Inaugural-Dissertation

zur Erlangung der Doktorwürde der

Naturwissenschaftlich-Mathematischen
Gesamtfakultät

der

Ruprecht-Karls-Universität
Heidelberg

vorgelegt von

Andreas Spitz (M.Sc.)

aus Wiesbaden

Tag der mündlichen Prüfung:

Implicit Entity Networks: A Versatile Document Model

Andreas Spitz

Supervisors: Prof. Dr. Michael Gertz
Prof. Dr. Kai-Uwe Sattler

Abstract

The time in which we live is often referred to as the Information Age. However, it can also aptly be characterized as an age of constant information overload. Nowhere is this more present than on the Web, which serves as an endless source of news articles, blog posts, and social media messages. Of course, this overload is even greater in professions that handle the creation or extraction of information and knowledge, such as journalists, lawyers, researchers, clerks, or medical professionals. The volume of available documents and the interconnectedness of their contents are both a blessing and a curse for the contemporary information consumer. On the one hand, they provide near limitless information, but on the other hand, their consumption and comprehension requires an amount of time that many of us cannot spare. As a result, automated extraction, aggregation, and summarization techniques have risen in popularity, even though they are a long way from being comprehensive. When we, as humans, are faced with an overload of information, we tend to look for patterns that bring order into the chaos. In news, we might identify familiar political figures or celebrities, whereas we might look for expressive symptoms in medicine, or precedential cases in law. In other words, we look for known entities as reference points, and then explore the content along the lines of their relations to others entities. Unfortunately, this approach is not reflected in current document models, which do not provide a similar focus on entities. As a direct result, the retrieval of entity-centric knowledge and relations from a flood of textual information becomes more difficult than it has to be, and the inclusion of external knowledge sources is impeded.

In this thesis, we introduce *implicit entity networks* as a comprehensive document model that addresses this shortcoming and provides a holistic representation of document collections and document streams. Based on the premise of modelling the *cooccurrence relations* between terms and entities as first-class citizens, we investigate how the resulting network structure facilitates efficient and effective *entity-centric search*, and demonstrate the extraction of *complex entity relations*, as well as their *summarization*. We show that the implicit network model is fully compatible with *dynamic streams of documents*. Furthermore, we introduce document aggregation methods that are *sensitive to the context* of entity mentions, and can be used to distinguish between different entity relations. Beyond the relations of individual entities, we introduce *network topics* as a novel and scalable method for the extraction of topics from collections and streams of documents. Finally, we combine the insights gained from these applications in a versatile *hypergraph document model* that bridges the gap between unstructured text and structured knowledge sources.

Zusammenfassung

Unsere Zeit wird oft als das Informationszeitalter bezeichnet, obwohl eine Charakterisierung als Zeitalter des konstanten Informationsüberflusses ebenso treffend wäre. Nirgendwo sind Informationen so präsent wie im Internet, das eine unversiegbare Quelle an Nachrichtenartikeln und Beiträgen aus den sozialen Medien ist. In Arbeitsfeldern wie dem Journalismus oder der Medizin, die sich mit der Verwaltung oder Beschaffung von Informationen und Wissen beschäftigen, ist diese Informationslast oftmals noch stärker ausgeprägt. Die Menge an verfügbaren Dokumenten ist dabei für den Leser häufig ein Fluch und ein Segen zugleich. Auf der einen Seite bietet sie Zugang zu fast unbegrenzten Informationen, aber auf der anderen Seite erfordern Lektüre und Verständnis einen Zeitaufwand, der kaum zu rechtfertigen ist. Aufgrund dieses Problems hat die Verwendung von maschinellen Extraktions-, Aggregations- und Zusammenfassungsverfahren stark zugenommen, stößt aber an ihre Grenzen. Im Angesicht eines solchen Informationsüberflusses liegt für den Leser oft die Suche nach Mustern nahe, um Ordnung in das Chaos zu bringen. Dies können bekannte Personen in Nachrichtenartikeln sein, Präzedenzfälle im Rechtswesen, oder Symptome in der Medizin. Mit anderen Worten: Wir suchen nach uns bekannten Entitäten als Referenzpunkten und hangeln uns dann an den Beziehungen zu anderen Entitäten entlang, um den Inhalt der Dokumente zu verstehen. Genau dieses Vorgehen wird aber von existierenden Dokumentenmodellen auf technischer Seite nicht unterstützt, da diese keine Entitätsrelationen berücksichtigen. Somit wird die Informationsgewinnung aus unstrukturierten Texten mithilfe existierender Dokumentenmodelle schwieriger als notwendig und die Einbindung externer Wissensquellen verhindert.

