the word we are currently looking at; or we could potentially introduce a weighting that attaches more importance to words close by. To be precise, this window based approach is often coined *positional co-occurrence*, but in the context of this thesis the two terms are used interchangeably and so-called *relational co-occurences* are ignored. Different window types and sizes extract diverse results, however, typically co-occurrence over a short distance indicates syntactical relationship between words while choosing a larger window for co-occurrence extraction yields more semantic similarities [12, p.14]. An extensive introduction to co-occurrences is given by Evert [15].

But why are we interested in co-occurrences in the first place? By counting the frequency of two words co-occurring,statistical association between the two terms can be inferred, i.e., it can be assumed that the two terms are connected, either syntactically or semantically. For example, we would expect that the terms *"coffee"* and *"cup"* occur more often together than two random words on average, since they are semantically related. In general, co-occurrences are more meaningful if two terms occur frequently in the same context. In the following, we introduce *co-occurrence matrices*, as well as a way to weight co-occurrences aside from simple frequency counts.

2.3.1 Co-Occurrence Matrix

A *co-occurrence matrix* can be specified in multiple ways, but here *term-term matrices*, also called *word-word matrices*, are discussed.

Definition 11 (Term-Term Matrix) *A* ***term-term matrix*** $M^{|V|\times|V|}$ *is a matrix where each entry* m_{ij} *represents the number of co-occurrences of the target word* $w_i \in V$ *with the context word* $w_j \in V$*. Hereby,* V *is the given vocabulary, and* $|V|$ *is the number of words in this vocabulary.*

The co-occurrence frequencies between w_i and w_j are determined as described in Section 2.3. Hence, a high score in the matrix indicates relatedness between the two words. It should be noted that the term-term matrix can become huge, since our vocabulary can easily comprise more than 10,000 words. However, most of the entries of M will be 0 in most applications; and therefore, it is often beneficial to employ sparse matrix representations.

Another possibility to construct a co-occurrence matrix is to swap the columns of M and instead of counting co-occurrences with words, we capture the number of times a term appears in a certain document. Such a matrix is called a *term-document matrix*, and represents a different kind of co-occurrence from the one defined above. This approach relies on the assumption that terms are more similar to each other when they appear together in the same document. More formally:

Definition 12 (Term-Document Matrix) *A* ***term-document matrix*** $M^{|V|\times|D|}$ *is a matrix where each entry* m_{ij} *represents the number of times target word* $w_i \in V$ *occurs in document* $d_j \in D$*. Hereby,* V *represents the given vocabulary, and* D *the documents in the collection.*

2.3.2 Co-Occurrence Networks

Co-occurrence networks provide an intuitive way to give insights into extracted co-occurrences of a document or some other context. Such a co-occurrence network can be constructed from any given co-occurrence matrix: For a term-term matrix, a network is obtained in which each node represents a term, and the edges between the nodes represent co-occurrences. Thereby, edge weights can be introduced with regard to the entry in the matrix. For a term-document matrix a bipartite graph is yielded, in which one set of nodes consists of terms, and the other of documents. Edges between the two sets indicate in which document a specific term occurs.

However, oftentimes we are not interested in including all terms or documents, but only take into account certain terms, e.g., only persons or locations. This allows for extraction of networks in which persons that appeared in the same context can be observed, or of bipartite graphs that visualize persons who are associated with specific locations. For analysis of these graphs, network science already provides a wide range of tools, which further emphasizes the advantages of representing co-occurrences by such networks.

2.3.3 Term Frequency - Inverse Document Frequency

Term frequency - inverse document frequency (tf-idf) is a popular approach for term weighting, which among other things helps to filter stop words in text data. Unsurprisingly, tf-idf is composed of term frequency (tf) and inverse document frequency (idf), both of which are defined and calculated independently. An introduction to tf-idf is also given by Manning et al. [38].

The simplest choice for tf is the raw count of a term in a document d. For tf, we denote

$$\text{tf}(t, d) = f_{t,d}, \tag{2.7}$$

with $f_{t,d}$ being the raw term count of term t in document d. Various other variants of tf exist, e.g., a logarithmic scaling can be introduced, i.e., $\text{tf}(t, d) = log(1 + f_{t,d})$.

Idf in regard to a term and a document set D is typically defined as follows:

$$\text{idf}(t, D) = log\left(\frac{N}{1 + |\{d \in D : t \in d\}|}\right), \tag{2.8}$$

with N being the number of documents in the set, i.e., $N = |D|$, and the denominator being the number of documents in which the term occurs. We add 1 to the denominator to prevent zero-division. As for tf, there exist more complex schemes to calculate idf.

Finally, tf-idf is defined as

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D). \tag{2.9}$$

Tf-idf achieves two things. Firstly, it weights terms in regard to the number of their occurrences, which is desirable, since words that occur more often are also likely to be more important. Secondly, terms that occur in many documents of our document set are not specifically meaningful for one document, but are only often used in general; they do not aid to differentiate between two documents. Applying tf-idf, hence, enables us to find terms that are unique to only a few documents, and that may stand out in some regard. It should be noted that while tf-idf is not directly applicable to term-term matrices, they offer a great way to weight term-document matrices.

2.4 Temporal Information Extraction

One of the key tasks in NLP is the extraction of meaningful and correct temporal information. For example, we might be interested in events that happened during *"October 1997"* or *"last week"*. We might even have more complex questions like *"Who was German chancellor after Konrad Adenauer?"*, which requires us to know when Konrad Adenauer was chancellor as well as a comprehensive list of German chancellors that includes years of service. With that in mind, *HeidelTime* is presented, a state-of-the-art multilingual temporal tagging tool introduced by Strötgen and Gertz [66], which allows for the acquisition of temporal information from textual data.

HeidelTime

The publicly available tool Heideltime[3] tackles two problems in the context of temporal information extraction: Firstly, it identifies temporal expressions; secondly, it normalizes these expressions with respect to a standard format. In the following, the different temporal expressions HeidelTime extracts are presented in the context of the *TimeML* standard markup language introduced by Pustejovsky et al. To mark temporal expressions in a text, TimeML employs the *TIMEX3* tag [18]. However, this section serves only as an introduction to TimeML. For a comprehensive description, we refer the interested reader to the work of Pustejovsky et al. [51] as well as to their website[4].

Different Expressions

Firstly, temporal expressions can be categorized with respect to a *type* as follows:

1. *Date* refers to a specific calendar time, and its granularity is always a day or larger unit, i.e., week, months, etc. Examples include *"August 4, 1979"*, *"1961"* or *"today"*.
2. *Time*, which refers to a time of the day. A tag of type Time can be as specific as *"10:28 p.m."*, but also more vague, e.g., *"in the early morning"*.
3. *Duration* describes a period of time, e.g., *"2 hours"*, *"a decade"*, or *"all year"*.
4. *Sets* are used to describe reoccurring events, e.g., *"every week"*.

In the scope of this thesis only expressions of the Date type are relevant, since the constructed timelines in this work are at least on a granularity level of days. In tasks where sub-day granularity is to be explored, of course, the Time type would be of additional interest.

[3] `https://github.com/HeidelTime/heideltime` ; accessed: 1. Oct., 2019
[4] `http://www.timeml.org/` ; accessed: 1. Oct., 2019

Besides from different types, and according to Schilder and Habel [54], and Strötgen and Gertz [65], a distinction between explicit, implicit, and relative expressions is possible:

1. Explicit expressions are of the form *"July 4, 2000"*, *"May 1998"* or *"4 o'clock"*. Hence, the exact time (or period of time) can be deduced directly, without any further knowledge of the context.

2. Implicit expressions need background information to be applied. Most often, they are names of holidays, like Christmas or Independence Day, or events, e.g., the Punic Wars. When encountering a word as *"Christmas"*, the expression likely refers to *"December 25"*. However, there are also more complex examples like Memorial Day in the U.S., which takes place on the last Monday in May. Without any context or knowledge of the year, the exact temporal information cannot be deduced from such an expression. It should be needless to say that detection of such implicit expressions highly depends on the knowledge base that is used to extract them.

3. Relative expressions rely on context information. The specific date of *"today"*, *"last Monday"* or *"January 1"* cannot be deduced without any reference date. This can be the document creation time, or temporal information in the vicinity of the expression. For example, in a sentence like *"John F. Kennedy was assassinated 1963. In January of the following year..."*, we can assume that the referenced *"January"* is the one of the year 1964.

HeidelTime employs a rule-based algorithm that makes use of explicit, implicit as well as relative expressions to extract temporal information.

Normalization

Additional to the identification of temporal expressions in textual data, it is important to have an annotation standard, such that different expressions referring to the same date or period are tagged the same. For example, it would be beneficial that *"Christmas 1994"* and *"December 25, 1994"* are transformed into the same tag. As mentioned previously, HeidelTime employs the TimeML format, which normalizes dates with respect to the ISO 8601 standard [75], i.e., *"Christmas 1994"* would be normalized as *"1994-12-25"*. Generally, the ISO 8601 representation of a date is *YYYY-MM-DD*, and a specific date is indicated as *value* attribute in the corresponding TIMEX3 tag. Days and months can be omitted to obtain dates of coarser granularity, e.g., *YYYY* represents a tag of year granularity. Tags can also refer to a specific week by employing the ISO 8601 representation *YYYY-Wnn*, where *nn* is the number of the week, e.g. *"1990-W20"*, with the first week of a year being the one

that has the first Thursday of the year. Additionally, the TIMEX3 tags used by TimeML also employ normalization for different cases, and store further information, however, those are not discussed here, since they are beyond the scope of this thesis.

Summary

HeidelTime utilizes the different forms of expressions discussed above to extract temporal information from documents. It then provides its results in the form of the TimeML standard, i.e., we retrieve our annotations with respect to the ISO 8601 format. For an even more thorough introduction, we refer to Strötgen and Gertz [66, 67]. Finally, we also provide our own Python wrapper, which is described in more detail in Section 4.1.

2.5 Related Work

Temporal information is of crucial importance in many fields of study related to information extraction, e.g. question answering [11, 22, 53, 55] or summarization tasks [1, 25]. Furthermore, temporal expressions are highly relevant for search engine development, since they are employed frequently in search queries as discussed by Nunes et al. [48]. Manica et al. [36] gives an introduction how web search engines employ temporal information, as well as providing possible improvements to the present state. Jatowt et al. [21] aim to extract the time a document refers to, hence, distinguishing between atemporal and temporal documents, and putting documents into a temporal context. In the following, we give an overview of three research areas relevant to this thesis: Firstly, the clustering of time-based data, which enables us to summarize temporal information; secondly, co-occurrence networks, which can be employed to describe the relation of entities and terms in a document collection, and which are not limited to the study of temporal data; and thirdly, the extraction of timelines, one of the central topics of this thesis.

2.5.1 Clustering of Time-Based Information

For many information extraction tasks, clustering is a crucial step to yield meaningful insights. One key field of research is the clustering of documents [50] with the goal of grouping results of search engine results [13, 52]. This can immensely improve user experience, since such a grouping can reduce the number of items the user needs to go through from hundreds to only a handful. In an early study by Aula et al., users emphasized that clustering, or categorizing, information is a useful tool for getting an overview of a topic [8]. In such a scenario, the user can get category suggestions, which is especially beneficial for exploratory tasks in cases where the user is unfamiliar with the topic, and categorization can uncover

documents ranked low by the ranking algorithm. A well-known metasearch engine is Yippy[5], originally known as Vivisimo and later as Clusty, which aims to integrate clustering into its search engine. Koshman et al. conducted a study about the Vivisimo user habits, evaluating the usefulness of clustering in this context [29]. While Yippy does not incorporate temporal aspects, besides from document creation times of web pages, into their search engine, Alonso and Gertz [2] employed temporal attributes to cluster search results. Therefore, they utilized metadata as well as temporal information they extracted using named-entity recognition, and thus, exploiting time-based expressions in the documents itself. In this approach, the clusters are represented by the respective temporal named entity, which can then be arranged as a timeline. Recent research by Lange et al. shows that temporal features can be effectively employed to classify news articles using k-Nearest-Neighbours and Decision Trees [32]. However, they subdivided the data set only into three categories for the German data, and four categories for the English data set, limiting to one domain of data, hence, leaving room for improvement. Since time-based clusters that are arranged along a timeline are a widely used strategy for timeline reconstruction, both topics are largely interconnected [6]. Typical use cases for timeline extraction and their benefits are discussed later.

2.5.2 Co-Occurrence Networks

Co-occurrence networks are nowadays one of the most important concepts in information extraction, enabling us to model relationships between entities in many fields of study: One of the first fields picking up co-occurrence networks was sociology to describe social structures [39, 56]; in biology, they are employed to research microbial interactions [10], and a magnitude of different cases [23]; and in pharmacology, they are utilized for drug discovery [20]. Though, this list is not exhaustive, and use cases can be found in various areas of research. One of the first theoretical works considering co-occurrences for information extraction tasks is by Van Rijsbergen [72], but since then research gained a lot more insight. More recently, Evert [15] researched the value of co-occurrences in computational linguistics, and hence, in the context of textual data, giving an extensive introduction to the topic. Utilizing co-occurrences extracted from textual data, Spitz and Gertz [59] established an entity-centric network model, employing a multipartite graph in which each partition represents its own class of entities. Since they distinguish between locations L, organizations O, actors A, and dates D, they coined it *LOAD graph model*. Building up from this model, they further proposed the exploration tool *EVELIN* that provides an interface for exploring entities and their relations [62]. Even more recently, they also utilized the model for topic exploration [60]. Spitz discussed the applicability of this and similar models to a variety

[5] `www.yippy.com` ; accessed 5. Jan., 2020

of information extraction tasks in much detail, even providing an extension to hypergraph models [58]. With the recent development of the internet, various new data sources emerged that were analyzed with the help of the models described above. One prominent example for such data is Wikipedia which was among other things utilized by Geiß et al. to extract a social network [16].

2.5.3 Timeline Extraction

Extraction and construction of timelines allow for an efficient summarization of complex information, utilizing our visual senses to transport as much information as possible in only a short amount of time. Timelines are helpful in many areas: Medicine and hospitals can profit from patient's timelines to get an overview of their medical histories [17, 46], whereas most clinical records come in the form of narratives written in natural language [68]; naturally, news reports can be utilized to extract events, and construct timelines [41, 71], summarizing and analysing for example political news stories [9]; Knight et al. [26] consider temporal information in a legal context, and Lagos et al. [30] discuss the value of timelines for legal case building, however, not much research about time-based data has been done in the legal domain yet. Social media also offers a wide variety of sources from which timelines can be constructed, e.g., Li and Cardie [34] generated timelines for specific twitter users, and Alonso et al. [7] used twitter hashtags to construct timelines for certain events, as well as data from twitter containing the hashtag *worldcup* to generate timelines for sport events [3]. Another creative way to use data can be seen in the work of Tran et al. [69], who utilize relevant newspaper headlines to reconstruct timelines for long-lasting news stories, significantly limiting the amount of text to be processed. They considered a headline to be relevant if it is informing, influential (with regard to future events), and if they spread, i.e., there are many headlines referring to the same event. A further important aspect of timelines is the selection of meaningful dates: A timeline that uses all timestamps present in a data set likely leads to cluttered visualizations and fewer insights. The necessity to prevent such cluttering can also be observed when looking at manually created timelines which are usually limited to the most important dates for the event or period. This selection is a large field of research in itself. One approach was proposed by Kessler et al. [24] who used supervised machine learning techniques, while Tran et al. [70] proposed a similar model using unsupervised methods. Building up from this, timeline summarization techniques combine this automatic identification of key dates with short descriptions of the respective date, s.t. they yield a set of dates associated with human readable sentences. Recently, Steen and Markert [63] introduced an abstractive timeline summarization model that computes timelines completely unsupervised using multi-sentence-compression, comparing their method with existing ones

and conducting a user study. This timeline representation as well as most of the other work presented above, aims at timeline summarization, i.e., to automatically provide a timeline to give an overview of a topic or period. However, the work presented in this thesis focuses on exploratory applications for time-based data and timelines in particular, and most of the work above is not suitable for exploratory tasks. In this context, Alonso et al. [4] present a timeline visualization for the exploration of search results, but to the best of our knowledge there are no recent advances in this field. Probably most similar to our work is the model of Spitz et al. [61] who provide a weighted bipartite graph model which is partitioned into dates and other (non-date) terms. Since this model is highly influential for our work, it is discussed in more depth in the following chapter.

3 Time-Centric Exploration

In this chapter, the model underlying our approach is presented. Firstly, an appropriate representation of document collections is introduced. Secondly, the concept of time and timelines is discussed, and the respective implications for this thesis are outlined. Thirdly, different time-centric graph representations are presented. In this context, a focus is placed on time-centric co-occurrence extraction, and time-centric co-occurrence graphs, which are the core of the proposed model, are introduced. Afterwards follows a discussion in which ways time-centric co-occurrence graphs can be employed for time-centric exploration, as well as an introduction to tf-itf, which is a metric based on term frequencies that scores the relevance of terms in regard to a point in time. Finally, it is discussed in which ways the proposed model can be evaluated which builds up to the work in Chapter 4.

3.1 Problem Setting

Goal of this thesis is to provide methods that enable users to organize and explore document collections in a time-centric fashion. To achieve this, techniques from graph theory as well as natural language processing, e.g., named entity recognition, are employed to construct flexible time-based representations that allow for the dynamic exploration of content. One key to this are co-occurrence graphs, which are utilized to build representations based on dates found in the processed data set with the help of HeidelTime. The most important questions discussed in this chapter are:

- How can time-centric data be represented using graph structures?
- In which ways can time-centric graph structures be employed for exploratory tasks?
- How can results produced by the proposed method be evaluated?

Furthermore, different granularity levels, and their utilization for time-centric exploration, are discussed, as well as different weighting schemes to construct more meaningful graph representations. Finally, means are provided to construct timelines using the proposed model.

3.2 Document Collections

In this section, a brief formal introduction of the data sources employed in this thesis is given. Thereby, the definition is based on a model that was described by Spitz and Gertz [59]. Firstly, a *document collection* $\mathcal{D}$ is defined as a set of text data that can be subdivided into documents $d_i \in \mathcal{D}, i \in \mathbb{N}$. Each document d further consists of a set of sentences $s_i \in d, i \in \mathbb{N}$, and we denote the set of all sentences with

$$\mathcal{S} = \{s \in d \mid d \in \mathcal{D}\}. \tag{3.1}$$

Furthermore, each sentence s consists of a set of terms t with

$$\mathcal{T} = \{t \in s \mid s \in \mathcal{S}\} \tag{3.2}$$

being the set of all occurring terms. While a term is often a single word, it is often specified even further, and can be among other things

- A named entity,
- A temporal expression: If it is a date, it is often called a timestamp,
- An expression consisting of multiple words,
- A marker to indicate a stop word,
- Part-of-speech tags, e.g., *Noun*.

While two or more sentences can have identical content, this model treats them as separate, and hence, every sentence can only occur once. Therefore, each sentence is identified by a unique natural number in regard to the document it is extracted from, i.e., each sentence of a document has a unique identifier. The identifiers are ordered, consecutive and monotone increasing, s.t. the first sentence of a document is assigned 1, the second 2, and so on. Since this ordering allows for determining neighbouring sentences for a given sentence, this is also helpful for the later extraction of co-occurrences.

For exploratory tasks, however, it is crucial to not only determine a representation that divides documents into sentences and words, but also links words to their respective sentences, and sentences to their respective documents. Consider the case where we have identified that the terms *"Angela Merkel"* and *"Barack Obama"* co-occur unusually often. In an exploratory environment, it is desirable to not only know that they frequently appeared in the same context, but *what* that context was about. Hence, a functionality that determines and presents

sentences to the user, in which both entities co-occurred, is highly desirable to improve the understanding of a document collection. For this purpose, the function $\theta : \mathcal{S} \rightarrow \mathcal{D} \times \mathbb{N}$ is defined as follows:

$$\theta(s) = (d, n) \text{ with } s \in \mathcal{S}, d \in \mathcal{D}, n \in \mathbb{N}. \tag{3.3}$$

Hereby, n denotes the index of the sentence in regard to the document. Thus, θ maps each sentence to its respective document d as well as to its associated unique identifier n. Accordingly, $\phi : \mathcal{T} \rightarrow \mathcal{S}$ defines a mapping that assigns to each term t its corresponding sentence s:

$$\phi(t) = s \text{ with } t \in \mathcal{T}, s \in \mathcal{S}. \tag{3.4}$$

With the help of θ and ϕ, we have the means to not only make a connection from a document to its associated sentences and terms, but also the other way around, i.e., given a term, we can go back to its origin in the document.

3.3 Time and Timelines

This section gives an overview about times and timelines. Firstly, the concept of time is examined more closely, and then some important vocabulary is established. Secondly, a base timelines is presented, and the meaning of the term *event* discussed.

Time

Since this thesis evolves around time, a concept of time needs to be established beforehand. While the exact definition of time itself is a topic for debate, we stick with the physical explanation that time is *"what a clock reads"*. Time is a so-called base quantity, it cannot be defined with the help of other physical quantities, and hence, is described by its own SI unit, the second. While clocks provide a fitting approach to measure the progression of time, they are not always a good mean to specify the time or duration of events. Therefore, other intervals of time were defined, like months and years, and a fixed origin from when we start counting was agreed upon: AD 1. For this thesis, from now on a day is chosen as the atomic time interval, and thus all occurrences of lower granularity are ignored. One consequence of this arrangement is that all events linked to a single point in time have to occur in an interval of time, i.e., in a day, month, year, etc. While granularity of time is a highly complex subject of research itself [14], we stick to a simple scheme in which we address days, weeks, months and years only. Depending on the frame of reference, the smallest addressable unit

is called a *chronon*, i.e., in a system in which only weeks, but no smaller granularities, can be addressed, the chronon is a week. For example, in case of the prevalent Gregorian calendar, the chronon is usually a day, but also coarser representations, in which only weeks, months or years are expressed, are valid under this definition.