In dieser Arbeit führen wir daher *implizite Entitätsnetzwerke* ein, die eine vollständigere Repräsentation von Dokumentensammlungen ermöglichen. Basierend auf der Modellierung von *Kookkurrenzrelationen* zwischen Entitäten und Worten als primäre Komponente des Modells untersuchen wir effiziente und effektive Methoden zur *entitätsbasierten Suche* in Dokumenten sowie der *Extraktion von Entitätsrelationen*. Wir zeigen weiterhin, dass implizite Entitätsnetzwerke auch genutzt werden können, um dynamische *Ströme von Dokumenten* zu modellieren. Basierend auf dem *Kontext von Entitätsrelationen* extrahieren wir *Topics aus Netzwerken* und setzen diese in Kontrast zu Topicmodellen. Schließlich verallgemeinern wir das Modell der impliziten Entitätsnetzwerke zu einem *Dokumentenmodell basierend auf Hypergraphen*, das die direkte Kombination von unstrukturierten Texten und strukturierten Wissensbasen ermöglicht.

Acknowledgements

First and foremost, I would like to thank my advisor, Michael Gertz. I am grateful not only for the opportunity to pursue my PhD, but especially for being able to do so in the excellent research environment that he has created in his group. He has always had an open door and offered guidance whenever I needed it, given me the freedom to find my own path when I (felt like I) did not, and provided the support that allowed me to walk it. I am grateful to my second supervisor, Kai-Uwe Sattler, for constructive discussions in our team meetings during the early phase of my PhD studies.

I would also like to acknowledge the support that I received from the Faculty of Mathematics and Computer Science at Heidelberg University, and from the members of the Database Systems Research group. Special thanks go to Johanna Geiß for the especially thankless job of cleaning and annotating so many of our data sets, and for all our collaborations. My gratitude goes to Jannik Strötgen for getting us started on a paper that ultimately motivated much of the research described in this thesis. I would like to thank Erich Schubert for our collaborations and for always giving helpful advice, sometimes even for problems that I did not know I had (but did). I appreciate the assistance of Satya Almasian on numerous programming and data cleaning tasks over the years, but I would especially like to thank her for the (at times literally) single-handed supervision of the news article acquisition pipeline. My thanks also go to Sebastian Lackner for our work on a network community structure that makes even computer scientists look like social animals in comparison.

Furthermore, I would like to thank the members of the extended Database Systems Research group, including student assistants and collaborators. In particular, I appreciate the very constructive discussions and collaborations that I have had with Gloria Feher and Ludwig Richter. I would like to thank the members of our data mining competition team, Kai Chen, Diego Costa, Johanna Geiß, Jan Greulich, and Stefan Wiesberg, for seeing it through to the final iteration, despite their many other obligations. I appreciate the work with Xiaoyu Chen, who showed us what a successful last-in-first-out PhD looks like. I would like to thank Mohammad Dawas and David Stronczek for their work on our natural language processing pipeline and for their help in data annotation.

My thanks also go to those who took care of the day-to-day tasks that keep everything running. This especially includes Natalia Ulrich and Catherine Proux-Wieland, for keeping everything running smoothly on the administrative level, and our system administrators Niels Bernlöhr, Leonard Henger, and Holger Wünsche for doing the same for our servers.

I would like to thank the people who kindly took the time to read parts of this thesis and provided me with valuable feedback, namely Satya Almasian, Dennis Aumiller, Diego Costa, Gloria Feher, and Sebastian Lackner. Their comments were not just helpful, but often hilarious and funny enough to keep me sane through the long stretches of editing.

Finally, my thanks go to many others who have contributed to my endeavour over the years and who shall remain nameless, not out of lack of gratitude, but out of respect for anonymity in the face of implicit networks. This includes the ever changing cast of the Botanik Philosophers, who made me join for the coffee, and got me to stay for the conversation. Of course, last but far from least, I would like to extend my deepest gratitude to my family and friends for moral (and culinary) support during the last four years.

Contents

1 Introduction

Imagine, if you will, a world without Wikipedia. A world without knowledge bases, dictionaries, or encyclopedias that provide definitions of any word, entity, concept, or relation between them at the click of your mouse or the tap of your finger. Now before you dismiss this idea as the absurdly far-fetched nonsense that it is to anyone with a computer, smartphone, or any other means of accessing the Web, consider the perspective of a journalist, for example. Consider the perspective of someone who is not retrieving knowledge about something that is known, but looking for insight into something that is not yet known.

Even in such a world, there would still be search engines that enable you to identify relevant documents to any question you might have (minus, of course, the one blue link to Wikipedia at the very top of the results that you would usually select). Unless you are searching for something obscure, however, you are likely to be presented with thousands or millions of results, especially if your question was vague because you do not yet know what you are actually trying to find. Just like, for example, a journalist who is trying to understand what is going on in a large collection of documents that was just leaked by some anonymous source.