Timelines

A classical representation, that links a number of events to points in time, is a so-called *timeline*. A timeline displays events in a chronological order associating them with a certain point in time, usually arranged from left to right, or top to bottom, or vice versa. A *base timeline* is defined as an interval of consecutive day chronons, and is denoted as T_d [5]. This means, the sequence *"1. January 2019"*, *"2. January 2019"*, *"3. January 2019"*, would be a valid base timeline. A single instance in such a timeline, e.g., *"1. January 2019"*, is also called a *timestamp* or *date*. Advancing from this base timeline, coarser timelines can be defined by grouping subsequences of the base timeline, and thus, yielding a new consecutive sequence. For example, days can be grouped into years receiving a sequence like *"1990"*, *"1991"*, *"1992"*. By choice of different base chronon and grouping, diverse timelines can be constructed, each one offering other opportunities for analysis. Figure 3.1 shows an example of a timeline in which the base chronon is a minute, and the events represent goals of a soccer match. The coloured bars above the timeline show time periods that indicate which team was in the lead at the respective point in time. The concept of time periods is discussed in the following section.

Time Granularities

In practice, events can refer to different time granularities on the same timeline. This can be solved by breaking down coarser granularities to finer ones and representing them as an interval, e.g., *"October 2000"* can be expressed as *"1. Oct. 2000 - 31. Oct. 2000"*. While it is beneficial to display such intervals on a timeline, it is tedious to always denote them this way and we abbreviate such an expression with *"Oct. 2000"* or a similar term. We employ the Gregorian calendar in the context of this thesis, and organise dates in four groups: Year, months, weeks and days. Utilizing these four different granularities, an inclusion hierarchy can be formulated, in which for each day exists a week and a month in which it is included, and the same relation holds between months and years. Furthermore, the set of dates D is partitioned into $D = D_y \cup D_m \cup D_w \cup D_d$, where the indices denote years, months, weeks, and days respectively. Hereby, weeks refer explicitly to weeks defined by the ISO 8601 standard [75]. The hierarchy formulated above implies for an entry in the set D_d that the

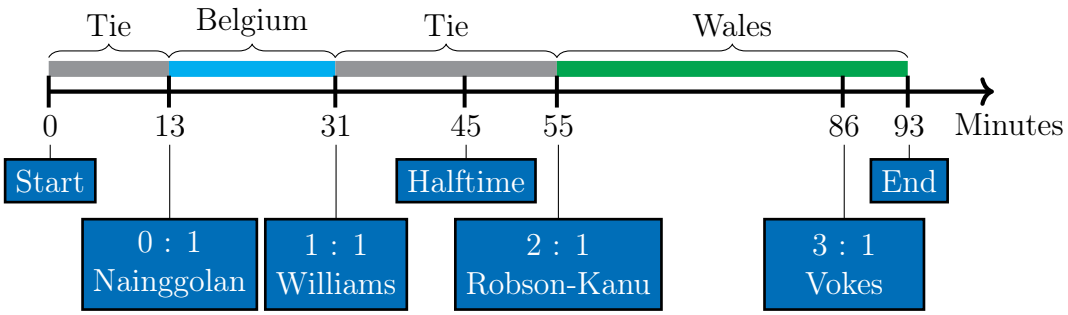


Figure 3.1: Example timeline that depicts goals as well as the current leading team of the match Wales vs. Belgium during the soccer world championship 2016.[1]

respective coarser element also to be included in the set D_w, D_m and D_y. The same holds for D_m in regard to D_y. Consequently, if *"3. Oct. 2018"* is contained in D_d, *"2018 - Week 40"* is included in D_w, *"Oct. 2018"* in D_m, and *"2018"* in D_y. It should be noted that there is no inclusion hierarchy between weeks and months, or between weeks and years, since a week can potentially span over more than one year, and hence, can be divided between multiple months or years.

Events

Until now, the expression *event* appeared quite often. But what exactly is an event? Attempts to properly define the expression yield various results, as for example shown by Navas-Loro and Santos [44] in the legal domain, but the most simple one is *"something that happens"*[2]. While this assumption is sufficient if a timeline is constructed manually, where suitable items are picked by hand, for automated creation of timelines, this definition of events is not sufficient. Instead of associating certain points on a timeline with such vague events, they can also relate with other representations, e.g., named entities, or networks. This has the benefit that entities and networks can be extracted from textual data, e.g., by constructing co-occurrence networks related to a point in time placed on the timeline. The next sections formalize this approach, and thus, lead to a timeline representation that is suitable for combination with automated text processing.

[1] `https://en.wikipedia.org/wiki/UEFA_Euro_2016_knockout_phase` ; accessed: 12. Oct., 2019
[2] `https://www.merriam-webster.com/dictionary/event` ; accessed: 11. Oct., 2019

3.4 Time-Centric Graph Representations

Since graphs are a natural tool to analyze and visualize the relations between terms, in this section suitable graph representations, that include time as a core component, are presented. Thereby, different properties that can be linked to time are discussed. Underlying for all these models are bipartite graphs as introduced in Section 2.1.3. Additionally, possible applications for the individual models are presented. In this section, it is assumed that suitable co-occurrences have already been obtained from the given data by employing a window based approach similar to the one described in Section 2.3. Later we get back to the subject of extracting co-occurrences for time-centric graphs. The multiset of co-occurrences is denoted as $\mathcal{C}$ and contains unordered tuples (s, t) each representing a single co-occurrence.

3.4.1 Time-Term Model

The *time-term model*, introduced by Spitz et al. [61], partitions the set of all terms $\mathcal{T}$, as defined in Equation (3.2), into a set of dates D, and a set T which includes everything that is not classified as a date. In this context, we will simply refer to T as terms or words, while strictly speaking dates are also terms according to our definition of terms. Given this partitioning, a weighted bipartite graph G is defined as $G = (D, T, L)$ with $L \subseteq D \times T$. An edge $l = (d, t)$ exists in L if the date d and the term t co-occur, i.e., there is a tuple in $\mathcal{C}$ that contains d and t. With regard to the hierarchy of different time granularities, see Section 3.3, it is implied that if an edge exists between a date and a term, there also have to be edges between the term and the associated dates of coarser granularity. The weighting function $w : L \to \mathbb{N}$ can be defined depending on the use case, e.g., a common function would be to weight the edges by the total number of co-occurrences between the two terms, i.e.,

$$w(d, t) = |\{(a, b) : (a, b) = (d, t) \wedge (a, b) \in C\}|. \tag{3.5}$$

It should be noted that the expression inside the cardinality is again a multiset, and therefore, w can evaluate to any natural number. Also the order of (a, b) is irrelevant since the elements of C are unordered. However, various different weighting functions can be utilized as well, e.g., the weighting can be scaled logarithmically. In cases where a constant function $w(d, t) = 1$ is employed, a non-weighted graph is obtained.

As the name indicates the time-term model associates temporal expressions with terms, and thus, enables to structure terms in regard to points in time they are associated with. But not only is it possible to identify the most prevalent points in time for a term, one can also find the terms that are most associated with a temporal expression. Especially, the

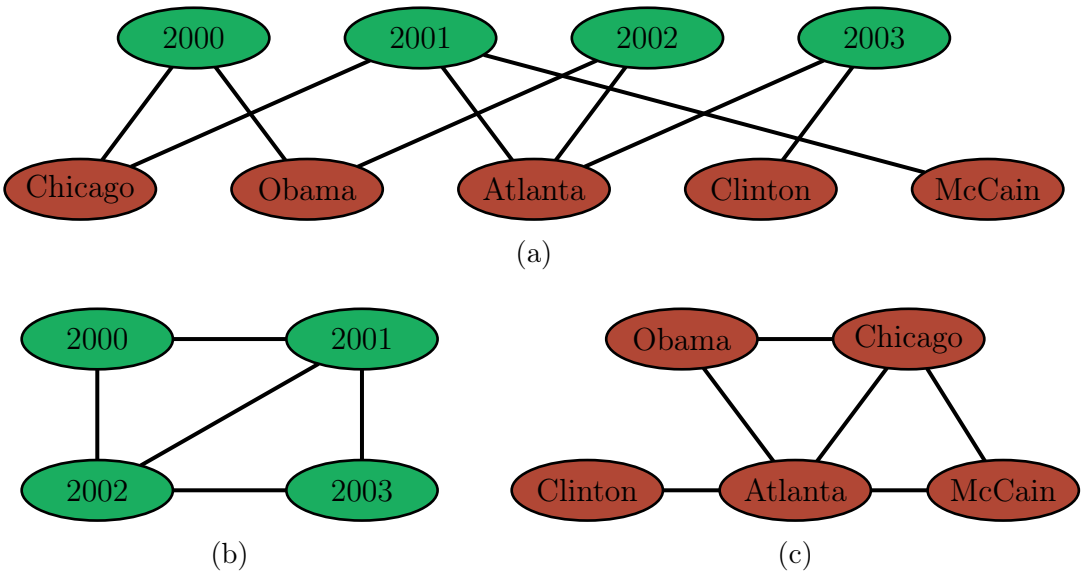


Figure 3.2: (a) An example graph G representing the time-term model with given green dates, and red terms. One mode projection of G onto (b) the time component, and (c) the term component.

second use case is highly beneficial for timeline construction. For example, a timeline could list the most important terms for every timestamp, where the displayed terms are those that are connected by a link to the temporal expression and have the largest edge weights. Employing the right weighting function for a certain task can then yield meaningful results, and give new insights into the data. A potential drawback is the large number of terms that is in practice often related to one point in time. Representing each timestamp by too many terms yields a cluttered visualization, which prevents meaningful interpretation of the outcome. Taking too few terms may reject too many substantial results, and hence, critical conclusions might be complicated. Furthermore, one-mode projection can be employed, as introduced in Section 2.1.3, to create two different interesting networks:

1. Projecting a time-term graph onto its time component yields a network that connects all temporal expressions that share certain terms, i.e., they potentially share a common concept. For example, when analysing a news data set, it could be expected to find a cluster of timestamps that are all linked to election days. On the other hand, certain terms may be so prevalent in news media that they would link basically all temporal expressions to each other. Imagine taking *"Donald Trump"* whose name occurs a lot on the news every day. Utilizing a trivial weighting function may connect all temporal expressions; however, one can also employ a weighting scheme like tf-idf, see Section 2.3.3, to weaken the influence of such omnipresent terms, and concentrate on those expressions that are specific to certain dates, which are then indicated by large edge weights.

2. Secondly, projecting a time-term graph onto its term component generates a network in which terms are connected that co-occur to the same temporal expression. This potentially extracts events that took place at the same time in respect to some granularity, i.e., either on the same day, the same month, or some other specified time period. Given a political news data set for example, we could then obtain topics and events that are connected to the same period of time. However, as with the projection of the graph onto its time component, certain dates may link so many terms that the resulting graph has more links than desired, and thus, a suitable weighting function has to be chosen to indicate and rank the strength of edges.

A straight-forward example for these one-mode projections is given in Figure 3.2. In conclusion, the time-term model separates the concept of time from any other concept in a data set, and allows for an analysis of both points in time and terms given in a document collection in regard to each other. Furthermore, this representation can serve as origin for a simple timeline construction that can give meaningful insight into textual data.

3.4.2 Time-Entity Model

Instead of considering all terms in the corpus, one can also limit the scope to certain entities, and adapt the model above to include only these entities, yielding a special case of the time-term model, which is called *time-entity model*. An entity in this context can be one of multiple classes, e.g., a location, an organization, or a person. Formally, the time-entity model partitions the set of all terms $\mathcal{T}$ into a set of dates D, and a set of entities E. E consists of all desired entity types, which may be only one type of entity like all persons, but it can also include multiple different types of named entities. Upon this partitioning a bipartite graph $G = (D, E, L)$ is built, where $L \subseteq D \times E$, and a link $l = (d, e)$ exists in L if the date d and entity e co-occur in the document collection. As with the time-term model there exists a hierarchy of time granularities, and we define a weighting function w to weight edges between points in time and entities.

The time-entity model relates points in time and certain entities. It allows a tailor-made construction of a network that only contains those concepts a user is interested in, and thus, ignores terms that may be prominent but not meaningful for a specific analysis task. In the context of timeline construction this opens up new possibilities:

- Firstly, timelines can be constructed that limit themselves to certain entity classes. For example, one might only be interested in investigating which companies are mentioned at different points in time to examine a given hypothesis.

- Secondly, it is possible to select only those entities a user is specifically interested in, and indicate on a timeline at which times they are mentioned. This way, patterns may be observed which are shared among various entities, finding substantial similarities and differences as a result.
- Thirdly, by limiting the scope of the model to a subset of entities, the aforementioned problem of cluttering is reduced as well. As a result, this model potentially leads to cleaner and more comprehensible visualizations.

Additionally, a one-mode projection can be applied to the graph:

1. A projection of the time-entity graph onto its time component yields, similarly to the time-term model, a network that connects all temporal expressions that share a co-occurrence with a named entity.
2. Projecting the time-entity graph onto its entities links those entities to each other that both co-occur with the same point in time. Thus, those entities that are possibly linked to each other can be isolated, and points in time they are both involved can be identified to further analyze specific relationships.

Both of these projections share the same problems as discussed in Section 3.4.1 for the time-term model, and hence, for the time-entity model it is also crucial to choose the right weighting function to obtain meaningful results. An advantage over the previous model is that employing a one-mode projection on a time-entity bipartite graph usually leads to sparser graphs than it is the case for the general model, which simplifies manual analysis in exploratory scenarios.

Time-Time Model

The *time-time model* is a special case of the time-entity model, where we restrict the selection of entities to temporal entities. This results in a partitioning of $\mathcal{T}$ into two sets of dates D_1 and D_2, which are a duplicate of each other, and the corresponding bipartite graph $G = (D_1, D_2, L)$, where L indicates links between D_1 and D_2 as described above. Since the time-time model represents co-occurrences between temporal expressions, D_1 and D_2 are identical, and the graph is symmetrical. Nonetheless, by one-mode projection it is possible to analyze the relation between points in time. For example,*community detection* can be employed to find cliques of expressions that are closely linked to each other. However, for timeline construction this model is probably one of the less suitable ones.

3.4.3 Time-Entity-Term Model

In contrast to the models presented before, the *time-entity-term model* does not only divide the set of terms $\mathcal{T}$ in two partitions, but in three. The resulting sets are dates D, selected named entities E and terms T. T includes everything that can neither be inserted into D nor E. Formally, a weighted tripartite graph $G = (D, E, T, L)$ is defined with $L = (D \times E) \cup (D \times T) \cup (E \times T)$. Equally to the bipartite models, an edge between a date and entity, a date and term, and an entity and a term exists if the respective two expressions co-occur in the document collection.

The time-entity-term model combines both the time-entity model and the time-term model, however, it is more than the sum of its parts. While it can accomplish similar tasks by ignoring either the entity or term set, the additional links between terms and entities provides further information. To allow for a better analysis, the definition of one-mode projection, see Section 2.1.3, has to be expanded to multipartite graphs:

Definition 13 (One-Mode Projection for Multipartite Graphs) *Given a multipartite graph $G = (\mathcal{N}, L)$ with $\mathcal{N} = N_1 \cup ... \cup N_m$, the one-mode projection of G onto N_i yields a graph $G' = (N_i, L')$ with*

$$L' = \{(n_1, n_2) : n_1 \in N_i \wedge n_2 \in N_i \wedge \\ (\forall j : \exists v : v \in N_j \wedge (n_1, v) \in L \wedge (n_2, v) \in L,\ 1 \leq j \leq m, j \neq i)\}. \tag{3.6}$$

Hence, such a one-mode projection yields a graph with vertices from one subset that have a link if they share a common node in *all other subsets* of G. Figure 3.3 illustrates one-mode projection for an example multipartite graph. Additionally, it should be noted that this is a rather strict definition, and for certain use cases it may be beneficial to require only a common node in certain different subsets, however, a formal definition is omitted here.

Applying the one-mode projection then allows to identify temporal expressions that do share multiple common entities or terms, and thus, even more refined relations can be found. For example, instead of taking all terms, a different sets of entities can be chosen, and a tripartite *time-person-location* graph can be constructed. However, there is an enormous downside to such representations: While it might be known that a point in time co-occurs with a person, and that person with a location that then again co-occurs with the initial point in time, such that they form a triangle, this does not mean that all three appear in the same context, since all three co-occurrences can be completely independent from each other. As a result, one-mode projection of such a model can only indicate that a person was at a certain place at a certain time, but such an assumption always needs to be verified by having

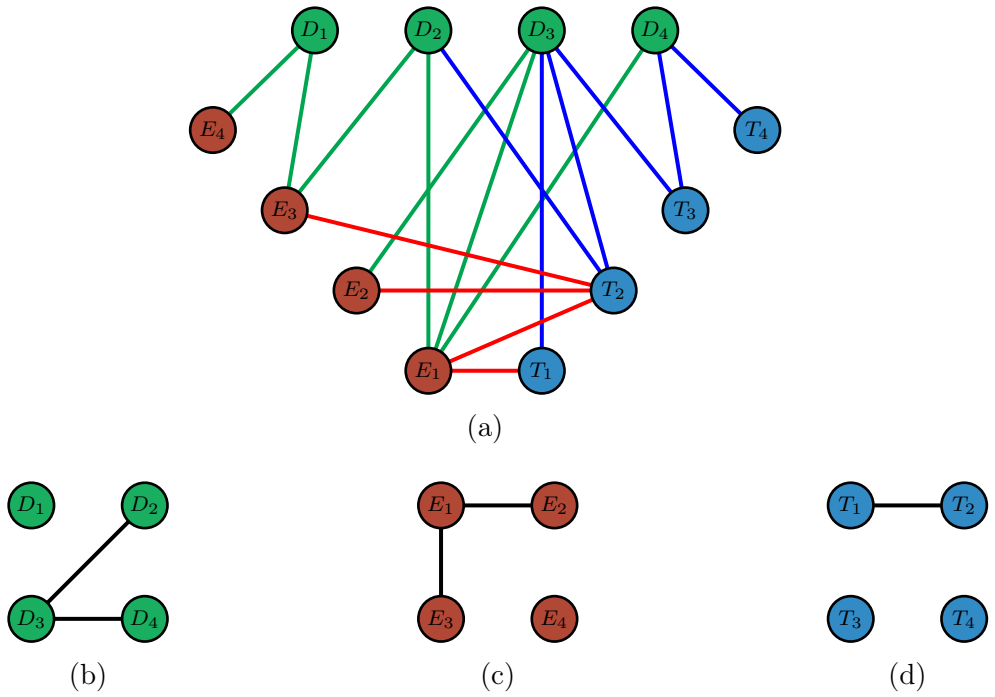


Figure 3.3: (a) A multipartite graph G with partitions D (green), E (red), and T (blue). Edge colors are only a visual help to distinguish between which two sets the edge spans. One mode projection of G onto (b) partition D, (c) partition E, and (d) partition T.

a second look at the data itself afterwards. A solution to this could be the introduction of multipartite hypergraphs that only introduce an edge if all three expressions co-occur in the same context. However, this is beyond the scope of this thesis, since we focus on other approaches to extract such information. Finally, it should be noted that the time-entity-term can be expanded, s.t. even more different partitions are constituted as for example proposed by Spitz and Gertz [59]. However, with increasing complexity of the model, the interpretation of results becomes more and more complicated and uncertain.

3.5 Time-Centric Co-Occurrence Graphs

The previous models all provide methods that relate points in time with terms they co-occur with. However, they have one substantial disadvantage: Oftentimes, we are not only interested in what is mentioned at a certain time, but we are interested in the relations between terms around this time. In an earlier example, it was tried to bring time, persons and locations together, but by employing a dyadic graph this is not a simple task. Therefore, here another model is introduced that maps a co-occurrence graph to each point in time, resulting in a collection of networks. To achieve this, time-centric co-occurrence extraction is discussed, upon which a collection of time-centric co-occurrence graphs is built.

3.5.1 Time-Centric Co-Occurrence Extraction

In the previous section, it was assumed that a set of co-occurrences extracted by a window based approach is already given. However, for the following model, a different co-occurrence extraction method has to be specified, since it differs crucially from common approaches. Instead of going over the whole document to derive co-occurrences, first temporal expressions need to be identified in the text as described in Section 2.4. As a result, a set of timestamps Ω is obtained, for which the temporal information they carry as well as the locations of their instances in the document are known. A timestamp (or date) $\omega \in \Omega$ is hereby some kind of temporal information, or temporal entity, that is extracted in a previous step, e.g., in this thesis with the help of HeidelTime. With this prerequisite given, a simple algorithm to extract co-occurrences associated with each timestamp $\omega \in \Omega$ selects pairs of every two words in a window around the timestamp, ignoring self-references. Pseudocode for this algorithm is given in Figure 3.4. A mapping $\Psi : \Omega \to \mathcal{C}$ is then defined with $\Psi(\omega) = c_\omega$ where $c_\omega \in \mathcal{C}$ is a multiset of co-occurrences around timestamp $\omega \in \Omega$ with $c = (s, t) \in c_\omega$ being an unordered tuple given by the co-occurrence of term s and t. Note that the timestamp itself co-occurs with all other expressions contained in tuples of c_ω. Of course, the time-centric co-occurrence extraction algorithm can be altered and more refined, e.g.,

- Instead of selecting all pairs of words around a given date t as co-occurrences associated with t, a second window can be introduced around each word. Only words that co-occur in this context are then used to extract co-occurrences. This new window can have a different width than the one around the timestamp itself.
- Following up on the first point: *"Should one also consider in this new context terms that are beyond the initial window around the timestamp?"*.
- Taking into account the distance between two words for weighting, and hence, putting more emphasis on words that are closer to each other.
- Including a weighting that considers the distance from the timestamp itself is also beneficial in some information extraction tasks.
- Excluding co-occurrences of the timestamp and expressions around it, since trivially all words in the co-occurrence set c_ω also have to co-occur with their associated timestamp ω.