How do you cope with such a flood of information? How do you distil the bits and pieces that are relevant to your information needs from a wall of text that consists of document after document? Your first instinct might be to read all of the documents, or at least to read them sequentially until you find the relevant piece of information. Unfortunately, this might not be a viable option. In a more probable scenario, you are like the journalist who has to make a deadline or beat the competition to the story, and would rather not spend this time if it can be avoided. After all, it is not just journalists that are trying to uncover new insights and knowledge from texts, but also data analysts, paralegals, and, not least, researchers and scholars. If the amount of documents is too substantial to read, how can one then approach an investigation? Here, automated approaches that aid the reader in selecting their reading material are a potential answer.

Consider, for example, a journalist who is working on the Panama Papers [102], or a data analyst who is tasked with finding information to aid Robert Mueller's investigation

into collusion during the U.S. election of 2016. In these and many other investigative professions, there might simply be no descriptive a priori knowledge that could be used to determine valid starting points. Instead, the structure is likely to emerge during the investigation as the relations between entities such as persons, organizations, or other concepts are discovered. It is this structure that guides the investigation from person to person or from concept to concept. Therefore, it seems sensible to consider the relations between those entities and the structures that emerge from their relations. Or, in other words, it makes sense to use the network of entity relations that is implicitly contained in the documents to put the relations themselves into context.

Based on this intuition, we consider how to assist investigators in the exploration of large collections or even streams of documents, by describing novel methods for the extraction, representation, and utilization of such network structures. We show how these networks can be used to represent the entirety of the documents and support a multitude of potential retrieval, exploration, and visualization operations to aid an investigation into the documents' content. We revisit the example of journalists and analysts throughout the following chapters to highlight the potential of this model in practical applications.

1.1 Implicit Entity Networks

The worst-case scenario that we described above raises the question why the complete lack of a knowledge base like Wikipedia even constitutes a major stumbling block for the investigation of a document collection. The reason that such a lack is problematic, is our reliance on entities and concepts in descriptions and discussions. Whether we talk about events that happen to people, at some locations, or at some time, or whether we discuss more abstract concepts, entities constitute the core of descriptions. Whenever we encounter familiar entities in a text, they provide us with context in the form of some content that we already understand. Thus, these known entities act as a starting point for understanding the remaining content that we do not yet understand. In cases where we know nothing about any of the involved entities, trying to understand the content is difficult. Therefore, it is not just entities that capture our attention and aid our understanding, but also their relations.

However, a priori knowledge is not strictly a necessity for developing an initial understanding. When we encounter the same entity in a different context, we can start to investigate relations between those contexts. This idea is reflected in the much broader

context of linguistics by J.R. Firth's well known quote *"You shall know a word by the company it keeps"* [62], which indicates that some of the most discriminative features of a word are the other words that occur in its proximity. The use of *company* is interesting in that it emphasizes an aspect of connectedness and includes a social connotation, which immediately suggests a network of sorts. After all, networks are a natural model of choice for representing things that are connected [17].

What we can take from this observation for the analysis of the content of documents, is the potential for constructing relations between words and entities or among entities from nothing but their cooccurrences in the documents. The individual relations that repeat in different contexts then form complex structures when they are combined across all available documents, and describe the relations of entities relative to the remaining content. As a result, each entity itself acts as the context of other entities, and the network as a whole represents the threads that need to be unravelled to obtain an understanding of the document collection, or relevant aspects thereof. Due to its conception, we therefore refer to such a network as an *implicit entity network*, in which relations are derived from implicit cooccurrences instead of being explicitly specified.

The question, of course, is how such a network can be constructed to enable the user to uncover relations and gain insight, and what retrieval operations on the network can be used to support them. This is the challenge we address in the remainder of this thesis.

1.2 Challenges and Contributions

As we have motivated above, the representation of document collections as implicit networks can serve as a bootstrapping approach for inducing structure in otherwise unstructured texts through the extraction of implicitly given relations between entities in the text. By using this network structure, the discovery of relevant relations or an exploration of the documents' contents then becomes possible. Such a task, of course, comes with a number of technical challenges that we discuss in the following, before briefly summarizing our contributions that address them.

1.2.1 Challenges

The challenges that arise from the modelling of entity-centric cooccurrences as a network are numerous, and range from algorithmic aspects of an efficient implementation to the interactive usability of a resulting system. To identify viable angles of addressing these issues, we identify six primary challenges.

How can relations be extracted?

The most obvious among the challenges concerns the extraction or generation of a network from cooccurrences that encodes relations between the involved entities and words. The literature has numerous models that consider word cooccurrences in some way, such as collocations [56], learned word embeddings [22], or knowledge extraction [211]. However, most of these models are designed with specific applications in mind, or based on assumptions that may not be generalizable. Thus, the first challenge is to identify related approaches that could provide helpful insights into obtaining relations from plain text, and determine their viability. Based on these insights, we can then proceed to design a model that captures entity and word relations in a meaningful way, and has the necessary properties to support an efficient interaction with the data.

How can relations be described or visualized?

From the perspective of an investigator of the data, the interaction with the model is central. Beyond the mere extraction of relations, their visualization is a key aspect of the usability. However, not only the relations themselves are of interest, but also their descriptions, the topics in which they are used, and how they can be explored. Especially for a bootstrapping approach in which relations are used to gain insights into the contents of the documents, the extraction of descriptions and explanations for relations and participating entities from the documents is necessary. Effectively, this amounts to an extractive summarization of content from the network [142]. However, since the network structure is itself a description of relations, this challenge is not limited to textual summaries, but extends to the extraction and visualization of informative subgraphs around relations.

How are the networks handled efficiently?

Given the potential size of document collections, such as news streams with thousands of articles per day or Wikipedia with millions of documents, the number of distinct words that are contained within them is enormous. A network that models all possible relations would therefore be extremely dense, and even a network covering only the relevant relations is still likely to contain billions of edges. This raises the question of how the data can be represented and processed effectively and efficiently to enable user interactions with the network. Thus, we need to address the challenge of logically and physically representing the network to support the subsequent retrieval of relations. The question of efficiency then extends to the extraction of the network, and to queries to the data.

How to handle streaming data?

When one considers a corpus or collection of documents, its contents are rarely created on a single day, but instead emerge over time and are combined only after the fact. While this may not appear relevant in some cases, such as a collection of novels, there are numerous examples in which the temporal dimension is an important aspect of the data, such as news articles, social media posts, blogs, or proceedings of the regular meetings of government bodies. As a result, the data can be considered as a stream of documents, rather than a collection. In the most extreme example, it could be a live stream in which new content is continuously produced, such as social media posts. This application scenario, of course, is highly relevant to an investigate scenario. Therefore, the third challenge concerns the modelling of streaming data, in regard to both processing and representation, as well as querying the data in a temporal dimension.

How can we account for the context of relations?

The occurrences of rarely mentioned entities are likely confined to a single context in a collection of documents, and their infrequent relations are unlikely to be ambiguous. More frequent pairs of entities, however, which might cooccur in multiple different contexts, are likely to participate in more than a single relation. Consider, for example, co-workers that have a shared hobby outside of work and are thus connected by at least two different relations. To disambiguate these relations, it becomes necessary to distinguish between the contexts of the mentions. Based on a network representation, the challenge then concerns how individual mentions of relations can be attributed to specific contexts, and how to determine when such contexts overlap and the relations can be considered equivalent.

How to ensure compatibility with previous approaches?

Since there are numerous methods for retrieving specific bits of information from documents, a key issue is the compatibility with these existing methods and their underlying models. Without such a compatibility, the network model could provide only a subset of potentially useful functionality, and make the use of multiple parallel models necessary. If, however, the model can be designed in such a way that it is compatible with other approaches to retrieving information, it becomes more flexible. Therefore, we see the final challenge in ensuring that a network representation not only supports network-centric tasks, but also readily enables a transition to other models and methods.

1.2.2 Contributions and outline

To address these challenges, we introduce implicit networks as an entity-centric document model for large and heterogeneous document collections or document streams. Specifically, we make the following contributions to advancing the analysis of entity relations in such a setting. Since our contributions are largely aligned with the structure of the thesis, we present them jointly.

I To approach the challenge of identifying viable strategies for constructing an entity-centric and network-based document model, we review the related literature in Chapter 2. We focus on the requirements of a suitable model for the exploration and analysis of large document collections, and discuss the specific aspects that such a model needs to satisfy. By putting these requirements in the context of existing models and methods, we identify the building blocks that are needed to construct such a holistic document model.

II Based on these building blocks, we proceed to the definition of a suitable document model. We formally introduce the **implicit network model** in Chapter 3, where we also provide a first evaluation of its performance on Wikipedia as a large document collection and against a state-of-the-art baseline. Furthermore, we demonstrate that the versatility of this network-based approach indeed supports a variety of **retrieval tasks** that are centred on entities, their relations, or their context(s).

III In the following chapter, we demonstrate the efficiency and effectiveness of the implicit network model on two concrete application examples. In the first part of Chapter 4, we describe the construction of an **entity-centric search engine** that enables the user to interact with the underlying document collection in near-real time, including the **extraction of subgraphs**.

IV In the second part of Chapter 4, we address **relation summarization** as an application by focusing on the extraction of sentences from the network. In addition to the bootstrapped generation of descriptions for entities and entity relations from their contexts, we demonstrate how implicit relations can be used to **identify complex entity relations** that are missed by traditional knowledge models.