The choice of the exact algorithm hugely depends on the actual use case. In the context of this thesis, the algorithm presented in Figure 3.4 is employed unless stated otherwise.

Algorithm 1 Extract Co-Occurrences around Timestamps

```
1: function COOCS_AROUND_TIMESTAMP(timestamp, document, window_size)
2:     resulting_cooccurrences ← new list
3:     start_idx ← max(0, timestamp.sentence_id − window_size)
4:     end_idx ← min(len(document), timestamp.sentence_id + window_size)
5:     for i ∈ {start_idx, ..., end_idx)} do
6:         for j ∈ {start_idx, ..., end_idx)} do
7:             for word_i ∈ document.sentences[i] do
8:                 for word_j ∈ document.sentences[j] do
9:                     resulting_cooccurrences.append(word_i, word_j)
10:    return resulting_cooccurrences

11: for each timestamp do
12:    COOCS_AROUND_TIMESTAMP(timestamp, document, window_size)
```

Figure 3.4: Pseudocode for the extraction of co-occurrences around timestamps.

3.5.2 Time-Centric Co-Occurrence Graph Collections

Now that ways to extract co-occurrences in respect to timestamps were discussed, it remains to build a meaningful representation of the data. To achieve this, extracted co-occurrences can be represented by using a graph structure. Therefore, *time-centric co-occurrence graphs* are defined:

Definition 14 (Time-Centric Co-Occurrence Graph) *Given a set of timestamps Ω, a **time-centric co-occurrence graph** is a weighted graph $G_\omega = (N_\omega, L_\omega)$ with nodes N_ω being the terms extracted in a window of x sentences around timestamp $\omega \in \Omega$, and links L_ω that represent the co-occurrences between them.*

By weighting G_ω a significance can be assigned to each edge, which is usually done by counting the number of co-occurrences between two terms, and adjusting the result by normalization or taking the logarithm.

The set of all G_ω is denoted by

$$G_\Omega = \{G_\omega \mid \omega \in \Omega\}. \tag{3.7}$$

It should be explicitly noted that for each timestamp $\omega \in \Omega$ only one time-centric co-occurrence graph G_ω exists. This means in particular that there is not a separate graph for each occurrence of ω, but co-occurrences around different instances of ω are aggregated in the same graph G_ω, yielding one graph representing the respective timestamp.

Figure 3.5 illustrates the construction of a time-centric co-occurrence graph given two sentences and a window size of $x = 0$, ignoring weights and considering only nouns for co-occurrence extraction. With the help of the edge colors, it can be easily observed that subgraphs constructed from one sentence have to be cliques, i.e., for the used co-occurrence extraction scheme, the induced subgraph is always complete. Some nodes (in the example given Obama and Atlanta) serve as a connector between different cliques due to their appearances in multiple context windows. The date itself is a special hub that is connected to each term present in the graph. This also implies that the graph always needs to be connected, and hence, there is only one connected component.

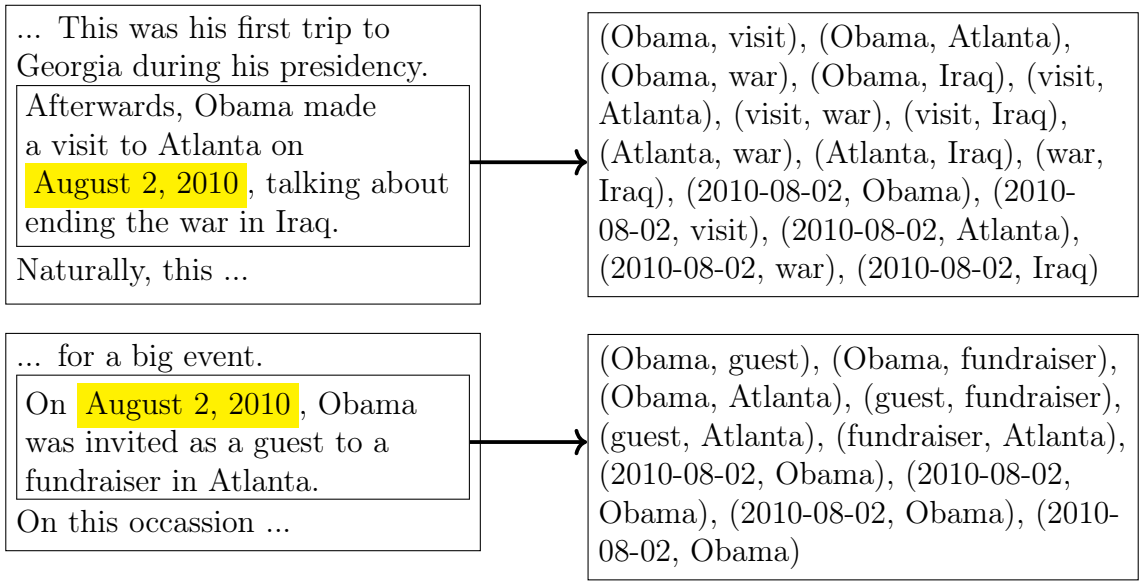
While a co-occurrence $c \in \mathcal{C}$ consists of tuples of terms where each term can be exactly associated with its sentence in the document collection as introduced in Section 3.2, a node $n \in N_\omega$ is only represented by its semantic information, i.e., in $\mathcal{C}$ there is a difference between two terms if they appear at different positions in the text, while expressions having identical spelling are merged in G_ω into one node. For temporal expressions, even different spellings are ignored, and timestamps are merged if they hold the same temporal information.

Furthermore, a mapping $\xi : \Omega \to G_\Omega$ is defined that maps each timestamp to its respective time-centric co-occurrence graph, s.t.

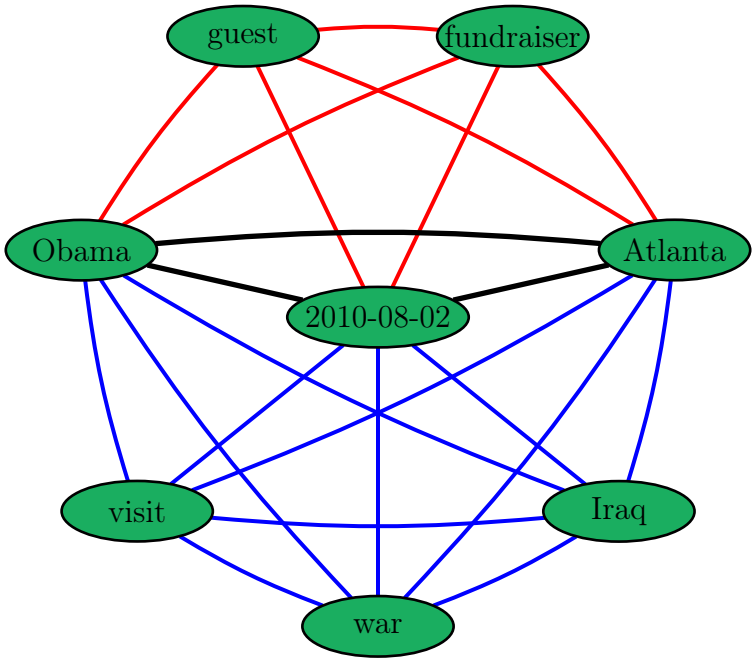
$$\xi(\omega) = G_\omega. \tag{3.8}$$

With the prerequisites given above, *time-centric co-occurrence graph collections* are defined:

Definition 15 (Time-Centric Co-Occurrence Graph Collection) *A **time-centric co-occurrence graph collection** is a triple (Ω, G_Ω, ξ) consisting of the temporal domain Ω, a set of co-occurrence graphs G_Ω associated with timestamps $\omega \in \Omega$, and the mapping ξ that maps each timestamp to its respective co-occurrence graph.*



(a)



(b)

Figure 3.5: (a) Extraction of co-occurrences around the timestamp August 2, 2010, using window size 0, and extracting only nouns and the date itself. (b) The resulting time-centric co-occurrence graph for August 2, 2010. Edge colors indicate whether the co-occurrence was extracted from sentence 1 (blue), sentence 2 (red), or from both (black).

A time-centric co-occurrence graph collection connects timestamps with information gathered around the respective temporal expressions belonging to the timestamp. This representation can be employed in various ways to extract information from document collections:

- Comparison of co-occurrence graphs of different timestamps. Network topology can vary from graph to graph, and networks with unique properties may be identified.
- Identification of entities that appear in many time-centric networks, and which are, thus, universal; and of entities that are specific to only a few temporal expressions.
- Timeline reconstruction and temporal exploration can highly profit from such a representation. The domain Ω of the mapping ξ is perfectly suitable for such a timeline construction. Since this is the focus of this thesis, this approach is discussed in detail in the following Section 3.6.

Additionally, a mapping **sent** is defined that assigns to each edge (n_i, n_j) the sentences from which it was created, i.e., in which two (not necessarily distinct) sentences the words n_i and n_j co-occurred. More formally: Given a graph $G_\omega = (N_\omega, L_\omega)$, the mapping $\mathbf{sent} : L_\omega \to \mathcal{P}(\mathcal{S} \times \mathcal{S})$ is defined by

$$\mathbf{sent}((n_i, n_j)) \coloneqq \{(\phi(a), \phi(b)) \mid (a, b) = (n_i, n_j) \wedge (a, b) \in \Psi(\omega)\}. \tag{3.9}$$

Note that n_i and n_j denote the respective node label, and not an internal node ID assigned to the node, (a, b) represent a tuple of terms occurring in the document collection, $\Psi(\omega)$ denotes the set of all co-occurrences extracted around timestamp ω, and ϕ maps a term to its the sentence it is occurring in. As a result, the function **sent** yields all tuples of sentences where the words n_i and n_j co-occurred, linking them to the actual terms a and b occurring in the documents. This is a crucial difference, since n_i and n_j are not directly associated to any words in the text, and hence, do not carry any positional information. As an example, consider the node n_e in G_ω having the label *Obama*: n_e is not directly associated with any term in the documents, however, there is at least one tuple in $\Psi(\omega)$ that contains *Obama*, and this occurrence can be linked to the sentence it was extracted from with the help of ϕ.

This closes the circle: Given a set of documents, (1) timestamps are extracted, (2) co-occurrences around these timestamps are obtained, (3) from these co-occurrences time-centric graphs are built, and (4) the contained edges then point to sentences in the document, from which they were originally extracted. Thus, this model enables to explore documents from a time-centric perspective, in which one can zoom in on parts of the text, that are associated with a certain point in time.

An advantage of this model over the previously discussed models in Section 3.4 is that an edge in a time-centric co-occurrence graph not only indicates that the two terms occurred in the neighbourhood of a timestamp ω, but that they were also near to each other, and co-occurred in the same context. This is surely a stronger proof for relatedness than applying one-mode projection onto a graph from the previous model. Moreover, it is possible to even get more insight into the origin of edges by employing the function **sent**, enabling users to explore the reasons *why* two terms are connected. Similar to the time-term model and its extensions, it is possible to restrict terms taken into account for co-occurrence extraction to a subset of all terms. For example, one could consider terms that are classified as some named entity, e.g., as a person or organization. This is also discussed in Section 4.1, where more insight into the implementational details of this thesis is given.

It should be noted that the presented model is highly suitable for storage in a relational database, s.t. the approach can be scaled as well. Using such an approach also allows for an easy insertion of new documents: Timestamps and edges can easily be included after initial creation. However, this will not be discussed in detail, since the design of relational databases is beyond the scope of this thesis.

3.6 Time-Centric Exploration

In this section follows a discussion about the construction of timelines and the exploratory scenarios that become possible by utilizing time-centric co-occurrence graphs as introduced above in Section 3.5.2. To construct a timeline, given a time-centric co-occurrence graph collection (Ω, G_Ω, ξ), timestamps $\omega \in \Omega$ can be arranged on a line with earlier points in time on the left or top, and more recent timestamps on the right or bottom as described in Section 3.3. However, now a co-occurrence graph is assigned to each timestamp ω, i.e., each point on the timeline links to a graph $G_\omega \in G_\Omega$, given by $\xi(\omega)$. Figure 3.6 visualizes such a timeline. In the following sections, possible use-cases and refinements of the proposed model are elaborated.

3.6.1 Finding Conflicting Data

To provide a first example of how timeline exploration using the proposed model can be employed, it is discussed how analysing a time-centric graph G_ω, given a timestamp ω, can facilitate fact checking and in which ways it enables the user to discover contradictory information in the data set. Consider the case, where a node labeled *Barack Obama* is contained in the graph G_ω, which is directly connected to *San Francisco* and *Atlanta* via

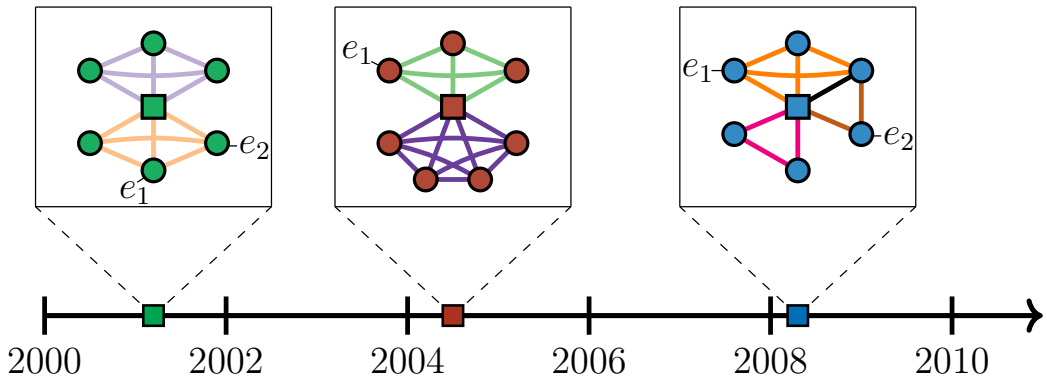


Figure 3.6: Timeline employing time-centric co-occurrence graphs for three different points in time (green, red, and blue rectangles). Each timestamp has a graph assigned that is visualized in its corresponding box indicated by matching node colour. In each graph the central node represents the respective date, the rest are arbitrary terms. Edge colours visualize cliques which are probably co-occurrences that were all extracted from the same context window.

edges e_1 and e_2. Such cases can have a reasonable explanation, e.g., Obama might have mentioned both cities in a speech given at the time represented by ω. However, the user might consider this to be unlikely: By utilizing the function **sent** (see Equation (3.9)), the user can directly investigate the origin of the relationship, inspecting the sentences where the respective terms co-occurred in the neighbourhood of ω. While this is not yet a suitable approach to scan a whole document collection for conflicting information, it can help to provide evidence for conflicts the user already suspects, saving a large amount of time that would be necessary if the user would need to search for such co-occurrences by reading the documents itself. A potential use case for this would be the detection of fake news by finding incidents where different articles contradict each other, a topic which became more and more relevant in the last years, see for example, Shu et al. [57], or Zhou and Zafarani [77] for an extensive introduction to the topic. Furthermore, possible applications are also found in the legal domain: Instead of reading through pages of court protocols, lawyers and judges can test assumptions linked to a temporal component by investigating a timeline, e.g., following up on a statement they think to be contradictory with previous testimonies. Certainly, timelines and time-centric co-occurrence graphs do not supersede the need to actually read the documents, however, they can hint the user to relevant parts of the text, saving her potentially hours of tedious work to scan the document herself.

3.6.2 Entity-Centric Timelines

Analogously to the time-entity model presented in Section 3.4.2, one can construct timelines that do not take into account all possible timestamps ω, but only those when the associated co-occurrence graph G_ω exhibits some desirable property. This characteristic can be highly complex, but for data exploration, usually a simple criterion, that also enables users to easily interpret the results, is beneficial. After all, the goal is to simplify document exploration, and not to add more layers of complexity to it. In the following, two straightforward criteria are discussed in more detail:

1. Requiring the graph G_ω to contain the node representing the entity itself, i.e., only timestamps are depicted that co-occur with the entity in question. This requirement can be expanded, s.t. multiple entities have to appear in G_ω, or the entity needs to occur a certain number of times in the vicinity of ω, employing a suitable weighting function.

2. Instead of looking for nodes representing a desired entity, it is also interesting to require a graph to contain a certain relationship between two entities, i.e., there needs to exist an edge between two nodes of G_ω for ω being incorporated in the timeline. One can also demand multiple existing edges as well as a minimal weighting of the edges.

In the following, we assume the case that only one node, one edge respectively, is required for ω being part of the timeline. Note that while this timeline construction is called entity-centric, of course, any expression can be chosen instead of an entity. Hence, it is also possible to think about this model as keyword-centric. The first criterion yields a timeline which illustrates points in time where an entity is involved in. Firstly, this means that one can easily build an at-a-glance visualization that depicts in which temporal context the entity was mentioned. While this is not yet an advantage to the time-entity model presented in Section 3.4.2, the exploration of the associated graphs is a substantial improvement in many cases. Not only can a user infer from the graph when the entity was mentioned, from the relations to other nodes a user also can possibly deduce why it was mentioned in exactly this context; and by employing the mapping **sent**, it is possible to dive even further into the details which enables to weigh up the relevance of the results. Hence, a user can analyze the relation between the entity and the timestamp with comparatively little effort, assuming that the graphs are already pre-computed. *Given a legal document collection, this offers opportunities to construct timelines that depict when a certain person was mentioned, or when a specific location was discussed.* Hence, instead of a tedious analysis of multiple documents, with the help of the function **sent**, it is trivial to determine if an accused person is related to a certain date, and to determine the context of this relation. Possession of such an extensive

timeline for a person can surely simplify and accelerate the work of many lawyers, judges and prosecutors. But even more, it can also give reasonable access to such data to a wider audience, and hence, constitute public interest.

The second criterion is essentially a specialization of the first one: By requiring an edge to be included in a graph, naturally both nodes involved have to be in the graph as well; but in this second case both entities have to be connected as well. Clearly, this is a stronger constraint than the first one, and, hence, by applying this criterion, it will always yield a subset of the timeline constructed by only requiring both nodes to be contained in G_ω. By such a construction, it is possible to analyze at which points in time two entities interacted with each other, or were involved in the same event. Again, the mapping **sent** enables users to get into more detail, and learn about the exact reason for their co-occurrence. Such an analysis is not directly possible using the time-entity model presented in Section 3.4.2, and hence, it expands the exploratory possibilities of the model.

An example for both cases can be constructed with the help of Figure 3.6: Establishing a timeline using the first criterion and requiring entity e_1 to be part in the networks, yields all three graphs as a result, since e_1 is part of all three networks. Demanding e_1 and e_2 to be part of the networks, results in the left (green nodes) and right (blue nodes) graph, while the middle (red nodes) graph is discarded, since it does not contain e_2. Utilizing the second criterion, and requiring an edge between e_1 and e_2 yields only the left graph as a result, since it is the only one in which both entities occur and are directly connected.

3.6.3 Zooming In and Out

The previous use cases all assumed a temporal domain Ω consisting of a variety of timestamps ω representing various granularities. However, as introduced in Section 3.3, it is often beneficial to exploit the partitioning of Ω into sets of dates of different granularities, usually days, months and years, between which an inclusion hierarchy was established. For time-centric co-occurrence graphs this hierarchy implies that a given graph associated with a finer granularity is always a subset of the time-centric co-occurrence graph representing the respective coarser granularity (as long as different node and edge weightings are ignored). An illustration of this principle is given in Figure 3.7.