V To account for document data that is not stored in static collections but accessible from a stream of documents, we consider the extension of our model to a **dynamic implicit network model** in Chapter 5. By attributing entity mentions with the context in which they occur in individual documents, we then introduce the **context-based merging** of relations to account for different types of relations. Using this

extended model, we show how a timeline-based analysis enables us to investigate the **evolution of entity and relation contexts** over time.

VI In a generalization of relation contexts, we address the extraction of descriptive subgraphs around specific relations in Chapter 6. As a result, we show how a **global ranking of entity relations** can serve to **identify relevant topics** in multiple entangled streams of documents, and how they enable a comparison of these streams or even of individual topics over time. Based on a comparison to traditional topic models, we highlight the flexibility and increased descriptiveness of network topics. In the second part of Chapter 6, we describe the implementation of an interactive user interface for the extraction, comparison, and **visualization of network topics** from a stream of news articles.

VII Having introduced the implicit network model for static and dynamic document collections, and having considered several application scenarios, we take a step back and consider the modelling of arbitrary word and entity cooccurrences. In Chapter 7, we therefore consider a generalization of the model to a **hypergraph document model** that allows us to fully **merge structured and unstructured data**. Based on this model, we demonstrate how not just implicit networks but a variety of established information retrieval methods can be implemented in a unified framework by using a small set of **hypergraph operators**. Furthermore, we discuss the practical implications of utilizing this hypergraph model, and give a first outline of the technical details of its realization.

Finally, in Chapter 8, we summarize the contributions that we make to the model, discuss the practical implications for the investigation of large collections or streams of documents, and give an overview of open questions and future research directions.

Our contributions in this thesis are based on and expanded from a set of peer reviewed publications, which are listed at the beginning of the respective chapters.

1.2.3 Notation

As a final note before we begin the discussion of related work, we give a brief intuition of the used notation. In the following, we typically use capital Latin and Greek letters to denote sets, such as documents, nodes in a network, or graphs. Most lower case Greek letters denote functions, and are primarily used to map edges or nodes of a graph to an attribute value, or to denote scoring functions that map to the set of real numbers. Lower case Latin letters or strings typically denote parameters of algorithms, or general functions. A complete list of the entire notation can be found in the glossary.

2 Background and Related Work

As a versatile representation of document collections for the entity-centric retrieval of information, the implicit network that we propose in this thesis touches on numerous applied tasks, such as document search and retrieval, summarization, topic detection and exploration, relationship extraction, or event detection. Since these applications are specialized and are relevant only in the localized context of the chapters in which we apply our document model to solving them, we discuss the specific details of those applications and the related work in the respective chapters. In the following, we instead focus on the basic building blocks that are necessary for extracting information from large document collections, on document models, and on general related work.

Structure. In Chapter 2.1, we introduce the basics of document segmentation strategies and tools, as well as models for the representation of documents and document collections. Afterwards, in Chapter 2.2, we briefly discuss related work on collocation analysis, which shares some similarities with our network-based approach. In Chapter 2.3, we introduce the concept of graphs and how they can be used to model and construct heterogeneous information networks, including large-scale knowledge bases. Since our proposed model is entity-centric, we describe the process of annotating and linking entities in Chapter 2.4. Based on this background, we then motivate the necessity for an entity-centric representation in Chapter 2.5.

2.1 Natural Language Processing and Document Models

The extraction of networks as representations of the content of documents is dependent on the recognition of words and entities in the text, and therefore on tools from natural language processing (NLP). As a field, natural language processing overlaps with the field of information retrieval (IR). Where the aim of the former is to model, understand or even reproduce the content and structure of human language, the latter aims to index, locate, and retrieve pieces of information from such content. For our work in the following, we are most interested in the aspect of structuring and modelling language on a basic level,

which then supports the construction of representative data structures and the retrieval of information from the content of documents. Thus, we focus on the basic principles of linguistics and automated language processing that allow us to segment the content of documents into individual components. For a brief historic synopsis and an in-depth overview of recent advances in the field, we refer to a review by Hirschberg and Manning [90].

2.1.1 Language segmentation and annotation

We begin with the segmentation and annotation of natural language texts, which forms the basis of subsequent all modelling. These steps are designed to recover the syntax of the documents, and are typically based on linguistic insights into the structure of the language at hand. As a result, they often (but not always) utilize a rule-based approach, in particular for the more basic steps or for low-resource language where insufficient training data is available for learning-based approaches. The steps that we describe in the following typically build on each other and are applied as a chain that creates a natural language processing pipeline. However, the difficulty and necessity of some steps may vary, depending on the language or language family. For example, the task of segmenting a text into words, which is typically rather simple for many European languages, is quite difficult for the Chinese language [186] due to its lack of whitespace delimiters. While the network representations that we introduce are language agnostic, we focus on English as the language of our document collections in this thesis. In the following, we describe the basic natural language preprocessing steps with this application in mind. Furthermore, we only give a brief overview over an area that cover decades of research into (computational) linguistics, and refer the reader to textbooks on the topics for further details [125, 212].

Sentence detection

The likely most fundamental task in the automated processing of language is the segmentation of a text into sentences, formally called sentence boundary detection. Since sentences can be viewed as meaningful units of thought that convey the basic structure of the text, their detection is the first step in structuring previously unstructured texts. For European languages, the detection of sentences is relatively unproblematic due to the use of punctuation, and is often regarded as a solved problem due to the extremely high performance on evaluation data sets. In practice, however, due to widely different content, the observed performance may drop significantly (for a more detailed review, see [153]). Since we primarily use sentences as a measure of the distance between words in our model

(see Chapter 3.2), this issue is less problematic for us, and we therefore rely on established tools for the detection of sentence boundaries as discussed there.

Tokenization

Once sentences are detected, the next smaller (and in many cases atomic) unit of speech are tokens. In European languages, tokenization is typically performed by splitting sentences based on whitespace delimiters, which breaks down a sentence into individual tokens. In the following, we refer to these tokens as *words*, in contrast to more complex multi-word expressions such as *named entities* or *terms*. Additionally, some special cases may be considered, in which different delimiters occur. For example, the colloquial English contraction *it's* consists of the two words *it* and *is*, which are not separated by a whitespace delimiter, but by an apostrophe. Especially when the tagging of parts-of-speech is of interest, disregarding *is* as the verb of a sentence would be ill-advised.

Part-of-speech tagging

Once tokens have been detected as the individual building blocks of a sentence, they can be further annotated with their function in the sentence, namely their part-of-speech. For example, in the sentence *They tag parts of speech*, the token *tag* would be annotated as a verb, whereas it serves as a noun in the sentence *Look at this tag*. As one might guess from this example, parts-of-speech are useful in subsequent processing, annotation, and extraction steps since they provide additional information about the sentence's components, and can be used to disambiguate potential meanings of individual tokens. As a result, extensive research has been devoted to the annotation of parts-of-speech, which is an ongoing topic of research interest. Typically, parts-of-speech include further linguistic components beyond words, such as punctuation, or even non-word utterances in spoken language annotation that are less relevant to our cooccurrence-based models. Several tagsets exist for English, as well as methods for the automated induction of part of speech tags [45]. Here, we adopt the widely used English tagset from the Penn Treebank [127].

Chunking

Finally, chunking describes the process of reconstituting multi-word expressions from the individual tokens. For example, consider the sentence *In Paris, I visited the Eiffel Tower*, in which the combination of the two tokens *Eiffel Tower* constitutes the name of a landmark

in Paris, and they should therefore be treated as a chunk. As is evident from this example, considering individual tokens is insufficient for the detection and extraction of (named) entities from a text. Therefore, chunking is a vital step in the natural language processing pipeline for any entity-centric model. However, for the ease of readability, we do not refer to tokens or chunks in the following. Instead, we simply refer to the atomic blocks of processed texts as *terms*, and assume that they have been chunked where necessary. Thus, the set of terms describes a superset of tokens and chunks.

A final step in the natural language processing pipeline is the annotation of chunks as entities or named entities. In contrast to the previous steps, this is a semantic annotation. Due to its importance for our model and the need for external knowledge bases, we dedicate an entire section to this step of annotating entities (see Chapter 2.4).

Stemming and lemmatization

A common problem in processing an annotated collection of documents is the matching of the vocabulary, where some words may be conjugated or declensed depending on the context, or may be derived from a common root and thus sufficiently similar to be treated as a single word. Consider, for example, the sentence *They run through the streets of Florence*, in which the verb is *to run*, and consider similar sentences in which the grammatical number of the subject or the tense is changed, so that the conjugated verb form might instead occur as *running*. While these sentences are slightly different, the conveyed idea is the same and the nuances may not matter in practice. In those cases, stemming is a helpful tool that reduces such words to a common word stem such as *run*, for example by stripping the suffix. For a more detailed introduction to English stemmers, see [123]. While stemming algorithms tend to be inherently language specific, there are implementations available for many Languages, such as the Porter stemmer [149].