Employing the inclusion hierarchy for timeline exploration enables users to explore content more dynamically. Displaying events of all granularities on the same timeline leads to cluttering, and subsequently makes analysis more difficult and tedious. Instead, it is usu-

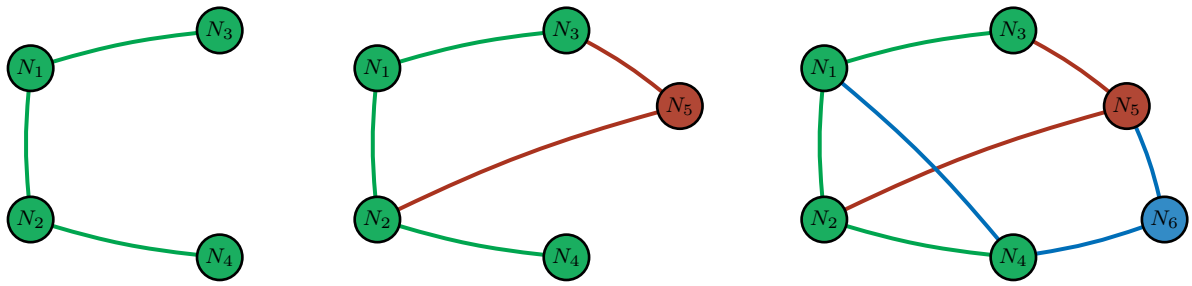


Figure 3.7: Time-centric co-occurrence graphs based on different temporal granularities, e.g., (a, green) *"1. January 2000"*, (b, red) *"January 2000"*, and (c, blue) *"2000"*. Coarser granularity can only add vertices and links to a graph, but cannot remove them, i.e., networks referring to the finer granularity are always subgraphs of networks corresponding to the respective coarser granularity.

ally beneficial to construct a timeline for each respective temporal granularity, and examine them separately. Analysis of a year-granularity based timeline may then reveal years that are especially interesting to the user. This granularity based approach then allows to inspect respective finer granularities, and examine different months and days to determine the exact reason for co-occurrences in the coarser structure. This exploration can be accelerated if information available from entity-centric timelines for finer granularities is simultaneously utilized, since this way one can choose certain nodes and edges when looking at the coarser timeline, and use those to limit the finer timelines to those time-centric networks that contain the specified nodes and edges. Naturally, the potential of this method is not only to zoom into finer granularities, but also to *zoom out* and observe the bigger picture. For example, in an news data set, there may be an event for a certain date, a user is particularly interested in. By zooming out, and having a look at a coarser granularity, she can examine networks that are based on months or years, which include the network represented in the finer structure. Thereby, zooming out empowers exploration to put isolated occurrences into a broader context, helping to understand more complex courses of an event. Zooming in and out are not processes disconnected from each other, instead it is usually beneficial to zoom in on certain events, and then to zoom out again to get a better understanding of the broader picture, after observing the sub-networks on a finer granularity. After all, content exploration is often about assembling pieces of a large puzzle, and combining the information to get the full picture.

In an alternative implementation for this method, the user starts with an arbitrary network G, in which she identifies relevant relationships and entities. Firstly, one can then provide a zooming out operation that displays the respective coarser network, which is a supergraph

of G. In this supergraph, the respective subgraph G can be highlighted, which allows to differentiate between terms that are present in only the supergraph, or in both graphs. However, a broader context can also be explored by observing how certain relationships are embedded in the bigger picture. Secondly, a zooming in operation can be provided that, given the same network G, enables the user to select a graph of finer granularity, that ranges in the same temporal interval as G. For example, when given the graph associated with *June 2000*, the user can select one of the days within *June 2000* for which a graph exists, investigating the origin of specific relations, and being able to identify crucial parts of the documents by utilizing the function **sent**. This process can also be implemented to provide a dynamic user interface, allowing for an alternate zooming in and out, which enables the user to investigate multiple graphs after one another.

3.6.4 Filtering Networks

Until now, a time-centric co-occurrence graph was assigned to each timestamp $\omega \in \Omega$, taking into account all co-occurrences that were extracted by our time-centric extraction method. In this section, we want to investigate constraints, or restrictions, that can be placed upon time-centric co-occurrence graphs $G_\omega \in G_\Omega$, as well as limiting our domain Ω to those temporal expressions ω that are more relevant than others. This helps to focus on the most important details of the timeline and its associated graphs. Additionally, limiting a timeline to fewer timestamps, or reducing the respective networks to fewer nodes and edges can speed up computation times when dealing with large amounts of data.

Relevant Dates

Timestamps that are only mentioned once or twice in the document collection are probably less meaningful for analysis than those that occur frequently. As a result, one natural filtering technique involves counting the number of occurrences of a date, and disregarding all timestamps below a user-specified threshold n. Beside this straight-forward approach, it is also possible to employ metrics like tf-idf (see Section 2.3.3) to weight the dates, and utilize those metric's scores for thresholding. Additionally, the user can be provided with a visual indication that represents the frequency, or another associated metric, of the corresponding date, e.g., by using different sizes or colours for points on the timeline. However, employing such a filter also has its disadvantages: The frequency of a timestamps does not necessarily correlate with its informative value, and some analysis might even be interested in low frequency timestamps. Hence, the choice of metric is highly influenced by the use case, and not all applications profit from this approach.

Aside from filtering dates by their frequency, one can also specify a start- and end-date for a constructed timeline. Dates before the start- and after the end-date are ignored, and are not depicted on the timeline. This focuses the timeline on dates the user is really interested in, ignoring parts that are irrelevant for the task at hand, accelerates computation times, and even partly ignores timestamps in the text that are misclassified, i.e., those temporal expressions to which an incorrect date was assigned during the temporal information extraction step. Misclassification is a subject that is discussed in more detail in Section 4.2.2.

In general, the filtering of dates can be conducted at two points in the processing pipeline:

- During the step of calculating time-centric co-occurrences, certain dates can be skipped, and hence, the set of extracted co-occurrences becomes smaller. This can accelerate computation times when the set of dates, that needs to be kept, is known from the beginning.
- During timeline construction, first all networks are created, but some of them are discarded at a later processing step. Since, co-occurrences only need to be extracted once, this accelerates computation time if one needs to construct multiple timelines from the same time-centric co-occurrence graph collection.

Since, it is also possible to restrict the set of dates to certain granularities, the second point is especially beneficial if one wants to construct three independent timelines of varying granularity that can then coexist side-by-side.

Reducing Networks

In large document sets co-occurrence graphs can become inconveniently large. While this is in some cases inevitable, and many network measures are dedicated to extract meaningful information from such graphs, for content exploration it is more desirable to yield graphs whose meaning can be easily grasped by a human. Therefore, in this section, different options for the reduction of the size of time-centric co-occurrence networks are discussed. Firstly, it is obvious that to achieve this goal, it is compulsory to delete, or for visualization purposes at least ignore, some nodes and edges. However, it is crucial to determine those nodes and edges that carry the least amount of information. Therefore, given a temporal domain Ω and time-centric co-occurrence graphs G_Ω, a weight is assigned to each node:

Definition 16 (Node Weights for Time-Centric Co-Occurrence Graphs) *Given a time-centric co-occurrence graph $G_\omega = (N_\omega, L_\omega) \in G_\Omega$, the mapping $\kappa_\omega : N_\omega \to \mathbb{R}$ assigns a weight to each node $n \in N_\omega$.*

One of the simplest approaches to weight nodes is to just count the number of times node $n \in N_\omega$ occurs in the vicinity of timestamp ω, i.e.,

$$\kappa_\omega(n) = \mathrm{tf}(n, \omega) = f_{n,\omega}. \tag{3.10}$$

Hereby, tf is the term frequency in the neighbourhood of timestamp ω, and $f_{n,\omega}$ denotes the raw count of the term represented by n. In this thesis, the term frequency is always normalized by the total number of words occurring in the context windows of ω, i.e.,

$$\mathrm{tf}(n, \omega) = \frac{f_{n,\omega}}{N_\omega}, \tag{3.11}$$

with N_ω being the number of words in all of ω's context windows.

However, this simple technique often yields networks that only contain very common words in regards to its domain. For example, it is not unusual to find words like *"report"* or *"politician"* in a political news data set. However, those words usually do not carry desirable information, they are too vague to be of specific interest. Instead, a different ranking function is employed that behaves similar to the tf-idf weighting scheme presented in Section 2.3.3. The weighting is altered in the way that instead of counting in how many documents a term appears, the number of co-occurrences with unique instances of temporal expressions is calculated. We call this the *inverse timestamp frequency* (itf), and define it as follows:

$$\mathrm{itf}(n, \Omega) \coloneqq log\left(\frac{M}{1 + |\{\omega \in \Omega : f_{n,\omega} > 0\}|}\right), \tag{3.12}$$

with $n \in N_\omega$, and M being the number of unique timestamps in the document set, i.e., $M = |\Omega|$. The denominator is given by the number of timestamps n co-occurs with. 1 is added to the denominator to prevent zero division.

Term frequency - inverse timestamp frequency (tf-itf) is then defined by

$$\text{tf-itf}(n, \omega, \Omega) \coloneqq \mathrm{tf}(n, \omega) \cdot \mathrm{itf}(n, \Omega). \tag{3.13}$$

Similar to tf-idf, this weighting scheme ranks nodes whose terms co-occur more often with the given timestamp ω higher, which is beneficial, since often occurring terms are probably more significant than rare terms. Additionally, it penalizes terms that appear in the context of a large amount of timestamps. This is desirable, because we aim to rank terms higher if they are particularly meaningful for a given timestamp, and not just omnipresent. However, the question remains how to employ this ranking function for timeline exploration: A useful

approach is to limit the number of nodes in the co-occurrence network to the top $x \in \mathbb{N}$ highest ranking nodes, or to set a threshold a node has to pass to be part of the graph. In summary, this scheme enhances the distinguishability of different co-occurrence networks.

A different approach to limit the number of nodes in time-centric co-occurrence graph is to allow only certain entity classes to be part of the network, e.g., only persons or locations. One can also employ entity-centric networks like the LOAD graph model [59], or any variant of the models introduced in Section 3.4, to structure time-centric co-occurrence graphs. In general most of the co-occurrence graph models, and analysis on these graphs, discussed earlier in this thesis, are also a suitable choice for the construction of time-centric co-occurrence graphs.

3.6.5 Projected Entity Networks

In Section 3.4, one-mode projection was employed to construct networks in which terms share an edge when they occur in the vicinity of the same temporal expression. In this section, a similar technique is presented that allows to create a similar projection based on time-centric co-occurrence graphs. Recall that the disadvantage of the one-mode projection approach is that it does not take into account if two terms co-occurred with each other, but only if they co-occurred with the same temporal expression. Hence, it cannot be asserted that two expressions, sharing an edge, really relate to each other. With the help of time-centric co-occurrence graphs, one can overcome this drawback. To do this, time-centric *projected entity networks* are discussed.

Definition 17 (Time-centric Projected Entity Network) *Given a time-centric co-occurrence graph collection (Ω, G_Ω, ξ), and associated graphs $G_\omega = (N_\omega, L_\omega) \in G_\Omega$, a **time-centric projected entity network**, or just projected entity network, $E = (N_E, L_E)$ consists of a set of nodes N_E with $n \in N_E$ if $\exists \omega : n \in N_\omega$, and links L_E with $l \in L_E$ if $\exists \omega : l \in L_\omega$.*

More descriptively, a projected entity network consists of all nodes that are part of a time-centric co-occurrence graph, and two nodes share a common edge if they also share an edge in one of the co-occurrence graphs. Additionally, an edge label, consisting of a set of dates, is assigned to each link, which indicates at which points in time both nodes co-occurred with each other. This is especially important for time-centric exploration, since it enables users to further analyze the relationship between two entities. To reduce the size of entity networks, one can employ constraints similar to those in Section 3.6.4. For example, it is possible to limit the set N_E to nodes that exhibit a minimum node weight as defined in Definition 16; or deduce edge weights from the weights given in the time-centric co-occurrence graph, ignoring edges whose weights are below a given threshold.

3.6.6 Evaluation of Timeline-Centric Co-Ocurrence Graphs

One of the challenges of content exploration is to determine the quality of results. Hence, also in the context of time-centric exploration, we have to ask ourselves, if the constructed graphs and timelines are representative of the reality, or at least its depiction in the documents, that are to be outlined. Since a ground truth is usually not available for exploratory tasks, metrics like precision and recall cannot be directly applied, and it is necessary to resort to more creative, and often manual, evaluation methods. This section discusses possibilities for evaluation, which are then also applied in Chapter 4 to evaluate the results of the presented timeline construction.

Reliability of Temporal Information Extraction

The description of the above defined models always assumed a set of given precise temporal expressions Ω. However, extracting these timestamps is already a challenge in itself as discussed in Section 2.4, and as a result, evaluation of this extraction is a crucial task. Only if it is investigated in which cases the chosen extraction method works correctly, and in which it fails, it becomes possible to assess subsequent results. However, not only the quality of the temporal information is important, but also its frequency. Results for dates that occur often in the document collection are more reliable than those for timestamps that only occur a few times. In the latter case, single outliers can affect the outcome significantly, lowering the reliability of results.

Furthermore, the temporal information extraction applied in this thesis, ensures expressions that are detected on day granularity are also part of networks of coarser granularity. Analysis of the results, hence, includes to assure that this is done correctly; and by assessing the fraction of co-occurrences that are extracted in respect to a coarser granularity, e.g., year-based, but not a finer granularity, e.g., month-based, it can be estimated how granular date specifications were in general.

Representation of Key Events and Entities

In every data set, one can identify some major events that are more important than others. For example, the September 11 attacks were so momentous that it can be assumed that a large news data set includes articles and reports about the incidents. Hence, the respective co-occurrence graph to *September 11, 2011* should be associated with keywords like *World Trade Center*, *New York*, or *terrorism*. A lack thereof indicates a lower quality of the given timeline reconstruction. On the other hand, it can be assumed that an American football team like the *Green Bay Packers* are mentioned more frequently in years when they won the

famous *Super Bowl*. Hence, such entities should be ranked higher in regard to their tf-itf (see Section 3.6.4) in such years.

For an easier evaluation, we define a measure, that allows for analysis of the tf-itf of the observed term in the respective time-centric co-occurrence graph in comparison to other entities, the tf-itf rank:

Definition 18 (tf-itf Rank) *An entity has* ***tf-itf rank*** *n in regard to a time-centric co-occurrence graph, if it is the entity with the n^{th} highest tf-itf.*

Hence, the tf-itf rank provides an at-a-glance way to see how important a term or entity is in regard to a given time-centric co-occurrence network.

Analysis like the one described above can only be done by choosing representative samples and, hence, is only an indicator. However, it is especially useful for smaller data sets in which events that are crucial can be identified more easily, and the quality of the given timeline can be then estimated by investigation of the most important key events and entities. Since the meaningfulness of this evaluation technique highly depends on the choice of those events and entities, it is closely linked to the goals we want to achieve. Therefore, for avoiding any bias during evaluation, it is beneficial, and even necessary, to identify crucial events beforehand.

4 Implementation and Results

In this chapter, first implementational details important to the proposed method are given. Second, the model is evaluated with regard to a document collection from the legal domain, namely court protocols from the NSU trial. In this context, the reliability of HeidelTime is discussed as well by investigating results generated from the given data set. Finally, it is shown that the model is applicable to more than one domain by evaluating a second data set about the Weimar Republic extracted from the German Wikipedia.

4.1 Implementational Details

This section outlines the implementation of the previous concepts as conducted during this thesis. It is by no means an extensive code review, and displays as little code as possible. Nonetheless, some of the details given in this section are critical to put later results into context, or discuss typical pitfalls when working with textual data involving temporal information. Firstly, we provide an overview of the most important steps of the program:

1. Preprocessing of documents,
2. Extracting timestamps using HeidelTime,
3. Process and annotate documents using spaCy [19],
4. Time-centric co-occurrence graph construction.

In the following, a short summary of the individual steps is given, and thereby, important details are pointed out.

Preprocessing of Documents

The first step of most information extraction methods from text is data preprocessing. The exact approach to this highly depends on the data and task at hand, e.g., some preprocessing steps might be necessary for certain encodings while they are unnecessary for others. Literature oftentimes does not differ from the preprocessing presented here and the following

natural language processing steps like named entity recognition. In this thesis, preprocessing among others includes normalisation of metadata like a given date stamp, i.e., converting different formats into a single one. Another pitfall for the presented method are non UTF-8 characters in the text which can usually be removed or altered without a problem, however, depending on the data, this can become rather tedious. A peculiarity for the algorithm developed during this thesis are the characters $<$ and $>$, which are removed since they can be easily confused with the XML-like structure of HeidelTime's tags.

Extracting Temporal Information

After preprocessing the documents can be processed by HeidelTime as described in Section 2.4. Since HeidelTime was developed using Java, and the most part of this thesis was developed in Python, we also provide a Python wrapper for HeidelTime[1]. The wrapper is built upon HeidelTime's standalone version, and provides the same functionality as the command line arguments described in the standalone manual[2] by Zell and Strötgen [76]. It is explicitly stated where command line arguments were provided for evaluation purposes. Multiple documents can be easily processed in parallel, which reduces computation times significantly for large document collections.

Processing Documents with SpaCy

For processing of the given documents with spaCy, the *'de_core_news_md'* model is loaded and altered, s.t. the following pipeline is employed:

1. sentencizer: Used to separate the text into different sentences.
2. tagger: Executes Part-of-Speech Tagging.
3. parser: Performs dependency parsing.
4. ner: Named Entity Recognition.
5. merge_entities: Unites multiple tokes that represent one unique entity into one token.

While the individual steps are not detailed here, it is important to consider that this procedure allows to organise the given textual data in the form of a document collection as described in Section 3.2. Additional to its spelling, every token also carries information about what type of entity it is (if it is one at all), and whether it is a timestamp recognized by HeidelTime. It should be noted that during this processing, it is critical to keep track

[1] `https://github.com/PhilipEHausner/python_heideltime` ; accessed 22. Nov., 2019
[2] `https://github.com/microth/heideltime.standalone-manual/blob/master/Manual.pdf` ; accessed 22. Nov., 2019

of these temporal expressions separately, since the tags generated by HeidelTime cannot be processed directly by spaCy together with the rest of the document. Hence, they have to be removed before this processing, and information about them is stored separately in regard to their character position in the text. Besides from this rather technical aspect, it is important to note four more steps that are undertaken:

1. All words, that are not an entity, are lemmatized by spaCy as introduced in Section 2.2.4, thus, yielding a more homogeneous vocabulary.

2. Additionally, stop words are removed, reducing the amount of data that needs to be processed during later steps. A term is considered a stop word if it is listed in a set of stop words pre-defined by spaCy, which is additionally expanded by the set of stop words the Natural Language Toolkit (NLTK) [35] employs.

3. Out-of-vocabulary words are removed, i.e., only words that are contained in a German dictionary are kept. Exceptions to this are all tokens that are marked as an entity.

4. Entities categorized as a person and consisting of more than one term can be reduced to their last term, thus, only their last name remains. While this method is not beneficial for all domains, it improved results for the later NSU data set, since different expressions for the same person were successfully unified. For the second Weimar Republic data set, however, this step is omitted.

As a result, this step yields a document collection, in which a sentence is represented as a bag of words, where each word is a lemmatized token, or an entity, and carries additional information; and as already elaborated, this model allows for an efficient approach to extract co-occurrences, which is the next step in the algorithm.

Time-Centric Co-Occurrence Graph Construction

Before construction of the graphs itself, first time-centric co-occurrences need to be extracted using the approach introduced in Section 3.5.1. Usually a window size of four sentences left and right of the timestamp is employed. Temporal expressions are transformed into a standardized format in this step, s.t. different expressions representing the same time are not distinguishable afterwards. Furthermore, in this context a temporal expression of finer granularity also counts as an expression of coarser granularity, where granularities of the form day, week, month and year are considered. This means an expression like *"January 2000"* is also employed to determine the time-centric co-occurrences for the year 2000. However, expressions of coarser granularity are not taken into account when determining finer granularities, i.e., *"2000"* does not represent *January* 2000, *February* 2000, etc. separately.

After co-occurrence extraction, the networks itself are constructed. Therefore, for each co-occurrence an edge is inserted in the associated graph, or the edge weight is adjusted if an edge already exists. Edge weights at this point only count the number of times both expressions co-occurred. Naturally, if an edge contains a term that is not present in the network, the respective node is generated as well. After this initialization, every graph is altered as follows:

1. Self-loops are eliminated, since they do not offer any information about relationships between different entities.

2. If the timestamp the network is associated with also appears as a node, this node and all its edges are removed.

3. Every node is assigned a node weight using tf-itf as presented in Section 3.6.4.

4. Edge weights are scaled, s.t.

$$\text{edge\_weight}_{\text{new}} = \frac{\text{edge\_weight}_{\text{old}}}{\text{edge\_weight}_{\text{max}}} \cdot \text{max\_weight}. \tag{4.1}$$

   Thus, every edge weight has a value between 0 and max_weight, which is chosen to be 10 if not stated otherwise. This is only valuable for later visualization, and can be omitted otherwise. It may even be undesirable for different use cases where a comparison of edge weights between distinct co-occurrence graphs is necessary.

5. Lastly, the network is reduced to the n nodes with the highest tf-itf, which are referred to as the top_n_terms. For analysis, n is usually set to be the number of nodes in the network, hence, no nodes are removed at all. For visualization, on the other hand, n is set to 10 in this work.

Additionally, all networks are ignored whose associated date is before a given date minimum min_date, or after a given max_date when visualizing the data on a timeline. This way less memory has to be reserved, and computation times are accelerated.

For exploration tasks, visualization of results is crucial. In this thesis, vis.js [74] is employed, a visualization library that can be integrated into any website using JavaScript. Vis.js enables to visualize timelines as well as networks, and is, hence, an ideal fit. The result can be seen in Figure 4.3.