Unfortunately, stemming is prone to errors in cases with irregularities, especially for verbs. Consider the past tense variation of the above sentence *They ran through the streets of Florence*. Since there is no common prefix to *run* and *ran* beyond the letter *r*, stemming is unlikely to produce a meaningful root that matches both words. In such cases, lemmatization is typically a better solution, which takes the part-of-speech of the word into account to improve the performance and map both words to the so called lemma of the word (in this case, *to run*). While lemmatization is more powerful, it also has drawbacks in comparison to stemming. Where lemmatization utilizes part-of-speech tags and the context of a word, stemming does not, meaning that it requires additional annotations as input.

Therefore, while lemmatization seems preferable in many applications, there are some settings in which this reliance on part-of-speech tags renders lemmatization infeasible, as we discuss for the case of implicit networks in Chapter 3.3.

In contrast to the above approaches, one can also entirely forego lemmatization or stemming and cluster semantically related words based on vector embeddings, which we discuss in Chapter 2.1.3.

2.1.2 Vector space representation of documents

The vector space model is a representation of term occurrences in documents that is primarily of historic significance, but provides a basis and intuition for many of the related works that we discuss in the following. We only give a brief overview, and refer to textbooks for further details (for example, see [125]).

From bags of words to vectors

To represent the text that is contained in a collection of documents in numeric form, the vector space model discards the word order and treats documents as sets of words (typically, multisets are considered, which are also called *bags*, hence the name). Each element in this set then corresponds to one component of a vector whose dimension $|W|$ equals the size of the vocabulary W in the collection. In order to obtain a representation of the content of documents, each document d is treated as a $|W|$-dimensional vector v_d. Entries of v_d equal zero if the corresponding word (or term) is not contained in the document, and are non-zero if the word is contained. In the most simple form, one could consider a binary vector whose entries equal 1 if and only if the corresponding word is contained in the document, but more sophisticated approaches are viable, as we discuss in the following. Intuitively, the vector space model thus represents each document as an index vector that encodes for each term whether it occurs in the document. A document collection then becomes a collection of vectors that allow the computation of the similarities of documents or the importance of their contents.

Term weighting

Index vectors with binary entries as described above have the downside of discarding the frequency information of terms, which can be useful in estimating the relevance of a term for a given document. Thus, instead of using binary entries, one could also use integer

values that count the frequency of each word of a document. This value is typically referred to as the term frequency $tf(t, d)$ and denotes the frequency of term t in document d. This, however, is prone to creating an imbalance in the vectors, since some words are simply more frequent than others, while not necessarily being more important. For example, consider common stop words such as articles, prepositions, or conjunctions, which add little content to a document but occur frequently. Thus, it makes sense to normalize the term frequency by an overall frequency of the word across all documents in the collection, which is denoted by $df(t)$. Since we want to encode greater word importances as larger values, it makes sense to consider the inverse of the document frequency $idf(t)$ for normalization. A commonly used version uses a logarithmic scaling and defines the inverse document frequency as

$$idf(t) := \log \frac{|W|}{df(t)}. \tag{2.1}$$

The normalization of the term frequency with this inverse document frequency is then referred to as the *tf-idf* score [173], and defined as

$$tf\text{-}idf(t, d) := tf(t, d) \log \frac{|W|}{df(t)}. \tag{2.2}$$

The resulting score can be used as component values of the document vectors. In contrast to simple frequency vectors, these normalized vectors now encode a sort of relevance information, since the component values are highest for terms that occur frequently in a small number of documents, and lower for terms that are either infrequent in the given document or frequent across many documents. This weighted vector representation then enables a scoring and comparison of document vectors.

Vector similarity

Based on the representation of documents as vectors, we now have a direct means of computing the similarity of documents as a similarity of vectors. Since each dimension of the vector space corresponds to one specific term and the component values of each document vector denote the weight of this term for the document, an intuitive approach is to use the angle between two vectors to derive a similarity. The more similar the content of the documents is, the smaller the angle, and vice versa. This is captured by the cosine similarity, which is given for two document vectors v_{d_1} and v_{d_2} as

$$cos(v_{d_1}, v_{d_2}) = \frac{v_{d_1} \cdot v_{d_2}}{\|v_{d_1}\| \, \|v_{d_2}\|}, \tag{2.3}$$

where · denotes the inner product of the two vectors, and || || is the Euclidean norm. While we introduced it for the similarity of document vectors here, the cosine similarity is also frequently used to compare vector embeddings as discussed in the following, or to measure the similarity of graph neighbourhoods [184].

The retrieval of information from document collections is then enabled by modelling queries as small documents in the same vector space as the documents in the collection. By comparing the vector that represents the query terms to the document vectors, the documents can be ranked according to the computed similarity score, and presented to the user as a ranked list of documents by descending relevance to the query. In Chapter 3, we use a similar intuition for graphs instead of vectors, by representing the joint content of the document collection as a graph, in which we then search the local neighbourhood of nodes that correspond to query entities and terms.