4.2 Juridical Protocols of NSU Trial

The first data set utilized for evaluation purposes is a publicly available document collection containing juridical protocols of the NSU (National Socialist Underground, or Nationalsozialistischer Untergrund in German) trial [47][3], or in short, the *NSU data set*. The NSU was a neo-Nazi terrorist group active until 2011. All documents are written by private contributors to the project *NSU-Watch*, and are no official governmental protocols. The NSU trial took place between May 6, 2013 and July 11, 2018 in Munich, and is one of the largest trials in German history, costing multiple million Euros, and attracting a lot of attention in the German public. The accused persons were *Beate Zschäpe*, and her alleged supporters *André Eminger*, *Holger Gerlach*, *Carsten Schultze*, and *Ralf Wohlleben*. They were accused of different crimes involving ten murders, robbery, arson, and establishing a terrorist organization that operated for more than a decade. Major protagonists in these crimes were also *Uwe Mundlos* and *Uwe Böhnhardt*, however, the two committed suicide after a bank raid on November 4, 2011. The trial drew additional attention, since it raised serious questions about the relationship between German police forces and racist organizations. On July 11, 2018, all accused received sentences between two and a half years and life imprisonment; though, this did not end public discussion about the trial itself, since many questions remain unanswered. For a thorough introduction, the interested reader is referred to the work of NSU-Watch [47], or to the *Bundeszentrale für politische Bildung*[4]. However, it should be mentioned that these resources are mainly available in German language, while Koehler [28] also gives an introduction to the case in English. In the following section, the data set is introduced, and the quality of the produced results given the proposed methods is evaluated.

4.2.1 Description of the Data Set

The NSU data set covers 387 documents, each representing one of the 437 trial days, and consisting of 180,091 sentences and 972,888 words after the processing described in Section 4.1. Protocols for fifty trial days are missing, because they are not available from NSU Watch. Most of these days are within the last 100 days of the trial, with the first trial day missing being number 276. The average number of sentences in one document is about 465.4, however, this number has to be taken with a grain of salt: While spaCy's sentence splitting is reliable in most cases, it nonetheless fails in some cases which is also reflected in the average number of about 5.4 words per sentence. Mainly, problems arise when name abbreviations

[3] The used data set is actually a cleaned version that has minor differences to the publicly available data set. All results presented here are in regard to this cleaned version.

[4] `http://www.bpb.de/politik/extremismus/rechtsextremismus/167684/der-nationalsozialistische-untergrund-nsu`

#	Entity	Total Occurrences	Occurrence in x Documents
1	Götzl	14552	374
2	Zschäpe	5086	353
3	Böhnhardt	4150	305
4	Mundlos	4027	303
5	Wohlleben	3391	310
6	Brandt	2886	139
7	Klemke	2400	225
8	Schultze	1762	148
9	Vorhalt	1508	166
10	Temme	1461	35

(a)

#	Entity	Total Occurrences	Occurrence in x Documents
1	Götzl	14116	367
2	Zschäpe	3999	343
3	Böhnhardt	2769	277
4	Wohlleben	2734	292
5	Mundlos	2712	275
6	Brandt	2453	95
7	Klemke	2148	190
8	Vorhalt	1491	163
9	Hoffmann	1308	145
10	Temme	1292	32

(b)

Table 4.1: The ten most frequent entities in the NSU data set by total occurrences for (a) entities of type person reduced to their last name, and (b) representing all entity by their complete name. The last column depicts the number of documents, the entity occurs in.

are present in the text, e.g., *"Sch."*, or *"Be."*, since spaCy oftentimes does not correctly identify them as an abbreviation, and interprets the dot as a sentence splitting symbol. As a result, trial days where many anonymous witnesses were heard, tend to produce many short sentences. Figure 4.1a depicts the cumulative distribution function of sentences over the given documents. It can be observed that most documents are split into a few hundred sentences, with a few outliers having only few, and some having more than a thousand sentences. Nonetheless, wrongly splitting the text into many short meaningless sentences leads to many co-occurrences being ignored that can contain meaningful information, which is a potential downside of this sentence splitting method.

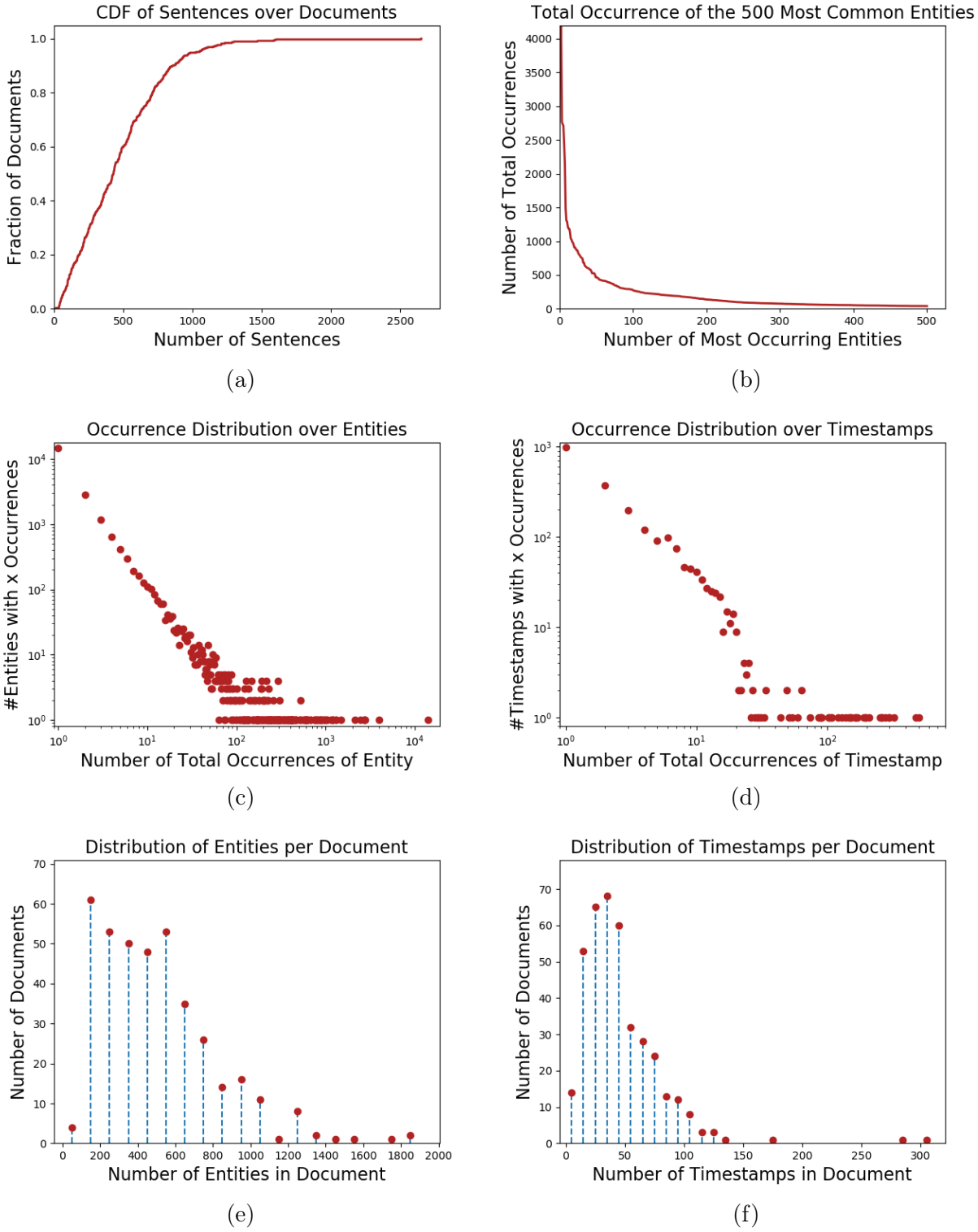


Figure 4.1: Statistics about NSU Data Set: (a) CDF representing how many documents have at least x sentences. (b) Number of occurrences of the 500 most frequent words, following Zipf's law. For better visibility of results, the number of occurrences for the most frequent word is out of limits. (c) Log-log plot of the occurrence distribution over entities, and (d) over timestamps. Both (c), and (d) resemble a power-law distribution. (e) Number of entities occurring per document, using a binning with intervals $[0, 50)$, $[50, 150)$, ... and (f) number of timestamps occurring per document, using a binning with intervals $[0, 5)$, $[5, 15)$, ...

#	Timestamp	Total Occurrences
1	Y1998	499
2	Y2000	473
3	Y2015	326
4	Y2011	299
5	Y1997	294
6	Y2001	279
7	Y1999	265
8	Y1996	259
9	Y2006	255
10	Y2004	210

(a)

#	Timestamp	Total Occurrences
12	Y2011-M11-D04	190
24	Y2015-M12-D09	109
30	Y1998-M01-D26	86
32	Y2006-M04-D06	64
34	Y2000-M05-D07	59
36	Y2004-M06-D09	51
39	Y1998-M08-D25	44
40	Y2007-M04-D25	34
40	Y2016-M01-D21	34
43	Y2016-M07-D21	31

(b)

Table 4.2: The ten most frequent identified timestamps in the NSU data set by total occurrences with (a) no restriction to granularity, and (b) only considering day granularity.

In total, 174,608 entities as well as 15,304 timestamps have been extracted from the document, resulting in 1078 individual time-centric co-occurrence graphs, 864 having day granularity, 192 month granularity and 22 year granularity. Figure 4.1b depicts the total occurrence count of the 500 most frequent entities, which resembles a typical Zipfian distribution. Table 4.1a shows the number of occurrences for the ten most occurring entities in more detail. The first thing that attracts attention is that all of those entities are persons, while for example no locations are present. This is odd, since it would be expected that Munich, the city the trial was held in, is a rather frequent term, but Munich is only found at place 133. The most occurring city is in fact Jena with 1,192 total occurrences at place 16, which is the city where Zschäpe, Mundlos and Böhnhardt were born and where they were part of the local neo-Nazi underground. The most-occurring entity in regard to Table 4.1a is Manfred Götzl who was the presiding judge at the trial, the entities 2 to 4, and 8 were directly accused or associated with the crimes. Rank 6 was a liaison officer, entity 7 a defense lawyer, 10 was employed at the Office for the Protection of the Constitution and a controversial figure regarding the murder of Halit Yozgat in Kassel. Hence all entities except for number 9, which is a regular German word wrongly classified as a named entity, were indeed major characters in the trial. Further frequently occurring entities are among others *Sch*, *S.*, or *G.* The highest ranking of those is *Sch* (rank 14), which is probably an anonymous witness, or to be more precise: Multiple witnesses whose surname was abbreviated the same way. In the documents, one can find among others *Robin Sch*, *Steffi Sch* and *Horst Thomas Sch*. While this is certainly also due to the fact that persons are reduced to their last name in the algorithm, *Sch* ranks also fifteenth even if persons are represented by their full name, and thus this does not solve the problem. Table 4.1b shows the ten most frequent entities

if an abbreviation of person names is omitted, and one can observe that most entities stay the same, even if the ordering is slightly different, and none of the entities consists of more than one term. Furthermore, the first entity occurring with its full name is *Uwe Mundlos* (rank 11 with 1218 occurrences), and the next frequent entity involving *Mundlos* is *Siegfried Mundlos*, the father of Uwe Mundlos, with 38 occurrences, which is only a small fraction of the occurrences of the surname. For the other most frequent entities similar results hold, and hence, in this data set the advantages of abbreviating the surname prevail. Another interesting aspect is the number of documents, the entities occur in: As Table 4.1 reveals most major characters, like judge *Götzl*, or the main defendant *Beate Zschäpe*, appear in almost all documents, while the witness *Andreas Temme* occurs only in some documents, but in those very frequently, indicating that some documents may heavily focus on one actor.

Figures 4.1c and 4.1d depict the occurrence distributions over entities and timestamps, respectively. It should be noted that in this results section, entities and timestamps are separated, s.t. a temporal expression is not considered to be an entity. Both distributions resemble a power-law, showing that most entities and timestamps only occur infrequently while a big part of the document collection concentrates on only few entities and timestamps.

Table 4.2a focuses on the ten most occurring timestamps: It can be seen that all of them are of year granularity; the first date of finer granularity is *Y2011-M11-D04*, which is the day of the suicide of *Uwe Mundlos* and *Uwe Böhnhardt*. Table 4.2b depicts the ten most frequent timestamps considering day granularity, indicating that for most days, there are only few occurrences. Additionally, Table 4.2 shows that the period for which most timestamps are found is reasonable, since they range from the beginnings of the NSU in the 1990's to the end of the trial in 2018, indicating that the extracted dates are indeed relevant to the trial.

Figures 4.1e and 4.1f show the distribution of entities and timestamps per document, i.e., the number of documents that contain a certain amount of entities or timestamps. One can observe that in most documents between a hundred and a thousand entities appear, and most documents contain less than a hundred identified timestamps with only a few outliers. However, the Pearson correlation coefficient between the number of words in a document and the number of found entities is about 0.897, and the correlation coefficient between the number of words and timestamps is 0.721. Both numbers indicate a high correlation between the two variables, and hence, it can be assumed that there are only few documents with an unusual high density of entities or timestamps, but it is mainly the length of the documents that differs.

	Day	Month	Year
Day	100	-	-
Month	79.9	100	-
Year	54.0	67.5	100

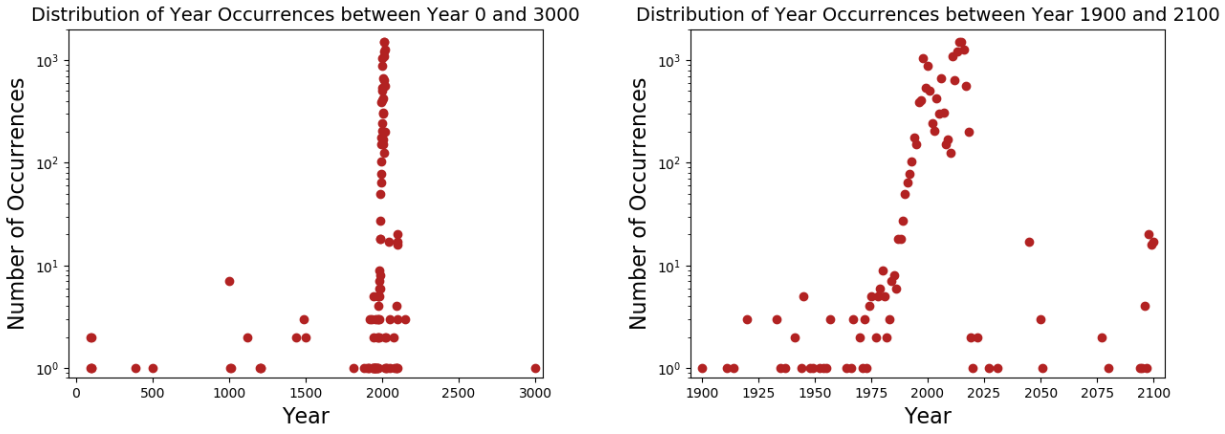
Table 4.3: Percentage of co-occurrences that were present in a time-centric co-occurrence of coarse granularity, and that were also found in one of the respective time-centric co-occurrence graphs of finer granularity.

4.2.2 Evaluation of HeidelTime

First of all, the reliability of timestamps generated by HeidelTime, and given the above described data set, is assessed by exploring the extracted results; examining cases in which HeidelTime does succeed, and in which it fails. This is not meant to be an extensive evaluation of HeidelTime, but nonetheless provides an examination that puts later results into context. Before discussion of results, the most relevant specifications used to apply HeidelTime are illustrated:

- Firstly, the document *language* is set to *GERMAN*, since all protocols in the collection are composed in German.
- The document *type* is set to *NEWS*, which mainly differs to type *NARRATIVES* in the regard that it considers the document creation time when determining the value of a temporal expression.
- *Document creation time (dct)* is set for every document to the date of the respective trial day, which is given in the data set. It is used as a reference date to resolve relative expressions like *"today"*.
- *Temponym tagging*, i.e., resolving dates by considering events like the murder of John F. Kennedy and deducing the date from such an expression, is disabled. This is due to the fact that, to the best of our knowledge, temponym tagging with HeidelTime is at the time only applicable for English documents.
- All other settings are not adjusted, and hence, default values are employed.

Figure 4.2 gives a first impression of the performance of HeidelTime. The overwhelming majority of extracted timestamps are around the year 2000, especially between the end of the 1990's and 2020. This coincides with the beginning of the NSU crimes, and the end of the trial, and thus, it can be assumed that at least the general period of the extracted temporal expressions is correct. Furthermore, Table 4.3 shows which percentage of



(a) (b)

Figure 4.2: Year occurrence distribution of the NSU data set considering the years (a) 0 to 3000, and (b) 1900 to 2100, respectively. Note the logarithmic scaling on the y-axis in both figures.

co-occurrences that were present in a time-centric co-occurrence graph of coarse granularity were also found in the respective finer time-centric co-occurrence graphs given the introduced date inclusion hierarchy. It is observed that 54% of co-occurrences found in a graph of year granularity were also part of a graph of the respective day granularity, and 67.5% were part of a graph of the respective month granularity. Hence, it can be concluded that about a third of co-occurrences is only extracted in regard to year granularity, while more than half of the extracted co-occurrences are also present in a network of the finest possible granularity.

In the remaining part of this section, the soundness of the different forms of expressions HeidelTime aims to detect are discussed: explicit, implicit, and relative expressions, mostly concentrating on the latter one, since this is the source of most misidentified timestamps. Note that this analysis only considers temporal expressions of the *Date* type, and does not touch any of the other functionalities of HeidelTime. Additionally, examples from the data set are given in German with translations provided in brackets.

Explicit Expressions

Explicit expressions, i.e., those of the form *"August 1, 2010"*, are generally extracted very well, and examples of missed dates are rare. This is especially important, since most dates found in the data set are of an explicit form. While the majority of occurrences is extracted correctly, it is nonetheless valuable to take a closer look at cases where the extraction fails. Note that in German dates are usually written in the form *dd.mm.yyyy*.

- The date *"5.Juni"* (June 5) is not identified correctly, probably because of a missing space character.
- *17.03* or *19.01* are not tagged either, while the complete expression *19.01.2001* is extracted correctly. The reason for this is unclear, since in cases like *June 5* the date is found even without a given year.
- An edge case is *"Asservat Nr. 2.2.66"* (court exhibit nr. 2.2.66), which is identified as *February 2, 1966* (and thus is strictly speaking more of a relative expression). However, this is a rare case which can be mostly ignored in this data set.

Hence, the extraction of explicit expressions works really well, which means that they only offer few ways for improvement that would mainly depend on the actual use case.

Implicit Expressions

The extraction of implicit expressions, e.g., Christmas, is harder to evaluate using the given data set, since only few such expressions are present. New Year's Eve and Christmas are reliably identified, and also *"Erntedankfest"* (harvest festival) is found, however *"Erntedank"* (also harvest festival) is not extracted by HeidelTime, which is rather odd, since they can be used interchangeably in German. Nonetheless, an analysis independent from the data set, detects among others the following holidays: *"Neujahr"* (New Year's Day), *"Palmsonntag"* (Palm Sunday), *"Karfreitag"* (Good Friday), *"Ostermontag"* (Easter Monday), and *"Pfingsmontag"* (Whit Monday) are all dereferenced to the correct date.

A problematic aspect of this extraction are coincidences with given names: In the NSU data set, one finds for example lawyer *"Freitag"* (Friday), or witness *"Jan Werner"*, whose surname is incorrectly identified as January. However, this is only a small problem, concerning only few edge cases, and can be ignored for most domains. If certain entities with such a name are common in a document collection, it would nonetheless be beneficial to identify such occurrences and remove them from the analysis.

One interesting aspect of this analysis is how well implicit expressions are extracted in regard to a given year: *"Weihnachten 2000"* (Christmas 2000) and *"Silvester 1998"* (New Year's Eve 1998) are identified correctly, but *"Weihnachten des Jahres 2000"* (Christmas of the year 2000) is extracted in regard to another reference date, and yields Christmas 2016. However, this is a problem that is largely connected to relative expression extraction, which is discussed next.

Relative Expressions

While for many cases relative expressions are extracted precisely, the most misidentifications of HeidelTime can be identified in regard to this category of expressions. In general, HeidelTime often relies on the given document creation time, which works in many cases like *"heute"* (today), or *"vergangenen Samstag"* (last Saturday) when the statement directly refers to the trial or trial day. Also in cases where the correct reference date is mentioned shortly before the occurrence of the temporal expression, the identification works reliably. However, in the following paragraphs, a few examples are pointed out, in which relative expression extraction fails, in order to provide insight in which areas HeidelTime could be improved.

- Dates with an abbreviated year representation are often dereferenced incorrectly. Examples are *"März 98"* (March 98), or *25.8.97*, which are considered to be in the years 2098, and 2097, respectively. This is due to the fact that the used reference date is also in the 21st century, and thus, HeidelTime assumes that the given date is in the same century. While this may be a valid guess in some domains, e.g., in historical texts examining multiple centuries, in a legal domain it would be beneficial to assume that the nearest year is the correct one. This could be complemented by a user input that explicitly prevents HeidelTime from assigning any dates beyond or below certain thresholds. In Figure 4.2b, one can examine how many temporal expressions are actually identified as a date between 2090 and 2099. A look at the data itself reveals this to be about 50 incidents, which is a comparably small amount.

- Specifications that follow a temporal expressions are mostly not taken into account. This has been already seen in the expression *"Christmas of the year 2000"*, but another example is *Dienstag, den 14. Mai* (Tuesday, May 14) where Tuesday only specifies which day of the week May 14 was. However, HeidelTime extracts Tuesday additionally in regard to the current reference date, s.t. the expressions results in two separate tags.

- Reference dates are often taken from a few paragraphs earlier, usually the most previous date in the text. In legal documents like the given one, however, this is not always beneficial, because oftentimes many testimonies and statements shortly follow one another. This leads to cases where the reference date was mentioned in a completely different context, and relative expressions are consequently not extracted correctly. This is particularly problematic if the relative expression does not relate to another date, but an event. An example of this would be a statement like *"A week after the acquisition of the gun..."*, where even for a human it is not necessarily an easy task to resolve when the gun was bought, which is necessary to determine the exact date.

- Given the example *"Im November und Dezember 2012"* (In November and December 2012), HeidelTime correctly extracts November and December as two temporal expressions, both referring to a month in 2012, while without specification of the year, it yields 2013. This indicates that HeidelTime also considers information later in the text to determine dates. On the other hand, an odd phenomenon occurs when looking at the expression *"vom 04. bis zum 08. November 2011"* (from 4 to 8 November), which yields results in two tags. The first identified date is November 4, 2015, and the second November 8, 2011. Apparently, HeidelTime is able to correctly deduce the respective month for the first date, however, the following year is ignored.

Summary

In summary, it can be observed that HeidelTime struggles with many corner cases, especially when taking reference dates into account. Nonetheless, most extracted timestamps are reliable, especially when processing mostly explicit expressions. It should be noted that even in the cases where HeidelTime struggles, humans would likely struggle as well when being entrusted with the same task. However, the most important improvement for HeidelTime, given a domain like the NSU data set, probably is an improved extraction of abbreviated year representations which at the moment assumes the date to be in the same century as the reference date.

4.2.3 Construction of Timeline

One of the key aspects of this work investigates the possibilities to manually explore a document collection with the help of timelines. Figure 4.3 depicts an extract from our timeline implementation utilizing time-centric co-occurrence graphs. The timeline is subdivided into three sub-timelines representing year, month and day granularity respectively. In the area of year granularity, the most occurring terms for the year are shown, and by clicking one of them, all dates are marked that have an associated graph in which the term occurs. Clicking on a date opens the associated time-centric co-occurrence graph, allowing for further analysis, and identification of crucial entities for the respective date. Here the network is restricted to the 10 terms with the highest tf-itf rank, s.t. the graph is not too cluttered. Other scenarios may benefit from presenting more or even fewer terms. The mapping **sent**, see Equation (3.9), which maps each link to the sentences where the two terms co-occurred, allows for the last step in this exploration scenario, showing in this case one of the many cases in which *Temme* and *Yozgat* co-occurred, two prominent figures in the NSU trial.

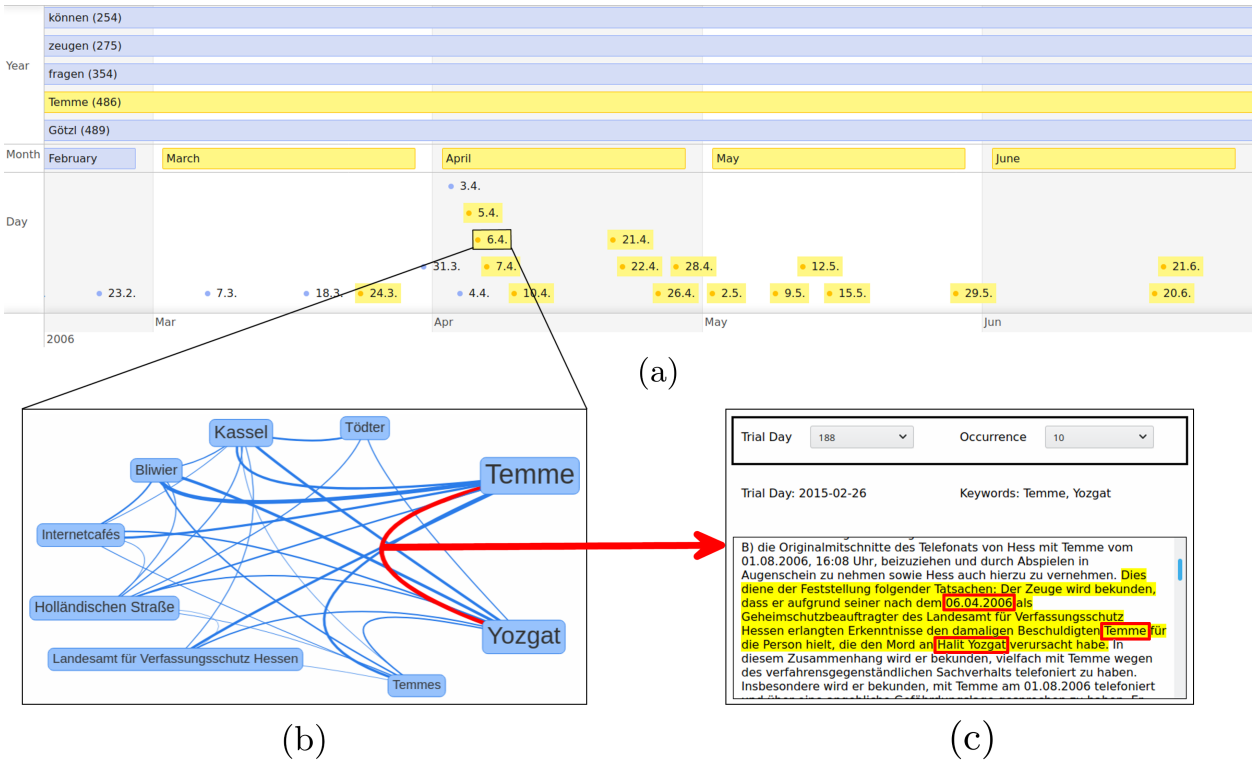


Figure 4.3: Illustration of a typical exploration process. (a) Timelines with different granularities. By selecting one of the five most occurring words in a year all networks are marked that contain the term. (b) Clicking on April 6 shows the associated time-centric graph reduced to the 10 nodes with the highest tf-itf rank. Size of nodes and edges depends on an assigned weight. (c) Clicking on an edge in the graph allows to investigate the origin of the co-occurrence.

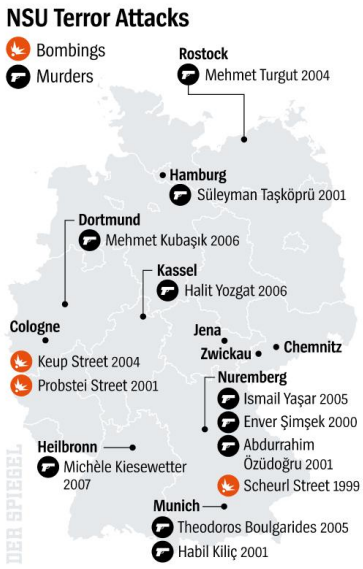


Figure 4.4: Map indicating where and when all of the bombings and murders of the NSU trio were committed.[5]

4.2.4 Representation of Key Events and Entities

In this section, important key events relevant to the NSU data set are identified, followed by an analysis of how well those events are extracted by employing the proposed methods to the data set. Firstly, it is necessary to determine the respective key events. Afterwards, it is analyzed how relevant the entities are with regard to the respective timestamps, and vice versa as discussed in Section 3.6.6.

Key Events and Entities

Key events and entities of the data set are selected by looking at the murders, bomb attacks, and robberies of the core NSU trio, the first two kinds of terror attacks being indicated in Figure 4.4. Naturally, all of those events should be present in the data set, since they were the basis for the charges, and hence, were discussed in great detail during the trial. The specific entities chosen for analysis are the cities of the respective crime, and in the case of the murders, the victims. For the robberies the street of the robbery is given as well. The relevant timestamps are the dates of the crimes. Table 4.4 gives an overview of all murders, bomb attacks, and robberies linked to the NSU trio. It should be noted that in the table the names of the cities are translated to their English equivalent, and as discussed earlier, names of the victims are only represented by their last name. For every entity, the respective tf-itf rank in the associated time-centric co-occurrence graph is specified as well.

[5]DER SPIEGEL, 16. July, 2018, https://www.spiegel.de/international/germany/neo-nazi-terror-trial-fails-to-answer-all-questions-a-1218727.html

Date	Victim	#Occs	tf-itf rank	City	#Occs	tf-itf rank
September 9, 2000	Şimşek	97	4	Nuremberg	238	3
June 13, 2001	Özüdoğru	88	1	Nuremberg	238	2
June 27, 2001	Taşköprü	88	1	Hamburg	90	6
August 29, 2001	Kılıç	96	1	Munich	216	4
February 25, 2004	Turgut	71	2	Rostock	80	1
June 9, 2005	Yaşar	103	1	Nuremberg	238	2
June 15, 2005	Boulgarides	79	1	Munich	216	2
April 4, 2006	Kubaşık	173	1	Dortmund	228	2
April 6, 2006	Yozgat	386	2	Kassel	292	3
April 25, 2007	Kiesewetter	162	2	Heilbronn	146	1

(a)

Date	City	tf-itf rank
June 23, 1999	Nuremberg	7
January 19, 2001	Cologne	18
June 9, 2004	Cologne	4

(b)

Date	Street	tf-itf rank	City	tf-itf rank
December 18, 1998	Irkutsker Straße	2	Chemnitz	7
October 6, 1999	Barbarossastraße	1	Chemnitz	3
October 27, 1999	Limbacher Straße	1	Chemnitz	3
November 30, 2000	Johannes-Dick-Straße	2	Chemnitz	1
July 5, 2001	Max-Planck-Straße	1	Zwickau	5
September 25, 2002	Karl-Marx-Straße	1	Zwickau	2
September 23, 2003	Paul-Bertz-Straße	-	Chemnitz	2
May 14, 2004	Albert-Schweitzer-Straße	2	Chemnitz	28
May 18, 2004	Sandstraße	1	Chemnitz	4
November 22, 2005	Sandstraße	-	Chemnitz	2
October 5, 2006	Kosmonautenstraße	-	Zwickau	1
November 7, 2006	Kleine Parower Straße	22	Stralsund	1
January 18, 2007	Kleine Parower Straße	19	Stralsund	1
September 7, 2011	Goethestraße	-	Arnstadt	1
November 4, 2011	Nordplatz	-	Eisenach	2

(c)

Table 4.4: (a) The dates, victims and cities associated with the murders of the NSU; (b) the dates and cities of their bomb attacks; and (c) the dates, streets and cities of the respective robberies. The tf-itf rank always refer to the rank of the term in the respective left column in the associated time-centric co-occurrence graph.

Most Prominent Nodes in Time-Centric Co-Occurrence Graphs

In Table 4.4a, it is observed that for all murders the name of the victim has at least tf-itf rank 4, mostly even rank 1, and hence is well represented in the respective time-centric networks. While not as predominant as the name of the victims, the respective city names also rank very high in regard to their tf-itf scores. Other entities that occur frequently in these networks, and have an tf-itf rank of 10 or above, are witnesses whose testimonies were linked to the date, the state prosecutors, or other cities. Moreover, a few entities are contained in the graphs that even specify individual streets that were mentioned during testimonies; as well as the murder weapon, a Ceska 83, whose acquisition was discussed extensively during the trial. In the case of the murder of Halit Yozgat on April 6, 2006, also Andreas Temme is contained in the network, a controversial figure in the trials who worked for the *Landesamt für Verfassungsschutz Hessen* (state office for the protection of the constitution), and who was suspected of being at the crime scene when the murder happened. The crime scene itself, an internet cafe, is also part of the graph. Out of the ten days where a murder took place, April 6, 2006 is also the most occurring one with 64 occurrences in the document collection. All other dates occurred around 10 and 20 times, with the exception of September 9, 2000 (26 occurrences), and April 25, 2007 (34 occurrences).

Table 4.4b shows that the cities the respective bombings took place are not extracted as well as in the case of the murders, but nonetheless all of them are ranked comparatively high. Only Cologne in the year 2001 is not contained in the top 10 ranked nodes, however, for this date *"Kölner"* (this has different meanings in regard to Cologne) ranks second, and *PP Köln* (police headquarters Cologne) third. Moreover, it can be observed that the terms *iranisch* (Iranian), and *Lebensmittelgeschäft* (grocery store) are on rank 9 and 5, and indeed the target of the bombing was an Iranian grocery store. For June 9, 2004, *"Keupstraße"*, and *"Kölner Keupstraße"* are found having rank 1 and 2, respectively. Keupstraße is the street where the bomb attack in Cologne took place at. Hence, it can be concluded that the places of the bombings are well represented in the respective time-centric co-occurrence graphs.

In Table 4.4c, one can observe that the city where the robbery took place can be extracted and linked to the respective date very well in most cases, with only one case in which the tf-itf rank of the city is below 10. The exact street of the crime can also be deduced in many cases, in some it is even ranked unexpectedly high. Moreover, for January 18, 2011, *Kleinen Parower Straße* even ranks 7^{th}, and for November 30, 2000, *Johannes-Dick-Strasse 4* is found on rank 9, which specifies the location even more. These examples show that a second look at the graphs itself is always valuable and improves the analysis. Other terms that occur

frequently are for instance the robbed banks, e.g., *Sparkasse Zwickau*, or *Sparkassenfiliale Sandstraße*. Generally, the timestamps occur 7 to 15 times, with the exception of December 18, 1998, with 20 occurrences, and November 4, 2011, with 190 occurrences. The unusual high number of occurrences of the latter date, however, is not unexpected, since this is also the day of the suicide of *Mundlos* and *Böhnhardt*. As a result, the main keywords related to this date are of less relevance to the robbery; terms that are ranked high are mostly persons involved in the NSU, or are related to the place where *Mundlos* and *Böhnhardt* were found.

For networks of month and year granularity, the identification of these events becomes less clear. For month granularity, all murder victims except for Yozgat are found at least in the 20 highest ranked nodes in regard to their tf-itf, however, most cities are not highly ranked. Instead most names refer to entities like the accused persons, multiple lawyers, criminal or governmental organizations, and many more. For year granularity this trend is even stronger, and most victims cannot be identified anymore. This indicates that in the given data set and in regard to coarser granularities less specific terms and entities are stronger represented, and terms hinting directly to singular events on the timeline are harder to detect in time-centric co-occurrence graphs of coarser granularity.

Most Prominent Timestamps for a given Entity

Previously, investigation focused on the relevance of relevant key entities with regard to a given timestamp. However, in the following it is discussed what the most relevant dates for a given entity are, and hence, to which date they are linked most closely. Table 4.5 shows the highest ranked dates for each murder victim in regard to the associated tf-itf score, ignoring any dates that are not of day granularity. By comparison with Table 4.4a, it can be concluded that the most relevant date for each victim is the day she was murdered, often followed by days closely around the date. If all date granularities are taken into account, the picture does not change much from the one given in Table 4.5: For 9 of the victims, the most prominent date, and hence the date they were murdered, is still in first place. Only in the case of Kiesewetter, the highest ranked date changes to the year 2006. In general, following on the second and third rank are often either the year or month of the respective murder case, further emphasizing the importance of the date in regard to the entity.

Date	tf-itf value
September 9, 2000	0.0003637
November 9, 2000	0.0002021
June 13, 2001	0.0002021

(a) Şimşek

Date	tf-itf value
June 13, 2001	0.0007071
December 8, 2011	0.0001326
September 9, 2000	0.0000884

(b) Özüdoğru

Date	tf-itf value
June 27, 2001	0.0007755
June 13, 2001	0.0001454
September 9, 2000	0.0000485

(c) Taşköprü

Date	tf-itf value
August 29, 2001	0.0006035
August 26, 2001	0.0002012
August 25, 2001	0.0002012

(d) Kılıç

Date	tf-itf value
February 25, 2004	0.0008110
January 26, 1998	0.0001707
April 25, 2004	0.0001281

(e) Turgut

Date	tf-itf value
June 9, 2005	0.0007683
February 18, 1995	0.0001707
June 13, 2001	0.0000854

(f) Yaşar

Date	tf-itf value
June 15, 2005	0.0004861
June 13, 2005	0.0000442
April 9, 2004	0.0000442

(g) Boulgarides

Date	tf-itf value
April 4, 2006	0.0009228
April 6, 2006	0.0002006
November 4, 2011	0.0001204

(h) Kubaşık

Date	tf-itf value
April 6, 2006	0.0019519
April 10, 2006	0.0002788
April 21, 2006	0.0001992

(i) Yozgat

Date	tf-itf value
April 25, 2007	0.0007219
November 4, 2011	0.0002280
February 13, 2006	0.0002280

(j) Kiesewetter

Table 4.5: The three highest tf-itf scores, and the date the associated time-centric co-occurrence graph is linked to, for each murder victim of the NSU trio, only considering networks of day granularity.

Date	tf-itf
June 9, 2005	5.690
June 13, 2001	4.623
Sept 9, 2000	3.912

(a) Nuremberg

Date	tf-itf
June 27, 2001	2.728
June 13, 2001	1.819
Sept 9, 2000	0.909

(b) Hamburg

Date	tf-itf
Aug 29, 2001	2.695
June 15, 2005	2.310
June 9, 2005	0.770

(c) Munich

Date	tf-itf
Feb 25, 2004	11.001
July 5, 2001	0.847
Jan 26, 1998	0.847

(d) Rostock

Date	tf-itf
Apr 4, 2006	4.817
Nov 4, 2011	3.335
Nov 15, 1995	2.594

(e) Dortmund

Date	tf-itf
Apr 6, 2006	16.109
Mar 18, 2006	5.858
Apr 4, 2006	2.563

(f) Kassel

Date	tf-itf
Apr 25, 2007	10.924
Nov 4, 2011	1.561
Nov 10, 2011	1.170

(g) Heilbronn

Date	tf-itf
June 9, 2004	6.329
Jan 19, 2001	0.791
June 8, 2004	0.791

(h) Cologne

Date	tf-itf
Nov 4, 2011	18.855
Nov 5, 2011	3.928
Nov 7, 2011	1.571

(i) Eisenach

Table 4.6: The three highest tf-itf scores, and the date the associated time-centric co-occurrence graph is linked to, for each city a murder of the NSU took place as well as Eisenach. Only networks of day granularity are considered, and tf-itf values are scaled, s.t. all entries have to be multiplied by 10^{-4}.

In Table 4.6, one can observe the highest ranked dates for all cities a murder or bombing took place as well as Eisenach, the place where Uwe Mundlos and Uwe Böhnhardt were caught after a bank robbery. For reasons of space, we refrain from taking Stralsund and Arnstadt into account, however, similar results as the one below hold for those two cities. Additionally, Chemnitz and Zwickau are omitted: For those two cities, the highest ranked dates are not linked to any of the crimes presented earlier, but mostly to earlier events, since the protagonists of the NSU grew up in those cities. Again only consider networks of day granularity are considered in the tables. By comparison with Table 4.4, it can be seen that for each of the cities the most prominent date is either one where a murder or bombing took place, or in the case of Eisenach the one the two NSU members were caught. If in a city multiple murders occurred they are either ranked on the second or third place, e.g., in the case of Nuremberg all three murders are found in the top three ranks, and their tf-itf does not differ largely. Hence, it can be concluded that our method extracts a well enough relation between the dates a murder took place and the city where the crime happened. On the other hand, in case of the bombings, the table shows that while January 19, 2001, ranks second in regard to Cologne, its tf-itf score is comparatively small. Nonetheless, this is in accordance with the fact that Cologne has only tf-itf rank 18 with respect to the time-centric co-occurrence network of January 19, 2001, as can be seen in Table 4.4. This can for example

indicate that the bombing in 2001 was not discussed in depth during the NSU trial, or that it was mainly discussed on one of the trial days missing in the data set. The representation of Nuremberg in regard to the first bombing attack on June 23, 1999, is even worse, since the date only occurs on rank 9 in regard to the city. At first look, this sounds counterintuitive, since the city ranks comparatively high in regard to the date as seen in Table 4.4. However, a look at the respective graph reveals that all nodes in the graph have a rather small tf-itf score, revealing that Nuremberg may rank high in the respective network, but the overall relevance of the date is comparatively low.

Furthermore, it can be observed that if two dates d_n and d_{n+1}, that are ranked on place n and $n+1$ in Table 4.6, where d_n is a crime date, and d_{n+1} is not, the respective tf-itf score usually drops significantly. The most noticeable exception to this is Cologne, where the score drops significantly despite the fact that on the second date also a bombing took place. The by far highest computed tf-itf scores are found in regard to Kassel and Eisenach. This puts additional emphasis on these locations, and also reflects the significance of the murder in Kassel, which attracted a lot of public interest due to the involvement of Andreas Temme, who worked for the state office for the protection of the constitution at the time, and whose part in the crime was discussed controversially. For Eisenach the highest ranked date is day of the suicide of Mundlos and Böhnhardt, which marked the end of the NSU terror.

Taking into account dates of coarser granularity, mostly year representations become dominant, many but not all of them linked to the year of a crime listed in Table 4.4. Only for Eisenach, Heilbronn, and Rostock, the highest ranked date remains to be the exact date of the crime; even for Kassel, the highest ranked date becomes *"Y2006"*, and the exact date drops to the second place. Hence, it can be assumed that in contrast to the murder victims, the cities were also discussed in a broader context, and as a result, coarser date representations become more dominant.

At this point, analysis of the relevance of dates in regard to the victims and cities depicted in Table 4.4 is concluded. Therefore, only the individual streets remain for analysis. However, most streets, except for Limbacher Straße are mentioned in the context of at most 5 different dates of any granularity, if they are contained in the data set at all. This makes an analysis trivial, and hence, we refrain from an in-depth investigation in this thesis.

4.3 Data Set of Weimar Republic

The second data set utilized in this thesis is a document collection extracted from the German Wikipedia, and centers around the Weimar Republic (we henceforth refer to the data set as *Weimar Republic data set*). The Weimar Republic terms the name of the German state from 1918 to 1933, which designates the era between the end of the Great War and the emergence of Nazi Germany. The data set consists of the German article about the Weimar Republic[6] as well as all Wikipedia pages linking to this page. Pages the original article refers to are not considered, since an article does not need to be directly relevant for events in the Weimar Republic, only because the article about the Weimar Republic contains a link to this article. Furthermore, documents are subdivided, s.t., every document contains one subsection of text in the Wikipedia page, and subsections that solely contain bibliographical references are ignored as well. This is done, since different sections in Wikipedia are usually independent of each other, and we aim to avoid misclassification of temporal expressions due to a non-desirable reference date usage of HeidelTime. By splitting the documents into smaller parts, reference dates are only utilized in the context of these smaller sections. Additionally, any articles that only consist of lists are removed, e.g., *Liste der politischen Parteien in Deutschland* (List of political parties in Germany).

In this section, first a description of the data set is given, second results obtained by the proposed methods are evaluated. An evaluation of HeidelTime is omitted in this case, since an analysis of its strengths and weaknesses is already given in Section 4.2.2. However, it should be mentioned that in contrast to the first data set, HeidelTime's *type* parameter is set to *NARRATIVES*, instead of *NEWS*, since there is no meaningful reference date associated with a Wikipedia article (the creation date usually is not relevant for the content of historical documents). All other HeidelTime settings remain unchanged. Moreover, in this data set entities, that are classified as persons, are not reduced to their last name, but the full name is kept. While this has the disadvantages discussed above, in such a large heterogeneous data set, it is highly unlikely that all entities sharing a surname refer to the same person.

4.3.1 Description of the Data Set

The Weimar Republic data set covers 62,431 documents, of which each one is a subsection extracted from a German Wikipedia article. In total, it counts 668,148 sentences and 5,851,550 words after document processing as described in Section 4.1. This averages 10.7 sentences per document, and 8.76 words per sentence, i.e., the data set mostly consists of

[6] `https://de.wikipedia.org/wiki/Weimarer_Republik` ; accessed 3. Feb. 2020

#	Entity	Total Occurrences	Occurrence in x Documents
1	deutsch	28026	14256
2	Deutschland	13464	8564
3	Berlin	11986	7096
4	Weimarer Republik	7275	6434
5	SPD	6710	3169
6	Stadt	6018	3585
7	NSDAP	5015	2617
8	Jude	4647	1793
9	Hitler	4485	1812
10	Frankreich	3310	2246

Table 4.7: The ten most frequent entities in the Weimar Republic data set by total occurrences. The last column depicts the number of documents, the entity occurs in.

rather short documents, however, the average sentence length is larger than in the NSU data set. Figure 4.5a shows the cumulative distribution function of sentences over the data set: Most documents consist of 50 or fewer sentences (98.7%), while only 16 documents, or 0.03%, have a sentence count larger than 200. Hence, this result is in accordance with the average number of sentences in the data set, and the fact that every document only consists of an article section.

In total, 1,366,200 entities and 388,436 timestamps are present in the data set. Figure 4.5b depicts the number of occurrences for the 500 most frequent entities, with the 3 highest ranking ones not displayed. It can be observed that a few entities occur more than a thousand times, after which a long tail follows, with the entity ranking on place 500 having 192 occurrences. Table 4.7 gives more details for the ten most frequent entities. Most terms in this list are completely reasonable: *deutsch* (German), *Deutschland* (Germany), and *Berlin* are the three highest ranking terms. Rank 4 is already taken by the term *Weimarer Republik* (Weimar Republic) itself, and rank 5 is the *SPD*, or *Sozialdemokratische Partei Deutschlands* (Social Democratic Party of Germany), which was one of the major democratic parties in the Weimar Republic. Rank 6 on the other hand is more curious: The term *Stadt* (town or city) is not a named entity by the definition employed in this thesis, however, Spacy recognizes the term regularly as an entity of type location, e.g., when a city like Berlin is in the vicinity.

Figures 4.5c and 4.5d show the occurrence distributions over entities and timestamps, respectively. As with the NSU data set, timestamps are separated from other named entities, s.t. they are treated different from other entities. Both figures again resemble a power-law distribution. Table 4.8a depicts the most frequent timestamps for the Weimar Republic data

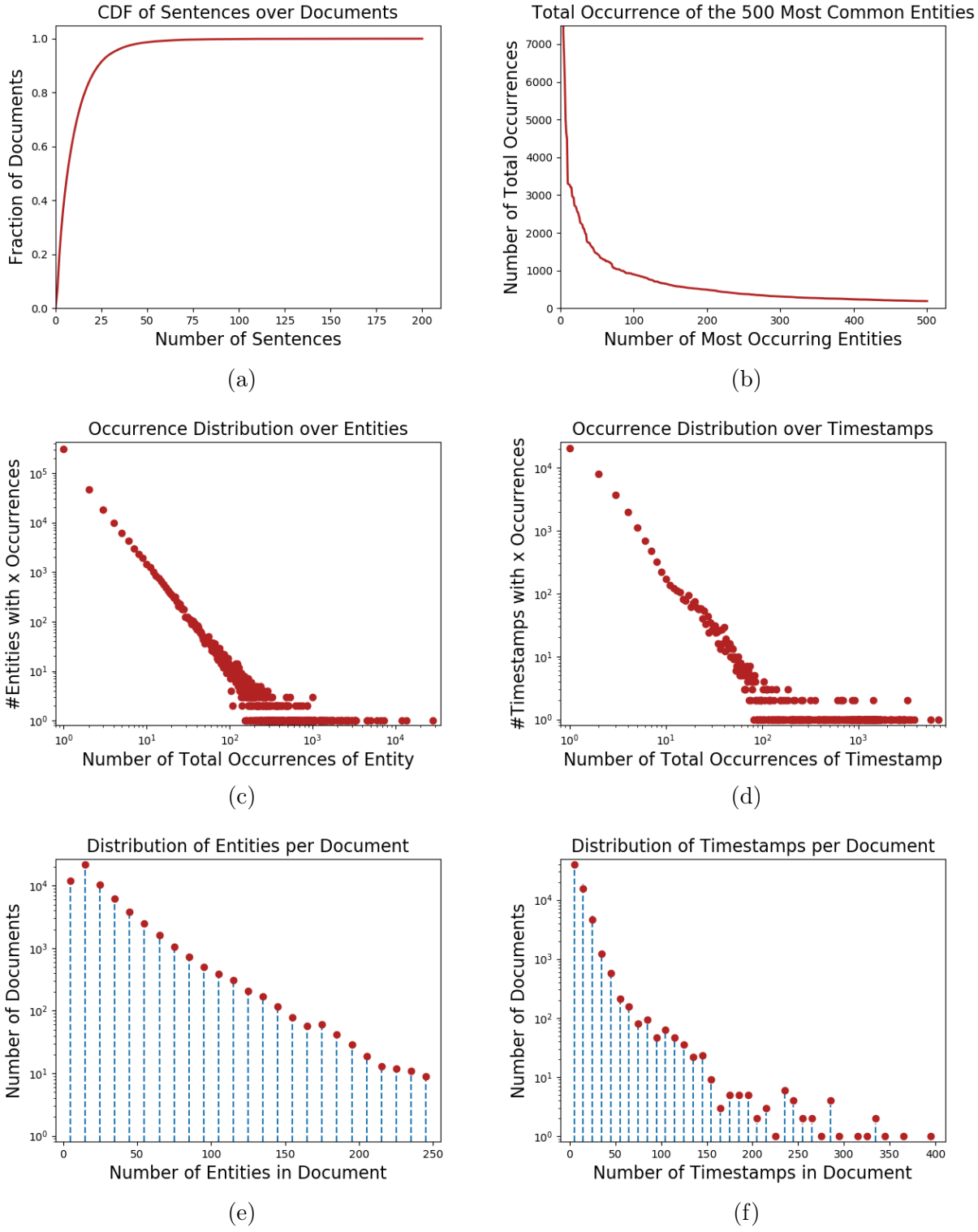


Figure 4.5: Statistics about Weimar Republic Data Set: (a) CDF representing how many documents have at least x sentences (cut at 200 sentences). (b) Number of occurrences of the 500 most frequent words. For better visibility of results, the three most frequent terms are out of limits. (c) Log-log plot of the occurrence distribution over entities, and (d) over timestamps. (e) Number of entities occurring per document, using a binning with intervals $[0, 10)$, $[10, 20)$, ... (cut at 250 entities), and (f) number of timestamps occurring per document, using a binning with intervals $[0, 10)$, $[10, 20)$, ...

#	Timestamp	Total Occurrences
1	Y1933	6781
2	Y1945	5668
3	Y1919	3802
4	Y1920	3521
5	Y1918	3428
6	Y1929	3261
7	Y1930	3208
8	Y1932	3208
9	Y1925	3059
10	Y1934	3058

(a)

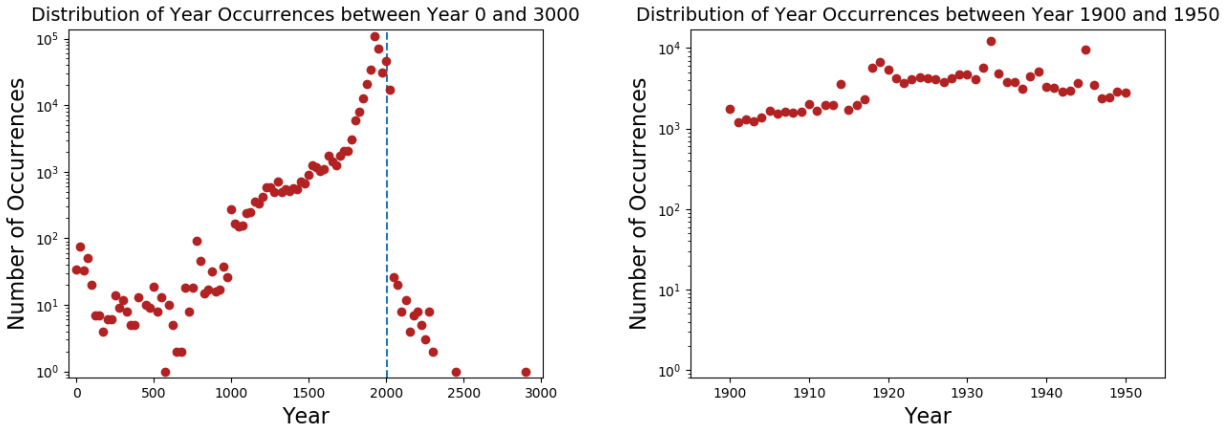
#	Timestamp	Total Occurrences
177	Y1921-M03-D20	387
199	Y1933-M01-D30	264
225	Y1944-M07-D20	203
227	Y1918-M11-D09	199
234	Y1932-M10-D01	184
275	Y1939-M09-D01	118
277	Y1933-M05-D01	117
298	Y1945-M05-D08	104
302	Y1933-M04-D01	102
325	Y1933-M02-D28	86

(b)

Table 4.8: The ten most frequent identified timestamps in the Weimar Republic data set by total occurrences with (a) no restriction to granularity, and (b) only considering day granularity.

set; it can be observed that most occurring timestamps range from 1918 to 1934, with the exception of the year 1945. This is in accordance with the fact that the German state from 1918 to 1933 is historically called Weimar Republic. However, it is noticeable that most years are either ranging in the beginning or end of the Weimar Republic, which indicates that the focus of the articles lies on the beginnings of the Republic as well as on its dissolution by the National Socialists. Table 4.8b, which focuses solely on dates of day granularity, presents a different picture: Most dates focus on the end of the Weimar Republic, or even later with the day of the German invasion of Poland, or the end of the war in 1945. Exceptions to this are the most frequent date of day granularity, which is the day of the Upper Silesia plebiscite, and November 9, 1918, the day the German Republic was proclaimed. Additionally, it is noticeable that the most occurring timestamp having day granularity in general only ranks on place 177, which is due to the high number of years occurring in such a historical data set. Figure 4.6 further confirms that the data set mostly evolves around the first half of the 20th century, and the large majority of dates lies between the years 1800 and 2020 (91.9%). Around 34.7% of the dates refer to the most relevant range between 1918 and 1945.

Figures 4.5e and 4.5f show the distribution of entities and timestamps per document, respectively. One can observe that most documents contain only a small amount of entities, which is not surprising due to the high ratio of documents with only few sentences. On the other hand, only few documents contain more than 200 named entities, and only 57 documents contain more than 250; the document containing most entities is about culture in Poland, and counts 1623 entities. Due to this long tail with only few measuring points, documents with more than 250 entities are omitted from Figure 4.5e. In Figure 4.5f, it can be



(a) (b)

Figure 4.6: Year occurrence distribution of the Weimar Republic data set considering the years (a) 0 to 3000 using a binning with intervals $[0, 13)$, $[13, 25)$, ..., and (b) 1900 to 1950 employing no binning, respectively. Note that while the left graph (a) employs a logarithmic scaling, the right graph (b) does not.

observed that there is an even larger difference between the number of documents with only few timestamps, and the ones with many timestamps. The majority of documents contains less than 50 timestamps, and only few exceptions more than 200. The Pearson correlation coefficient between the number of words in a document and the number of extracted entities in the document is 0.90; the correlation coefficient between the number of words and timestamps is 0.58, again indicating a high correlation, and as a result, it can be assumed that most documents have a comparable density of entities and timestamps.

4.3.2 Representation of Key Event and Entities

In this section follows an investigation of the relevance of entities with regard to the time-centric co-occurrence graphs constructed by the proposed method. It should be noted that graphs are only extracted for the range of years between 1918 and 1945, which results in 8496 networks, 8095 having day, 366 month, and 35 year granularity. This range limitation is firstly due to the fact that in this context the focus lies on this specific historical era, but also because of memory restraints: Since there is a large amount of date occurrences alone in the range from 1918 to 1945 (132,554 dates in total), the respective co-occurrence graphs become much larger than for the smaller NSU data set. This problem is even aggravated by the fact that we also have 8 times more individual dates in this larger data set (already limited to dates between 1918 and 1945). Hence, for this data set, more and larger networks need to be constructed.

For the Weimar Republic, it is more complicated to isolate the major important events than for the NSU data set. For analysis, a sample of events given by the English Wikipedia timeline of the Weimar Republic[7] is employed. Since most of the events cannot be linked to a single day, each event is discussed separately. The sample consists of the following events:

- Proclamation of the Republic,
- Spartacist Uprising,
- Treaty of Versailles,
- Kapp Putsch,
- Munich Putsch,
- Locarno Treaties,
- Preußenschlag,
- Hitler is appointed Chancellor.

When given multiple tf-itf ranks for one entity or term, this is due to different appearances, e.g., with and without surname, or due to inflections of the word.

November 9, 1918: Proclamation of the Republic

In the network of November 9, 1918, the most dominant entities are the first President of Germany Friedrich Ebert (tf-itf rank 1 as Ebert, and rank 7 represented by his full name), and the SPD (rank 2 and 3 as MSPD and USPD, respectively), which was the political party Ebert belonged to. Furthermore, a variety of persons relevant to the date are found: Karl Liebknecht and Philipp Scheidemann (rank 5 and 6, respectively), who both proclaimed the republic that day, and Max von Baden (rank 8 and 10), the previous chancellor of the German Empire who resigned on November 9. Moreover, terms representing events or organization are present like *November Revolution* (rank 4), *(German) Republic* (rank 9 and 12), or *Spartacus League* (rank 18). Emperor Wilhelm II., who abdicated that day, can be found on rank 52, 55, 67, and 73 in different forms. Lastly, Berlin, where the republic was announced is present on rank 16 (Berlin Palace) and 17 (Berlin Tiergarten).

[7] `https://en.wikipedia.org/wiki/Timeline_of_the_Weimar_Republic` ; accessed 17. Feb. 2020

On the other hand, for the described persons, their respective tf-itf scores are also strongly linked to the given date. The date November 9, 1918, ranks first or second in regard to all terms representing one of the persons described above, slightly varying depending on whether the full name or only the last name is employed for querying. The only instance of entities named above for which November 9 does not rank highly, is the *November Revolution*, which lasted until August 1919, and which is apparently discussed in a broader context on Wikipedia, since the dominant dates have year or month granularity.

In conclusion, it can be observed that the introduced key characters of the events of November 9, 1918, are well represented in the associated network. The only entity, which is explicitly mentioned in the employed reference timeline, but that is not present in the time-centric network, is Matthias Erzberger, who was part of the peace negotiations at the time, Kurt Eisner, who is also mentioned in the reference timeline, is found on rank 41.

Spartacist Uprising

The Spartacist uprising took place during the German Revolution between January 5 and January 12, 1919. Analyzing the top 10 terms for all time-centric co-occurrence graphs in this range, shows persons like Emil Eichhorn (Chief of the Berlin Police at the time), Friedrich Ebert, Gustav Noske (a controversial SPD politician, who supported bloodily suppression of the uprisings), Karl Liebknecht and Rosa Luxemburg (who both led the uprising). Furthermore, Waldemar Pabst, the commander of a military unit who captured Liebknecht and Luxemburg, and the Eden-Hotel, which was the headquarter of Pabst, are present. Additionally, terms related to the Spartacus League (e.g. Spartacists) and to Berlin (e.g. *Zeitungsviertel Berlin*), as well as more general terms like *uprising* are contained in the network. Finally, for January 10, 1919, the *Bremen Soviet Republic* is present, which was founded that day, but not directly related to the Spartacist uprising. Moreover, for the respective graph of month granularity, the Spartacist uprising is not present, however, terms like Liebknecht, Luxemburg, Ebert and November Revolution (German Revolution) are found are found for coarser granularities.

For all of the above-mentioned entities, their tf-itf scores are highest for dates relating to the beginning of January, especially between January 5 and January 15, which was the day of the execution of Karl Liebknecht and Rosa Luxemburg. Also the Spartacus League is mentioned during that time range, however dates relating to January 1919 generally have lower scores than those relating to November 1919. An exception to this is the term *Luxemburg*, which has higher tf-itf scores for networks between 1939 and 1940, which is due to an ambiguity of the term with the country Luxembourg in the German language.

Treaty of Versailles

On June 28, 1919, Germany signed the Treaty of Versailles, which is well represented in the respective time-centric co-occurrence graph: In different forms it ranks first, third, sixth, and eighth (e.g. *Treaty of Versailles*, and *Versailles Treaty*). Other terms in the network are related to Poland (tf-itf rank 2), the free city (most likely referring to Danzig, rank 5), Posen (rank 7) and Austria (rank 13), which reflects the regions and countries the German Empire had to cede and accept their independence. Moreover, high ranking terms include the league of nations (rank 10), which was also a result of the negotiations, and the treaties of Saint-Germain (rank 11), and Neuilly-sur-Seine (rank 15), which were treaties between the allied forces and Austria-Hungary, respectively Bulgaria. For the network of month granularity, the Treaty of Versailles is also found on rank 5 and 9, as are terms like *ultimate given by the victorious powers* (rank 3) that are closely linked to the event.

Analyzing the dates where the Treaty of Versailles scores highest in regard to tf-itf, are all dates in June 1919, mostly dating between June 21 and June 28. The exact date is highly dependent on the exact keyword, since there is a variety of different expressions representing the treaty of Versailles in the corpus. However, in general the Treaty of Versailles is well represented in the data set. Nonetheless, it should be noted that most high ranking terms refer to regions in eastern Europe, and e.g. Alsace-Lorraine, the Saarland and the Ruhr region, which were all also part of the treaty are not mentioned. Furthermore, there is no evidence of key characters that negotiated the treaty in the time-centric networks.

March 13, 1920: The Kapp Putsch

The Kapp Putsch, or Kapp-Lüttwitz Putsch was an attempt to overthrow the newly established Weimar Republic by right wing forces. The co-occurrence network for March 13, 1920, incorporates key characters of the coup, most prominently Walther von Lüttwitz (tf-itf rank 1 and 15), but also leaders of multiple participating paramilitary units like Siegfried Schulz (rank 9), Hermann Ehrhardt (rank 10 and 12), and Gerhard Roßbach (rank 17). On the other hand, also the Ruhr Red Army is present on rank 5, which opposed the putsch in the Ruhr valley. The coup itself ranks under different names second, fourth and 13-th. The Reichswehr (German military forces) ranks third, since they also played a major role during the coup, and president Friedrich Ebert ranks 11-th. However, in the top 10 nodes, two entities are found that are not directly connected, or at least not key, to the Kapp Putsch: The politician Wallbaum (rank 6), and the silent film *Sumurun*, for which the filming be-

gan on March 13 in Berlin. Hence, it can be concluded that for the respective time-centric co-occurrence network, many key characters are present, however, some protagonists are probably missing: For example, the namesake Wolfgang Kapp is not present in the network, which is probably due to his secondary role in the coup. On the other hand, this is not the case for the respective network of month granularity: In this graph, Wolfgang Kapp ranks 6-th, and the Kapp Putsch itself first and third. Furthermore, the terms Reichswehr (rank 2) and Lüttwitz (rank 4) are found among the top ranked terms regarding month granularity.

November 8 and 9, 1923: Munich Putsch

The Munich Putsch, among others also called Beer Hall Putsch or Hitlerputsch, was an attempted coup of the Nazi Party's leader Adolf Hitler. Inspired by Mussolini's *March on Rome*, the rebels aimed to march on Berlin and seize power. For the network of November 8, many protagonists of the coup are found: Ranking first is von Seeckt, who was commander in chief of the German Army at the time; rank 2 is Gustav von Kahr, a key person of the Bavarian government; rank 3, respectively 6 is Hitler; and rank 4 is the above-mentioned Hermann Ehrhardt, who refused to join the putsch. Further characters are Ludendorff (rank 5 and 7), Neubauer (rank 8) as well as Heimpel (rank 10). The city of Munich ranks on placc 9. The coup itself has rank 13 (Hitler-Ludendorff-Putsch) and 16 (Hitlerputsch), respectively. On lower ranks, other key characters of the putsch, e.g., Röhm or Göring, are found as well. Similar results hold for the network representing November 9, 1923, the second day of the uprising. For the respective network representing the associated month granularity, the term Hitlerputsch is found on rank 3 and 10, however, none of the above-mentioned key characters is present in the network.

The Locarno Treaties

The Locarno Treaties were negotiated between October 5 and 16, 1925, and led to Germany's admission to the League of Nations, and aimed to settle multiple border conflicts between the Weimar Republic and the bordering states. They were finally signed the same year on the first of December. For the Locarno Treaties, only few hints are found within the generated co-occurrence graphs: They are only represented by the term Locarno or Locarno Treaty in the networks for October 5 (tf-itf rank 3), October 12 (rank 9), October 16 (rank 1), and December 1 (rank 1). Additionally, the term Locarno is found for the network of October 1925 having rank 9. Besides of these occurrences, no indications of the treaties are found, e.g., no key characters known to the author of this thesis are present in the networks. Hence, it has to be concluded that these treaties are not well represented in the extracted time-centric co-occurrence graph, which can also hint to its minor importance in the data set.

July 20, 1932: Preußenschlag

The Preußenschlag was a successful coup on July 20, 1932, against the Weimar Republic during which at the time Chancellor Franz von Papen took control over Prussia. In the associate co-occurrence graph, highest ranked is Magnus Heimannsberg, who was a commander of the Berlin Police until the Preußenschlag. The term Preußenschlag itself is ranked fifth, and terms related to Prussia (e.g., Prussian) are ranked second and 10-th. The German Chancellor von Papen can be found on tf-itf rank 3 and 4, the Prussian Minister President Otto Braun is ranked 6-th, and German President Hindenburg 7-th. Furthermore, the Nazi Party is ranked on place 8, and the so-called *Altona Bloody Sunday*, which was a confrontation between Nazi Party and Communist supporters that triggered the events of the Preußenschlag, is ranked 9-th. In respective network of month granularity, no traces of the Preußenschlag are found. This is due to the Reichstag elections that were conducted that month, and that are more prominent in the data set than the discussed coup.

January 30, 1933: Hitler is appointed Chancellor

January 30, 1933, the day Hitler is sworn in as Chancellor of Germany, is often said to be the end of the Weimar Republic. Naturally, Adolf Hitler ranks high in the co-occurrence graph associated with the date (rank 1, 4 and 7), as do the political parties involved: The Nazi Party ranks second, the Centre Party third, and the DNVP 8-th. Additionally, further key protagonists are present, e.g., Hindenburg (rank 5 and 12), von Papen (rank 9 and 10) and Kurt von Schleicher (rank 11). Also the term Weimar Republic is found on rank 6. For the respective network of month granularity, similar results are observed. Persons like Hitler (rank 2), Schleicher (rank 3) Hindenburg (rank 5 and 6), Brüning (rank 7), and Papen (rank 9), as well as the Nazi Party (rank 4) are highly ranked.

Analyzing the highest tf-itf scores for all entities reveals that the date is also ranked comparatively high for the term Hitler, for whom it is the 12-th highest ranked date, only preceded by dates of year granularity. The second ranked day having day granularity is July 20, 1944 found on rank 27. For the term Adolf Hitler, the date ranks even 6-th, and the second highest ranked date having day granularity in regard to Adolf Hitler is November 9, 1923 on rank 11. Following are mostly dates of day and month granularity, which indicates that in a broader context, Hitler is only referred to by his last name, while for specific dates his name is more often spelled out. For the further mentioned persons involved, similar results can be observed: For all of them, January 30 has at least the 10-th highest tf-itf score, mostly preceded by dates of year granularity, or days around January 30. For the three political parties in question, however, results are different, and mostly dates of year granularity are found. Only for the DNVP January 30 is present on rank 17.

4.3.3 Discussion

The eight events described above show that the proposed method is also applicable to a historical domain. Especially, key characters as well as organizations, in this context mostly political parties, relevant to a date are extracted reliably. Furthermore, only few entries with little to no relevance to the respective events were found among the most highly ranked nodes in the associated networks, which documents also the low number of false positives the proposed method produces. However, it should be noted that time-centric co-occurrence graphs become less accurate for events that stretch over multiple days: While for the Spartacist uprising, many key characters and locations are found, dates that are directly related to the beginning and end of the event are more closely linked to the event than dates in between. For the Locarno Treaties, this becomes even more obvious, since besides of the term itself, no apparent hints to the event or persons involved in the negotiations, are found. Hence, it can be concluded that the method is more beneficial in cases where events are strongly linked to a single day, rather than events that span over multiple days or months. Moreover, for networks of month granularity, many terms are more general than for the finer day granularity, which can be for example seen in the graph of July 1932 in which most terms like political parties relate to the Reichstag elections, and terms that are linked to the Preußenschlag are not well represented in the network. For networks of year granularity this problem increases, and oftentimes, only general terms like Nazi Party, Weimar Republic, or Berlin are found, and hints for specific events are found only rarely.

5 Conclusion and Future Work

In this thesis, we discussed the usage of temporal information to structure large textual data sets by introducing a novel graph model. For this purpose the reader was first introduced to the necessary fundamentals from graph theory and natural language processing, especially focusing on word co-occurrences and temporal information extraction. After the following discussion of related work, already existing time-centric graph representations were presented, and finally time-centric co-occurrence graphs were proposed as means to structure document collections by employing contained temporal information in a novel way. In this context, we presented a way to extract word co-occurrences in a time-centric fashion, and contributed a graph model which introduces a word co-occurrence graph for each timestamp present in the data, resulting in a set of graphs. Furthermore, specific use cases of the model for time-centric exploration tasks were presented, e.g., entity-centric timelines and zooming operations. Additionally, we provided an implementation of the proposed model that arranges the results in a timeline representation. Finally, results produced by employing the proposed method on two different data sets were discussed, giving evidence that the model yields overall satisfying results for a document collection from a legal as well as a historical domain. This also included an analysis of the reliability of results produced by HeidelTime for one of the two data sets.

While the results have shown that events linked to a single date are well represented in the proposed model, it also became obvious that events that stretch over multiple days are harder to identify in this representation. This problem is also seen in graphs of coarser granularity, especially for networks representing a full year, in which mostly general terms occur that only relate to the data set at large. Future work could improve this matter, e.g., by introducing a different metric than tf-itf. One ansatz would be to apply a separate weighting for each granularity, since it can be assumed that there is no *single metric fits all* case for varying temporal resolution.

A further research topic is the handling of named entities in the text. Entities like *Hindenburg* and *Hindenburgs* are treated as two separate terms which leads to cases in which the same entity is represented multiple times in the same network. An extension of the

model could include more sophisticated entity matchers that eliminate and unify existing duplicates. This would also be advantageous, since the resulting entities would then score higher in regard to the employed tf-itf metric, further emphasizing their importance, and at the same time reducing the size of the network. Closely related to this is the problem of named entity disambiguation, which means linking a named entity to a specific existing one. An example for this is the term *Churchill* which can among others refer to various persons or locations. Resolving the appearance of such a term to its corresponding entity can separate entities that share a name, but are not the same. This topic was for example discussed in detail by Stronczek [64], whose work could also be applied to the model proposed in this work.

Lastly, we could think of analysis and extensions of the present implementation: Applying the current method to data sets that range over centuries and contain ten thousands of different points in time, leads to an enormous amount of generated networks, since for each timestamp an individual graph is generated. As a result, memory issues can arise, and future work could include to investigate such scalability issues. In this thesis, we already proposed and discussed methods to reduce the amount of networks in the model. However, one could think of more dynamic approaches that for example take into account the available memory and the number of timestamps in the document collection. Furthermore, employing more effective data structures could lead to a more compact storage of resulting graphs. However, such an endeavor could not solely concentrate on the reduction of needed memory, but would always be a trade-off between memory usage and computation times.

List of Figures

List of Tables

List of Definitions

Bibliography

[1] James Allan, Rahul Gupta, and Vikas Khandelwal. Temporal Summaries of News Topics. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 10–18. ACM, 2001.

[2] Omar Alonso and Michael Gertz. Clustering of Search Results Using Temporal Attributes. In *Proceedings of the 29th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 597–598. ACM, 2006.

[3] Omar Alonso and Kyle Shiells. Timelines as Summaries of Popular Scheduled Events. In *Proceedings of the 22nd international conference on world wide web*, pages 1037–1044. ACM, 2013.

[4] Omar Alonso, Ricardo Baeza-Yates, and Michael Gertz. Exploratory Search using Timelines. In *SIGCHI 2007 Workshop on Exploratory Search and HCI Workshop*, volume 1, pages 1–4, 2007.

[5] Omar Alonso, Michael Gertz, and Ricardo Baeza-Yates. On the value of temporal information in information retrieval. *SIGIR Forum*, 41:35–41, 2007. doi: 10.1145/1328964.1328968.

[6] Omar Alonso, Michael Gertz, and Ricardo Baeza-Yates. Clustering and Exploring Search Results using Timeline Constructions. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 97–106. ACM, 2009.

[7] Omar Alonso, Serge-Eric Tremblay, and Fernando Diaz. Automatic Generation of Event Timelines from Social Data. In *Proceedings of the 2017 ACM on Web Science Conference*, pages 207–211. ACM, 2017.

[8] Anne Aula, Natalie Jhaveri, and Mika Käki. Information Search and Re-access Strategies of Experienced Web Users. In *Proceedings of the 14th international conference on World Wide Web*, pages 583–592. ACM, 2005.

[9] Katarzyna Baraniak and Marcin Sydow. Towards Entity Timeline Analysis in Polish Political News. In *Intelligent Methods and Big Data in Industrial Applications*, pages 323–332. Springer, 2019.

[10] David Berry and Stefanie Widder. Deciphering Microbial Interactions and Detecting Keystone Species with Co-occurrence Networks. *Frontiers in microbiology*, 5:219, 2014.

[11] Bertram C. Bruce. A Model for Temporal References and Its Application in a Question Answering Program. 1972.

[12] John A. Bullinaria and Joseph P. Levy. Extracting Semantic Representations from Word Co-occurrence Statistics: A Computational Study. *Behavior Research Methods*, 39(3):510–526, 2007.

[13] Carlos Cobos, Henry Muñoz-Collazos, Richar Urbano-Muñoz, Martha Mendoza, Elizabeth León, and Enrique Herrera-Viedma. Clustering of Web Search Results Based on the Cuckoo Search Algorithm and Balanced Bayesian Information Criterion. *Information Sciences*, 281:248–264, 2014.

[14] Jérôme Euzenat and Angelo Montanari. Time Granularity. In *Handbook of Temporal Reasoning in Artificial Intelligence*, Foundations of Artificial Intelligence, pages 59–118. Elsevier, 2005.

[15] Stefan Evert. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD thesis, University of Stuttgart, 2005.

[16] Johanna Geiß, Andreas Spitz, and Michael Gertz. Beyond Friendships and Followers: The Wikipedia Social Network. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 472–479. IEEE, 2015.

[17] Cyril Grouin, Natalia Grabar, Thierry Hamon, Sophie Rosset, Xavier Tannier, and Pierre Zweigenbaum. Eventual Situations for Timeline Extraction from Clinical Reports. *Journal of the American Medical Informatics Association*, 20(5):820–827, 2013.

[18] TimeML Working Group et al. Guidelines for Temporal Expression Annotation for English for Tempeval 2010, 2009.

[19] Matthew Honnibal and Ines Montani. Spacy: Industrial-Strength Natural Language Processing, version 2.1.8, https://spacy.io/, accessed 24. Nov. 2019.

[20] Andrew L. Hopkins. Network Pharmacology: The Next Paradigm in Drug Discovery. *Nature chemical biology*, 4(11):682, 2008.

[21] Adam Jatowt, Ching-Man Au Yeung, and Katsumi Tanaka. Estimating Document Focus Time. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2273–2278. ACM, 2013.

[22] Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. TEQUILA: Temporal Question Answering over Knowledge Bases. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1807–1810. ACM, 2018.

[23] Björn H Junker and Falk Schreiber. *Analysis of Biological Networks*, volume 2. Wiley Online Library, 2008.

[24] Remy Kessler, Xavier Tannier, Caroline Hagege, Véronique Moriceau, and André Bittar. Finding Salient Dates for Building Thematic Timelines. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 730–739. Association for Computational Linguistics, 2012.

[25] Vikash Khandelwal, Rahul Gupta, and James Allan. An Evaluation Corpus for Temporal Summarization. In *Proceedings of the first international conference on Human language technology research*, pages 1–5. Association for Computational Linguistics, 2001.

[26] B Knight, J Ma, and E Nissan. Representing Temporal Knowledge in Legal Discourse. *Information and Communications Technology Law*, 7(3):199–211, 1998.

[27] Kristin Koch, Judith McLean, Ronen Segev, Michael A. Freed, Michael J Berry II, Vijay Balasubramanian, and Peter Sterling. How Much the Eye Tells the Brain. *Current Biology*, 16(14):1428–1434, 2006.

[28] Daniel Koehler. *Right-Wing Terrorism in the 21st Century: The "National Socialist Underground" and the History of Terror from the Far-Right in Germany*. Taylor & Francis, 2016.

[29] Sherry Koshman, Amanda Spink, and Bernard J. Jansen. Web Searching on the Vivisimo Search Engine. *Journal of the American Society for Information Science and Technology*, 57(14):1875–1887, 2006.

[30] Nikolaos Lagos, Frederique Segond, Stefania Castellani, and Jacki O'Neill. Event Extraction for Legal Case Building and Reasoning. In *International Conference on Intelligent Information Processing*, pages 92–101. Springer, 2010.

[31] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural Architectures for Named Entity Recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[32] Lukas Lange, Omar Alonso, and Jannik Strötgen. The Power of Temporal Features for Classifying News Articles. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 1159–1160. ACM, 2019.

[33] Vito Latora, Vincenzo Nicosia, and Giovanni Russo. *Complex Networks: Principles, Methods and Applications*. Cambridge University Press, 2017.

[34] Jiwei Li and Claire Cardie. Timeline Generation: Tracking Individuals on Twitter. In *Proceedings of the 23rd international conference on World wide web*, pages 643–652. ACM, 2014.

[35] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118108.1118117.

[36] Edimar Manica, Carina F. Dorneles, and Renata Renata Galante. Handling Temporal Information in Web Search Engines. *ACM SIGMOD Record*, 41(3):15–23, 2012.

[37] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT press, 1999.

[38] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Scoring, Term Weighting and the Vector Space Model. *Introduction to Information Retrieval*, 100:2–4, 2008.

[39] Peter V Marsden and Nan Lin. *Social Structure and Network Analysis*. Sage Beverly Hills, 1982.

[40] James H. Martin and Daniel Jurafsky. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson/Prentice Hall Upper Saddle River, 2009.

[41] Ida Mele, Seyed Ali Bahrainian, and Fabio Crestani. Event Mining and Timeliness Analysis from Heterogeneous News Streams. *Information Processing & Management*, 56(3):969–993, 2019.

[42] Diego Mollá, Menno Van Zaanen, Daniel Smith, et al. Named Entity Recognition for Question Answering. *Proceedings of the 2006 Australasian Language Technology Workshop*, 2006.

[43] Franco Moretti. *Distant Reading*. Verso Books, 2013.

[44] María Navas-Loro and Cristiana Santos. Events in the Legal Domain: First Impressions. In *TERECOM@ JURIX*, pages 45–57, 2018.

[45] Mark E. J. Newman. Scientific Collaboration Networks. II. Shortest Paths, Weighted Networks, and Centrality. *Physical review E*, 64(1):016132, 2001.

[46] Azadeh Nikfarjam, Ehsan Emadzadeh, and Graciela Gonzalez. Towards Generating a Patient's Timeline: Extracting Temporal Relationships from Clinical Notes. *Journal of biomedical informatics*, 46:S40–S47, 2013.

[47] NSU Watch. Aufklären & Einmischen, https://www.nsu-watch.info/2013/05/ sitzung-stermine/, accessed 25. Nov. 2019.

[48] Sérgio Nunes, Cristina Ribeiro, and Gabriel David. Use of Temporal Expressions in Web Search. In *European Conference on Information Retrieval*, pages 580–584. Springer, 2008.

[49] David D. Palmer. Tokenisation and Sentence Segmentation. *Handbook of Natural Language Processing*, pages 11–35, 2000.

[50] Paulraj Prabhu. Document Clustering for Information Retrieval - A General Perspective. *Available at SSRN 2190318*, 2011.

[51] James Pustejovsky, Robert Knippen, Jessica Littman, and Roser Saurí. Temporal and Event Information in Natural Language Text. *Language Resources and Evaluation*, 39 (2-3):123–164, 2005.

[52] Dragomir R. Radev and Weiguo Fan. Automatic Summarization of Search Engine Hit Lists. In *Proceedings of the ACL-2000 Workshop on Recent Advances in Natural Language Processing and Information Retrieval: Held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 11*, pages 99–109. Association for Computational Linguistics, 2000.

[53] Estela Saquete, Patricio Martinez-Barco, Rafael Munoz, and Jose-Luis Vicedo. Splitting Complex Temporal Questions for Question Answering Systems. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 566. Association for Computational Linguistics, 2004.

[54] Frank Schilder and Christopher Habel. From Temporal Expressions to Temporal Information: Semantic Tagging of News Messages. In *Proceedings of the ACL 2001 Workshop on Temporal and Spatial Information Processing*, 2001.

[55] Frank Schilder and Christopher Habel. Temporal Information Extraction for Temporal Question Answering. In *New Directions in Question Answering*, pages 35–44, 2003.

[56] John Scott. Social Network Analysis. *Sociology*, 22(1):109–127, 1988.

[57] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake News Detection on Social Media: A Data Mining Perspective. *CoRR*, abs/1708.01967, 2017.

[58] Andreas Spitz. *Implicit Entity Networks: A Versatile Document Model*. PhD thesis, University of Heidelberg, Germany, 2019. URL `http://www.ub.uni-heidelberg.de/archiv/26328`.

[59] Andreas Spitz and Michael Gertz. Terms over LOAD: Leveraging Named Entities for Cross-document Extraction and Summarization of Events. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 503–512. ACM, 2016.

[60] Andreas Spitz and Michael Gertz. Entity-centric Topic Extraction and Exploration: A Network-based Approach. In *European Conference on Information Retrieval*, pages 3–15. Springer, 2018.

[61] Andreas Spitz, Jannik Strötgen, Thomas Bögel, and Michael Gertz. Terms in Time and Times in Context: A Graph-based Term-Time Ranking Model. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, pages 1375–1380, 2015. doi: 10.1145/2740908.2741693.

[62] Andreas Spitz, Satya Almasian, and Michael Gertz. EVELIN: Exploration of Event and Entity Links in Implicit Networks. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 273–277. International World Wide Web Conferences Steering Committee, 2017.

[63] Julius Steen and Katja Markert. Abstractive Timeline Summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 21–31, 2019.

[64] David Stronczek. Named Entity Disambiguation using Implicit Networks. Master's thesis, University of Heidelberg, Germany, 2018.

[65] Jannik Strötgen and Michael Gertz. High Quality Rule-based Extraction and Normalization of Temporal Expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation. Los Angeles, California: Association for Computational Linguistics*, 2010.

[66] Jannik Strötgen and Michael Gertz. Multilingual and Cross-domain Temporal Tagging. *Language Resources and Evaluation*, 47(2):269–298, 2013. doi: 10.1007/s10579-012-9179-y.

[67] Jannik Strötgen and Michael Gertz. A Baseline Temporal Tagger for all Languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 541–547, 2015.

[68] Julien Tourille. *Extracting Clinical Event Timelines : Temporal Information Extraction and Coreference Resolution in Electronic Health Records*. Theses, Université Paris-Saclay, December 2018. URL `https://tel.archives-ouvertes.fr/tel-01997223`.

[69] Giang Tran, Mohammad Alrifai, and Eelco Herder. Timeline Summarization from Relevant Headlines. In *European Conference on Information Retrieval*, pages 245–256. Springer, 2015.

[70] Giang Tran, Eelco Herder, and Katja Markert. Joint Graphical Models for Date Selection in Timeline Summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1598–1607. Association for Computational Linguistics, 2015.

[71] Giang Binh Tran, Mohammad Alrifai, and Dat Quoc Nguyen. Predicting Relevant News Events for Timeline Summaries. In *WWW (Companion Volume)*, pages 91–92, 2013.

[72] Cornelis Joost Van Rijsbergen. A Theoretical Basis for the Use of Co-occurrence Data in Information Retrieval. *Journal of documentation*, 33(2):106–119, 1977.

[73] Maarten van Steen. *Graph Theory and Complex Networks: An Introduction*. 2010.

[74] vis.js. A Dynamic, Browser Based Visualization Library, https://visjs.org/, accessed 24. Nov. 2019.

[75] Misha Wolf and Charles Wicksteed. Date and Time Formats. *W3C NOTE*, page 26, 1998.

[76] Julian Zell and Jannik Strötgen. HeidelTime Standalone Manual Version 2.1. 2015.

[77] Xinyi Zhou and Reza Zafarani. Fake News: A Survey of Research, Detection Methods, and Opportunities. *CoRR*, abs/1812.00315, 2018.