2.1.3 Word embeddings

In contrast to the extremely large yet sparse vector space of the vector space model, whose dimensionality equals the number of words in the document collection, one can also consider vector spaces of reduced dimensionality. Based on this idea, vector embeddings of words are smaller, often learned representations of words with dimensions that typically range from 50 to 1,000. In addition to the reduced dimensionality, an underlying goal is the representation of similar or related words at closely situated points in the vector space. The label *embedding* stems from this process of placing and arranging the words in the vector space. Due to the reduced dimensionality, the resulting vector space is denser and easier to cluster. Since the embeddings of similar words occur at close proximity in the embedding space, it furthermore makes sense to perform such clusterings, which can be used to identify synonyms or words with similar meaning or function. Ideally, such embeddings can also capture semantics beyond similarity, and represent them as arithmetic operations on the vectors to allow vector-based reasoning. Thus, word embeddings serve a variety of tasks that benefit from the semantic relations between the represented words.

The current style of learning embeddings with neural networks dates back to the work by Bengio et al. [22], prior to which Latent Semantic Analysis [48] was used to generate word representations with reduced dimensions by decomposing occurrence matrices. The current literature on word embeddings is constantly evolving and too extensive to cover here. In the following, we therefore focus on two well known methods that we also use in our later models and evaluations. For further related work on this topic, we refer to the articles by Levy, Goldberg, and Schnabel et al. as starting points [75, 116, 162].

Word2vec

Unlike the sparse vector representation of the vector space model, word2vec as introduced by Mikolov et al. [131] takes a distributed approach. Components of the vectors no longer correspond to words, but represent abstract, continuous-valued descriptors. As a result, the representation is spread (or distributed) across all components of the vector. To train such vectors, the model proposes two different approaches, namely the continuous bag-of-words (cbow) and the skip-gram. Both approaches rely on the unsupervised training of shallow neural networks from the cooccurrences of words in context windows, which produce the embeddings as a byproduct of the training process.

Cbow. The continuous bag-of-words method takes a sliding window approach over the input data, centred on a focus word, with a fixed number of context words to each side. The neural network is then given the context words as input and trained to predict the focus word as output. After the training phase, the embeddings can be derived from the learned weights of the neural network.

Skip-gram. The concept of skip-gram is directly inverse to cbow. Instead of using the context words as input and training the network to predict the focus word, the focus word is used as input and the network is trained to predict the context words. After training, the embeddings are again derived from the weights of the neural network.

Beyond this neural network architecture, a key contribution of the word2vec approach is efficiency, since it includes several optimizations that allow it to be trained on large-scale data, thereby making the process viable. However, one of the primary limitations is the focus on a window-based approach that retrieves word cooccurrences from a localized scope for each occurrence, and thus fails to capture longer-distance relations.

GloVe

In contrast to word2vec, global vector embeddings (GloVe) aim to capture the global cooccurrence statistics of words in the document collection [146]. To this end, the model employs a different target function for training the embeddings that learns embeddings in such a way that they can be used to predict cooccurrence ratios. To train the model, data from the document collection is not streamed in a window based approach, but aggregated to a cooccurrence matrix prior to the training. The cooccurrence ratios from this matrix are then used to optimize word embeddings such that the dot product of two embeddings approximates the logarithm of the words' co-occurrence probability. Despite these differences in construction, the resulting embeddings of word2vec and GloVe tend to perform similarly well in many applications.

2.1.4 Graph representations of documents

Finally, there are two approaches to modelling documents that are based on graph structures, similar to the model that we propose (for a brief introduction to graphs, see Chapter 2.3). In the following, we briefly discuss these models and their limitations, in particular for the representation of documents with entity annotations.

Graph-of-word

The graph-of-word model [157] is a representation of documents that is designed for ad-hoc information retrieval tasks, and is thus similar in scope to our approach. Designed specifically in contrast to the bag-of-words model underlying the vector space model that disregards the sequence in which words appear in a text, the graph-of-word model utilizes a directed graph to capture the order of term occurrences in a small sliding window. However, the graph representation is entirely unweighted since the authors note that a version with cooccurrence weights performed worse in their evaluation. For the implementation of retrieval operations, the model instead uses the degree of nodes in the graph. Specifically, the model proposes an adaptation of the classic *tf-idf* score [173] to a *tw-idf* score that replaces the term frequency *tf* with a term weight *tw* that is derived from the indegree of nodes. This approach is conceptually similar to the adaptation of the *tf-idf* score that we use for the normalization of edge weights in Chapter 3.2, albeit less advanced in the use of relation information from the texts.

While the graph-of-word model takes an important step in the direction of modelling document collections as graphs, it has two shortcomings in the representation of word and entity relations, namely the window size and the edge directions. The window size is critical due to its limiting scope. Since entity mentions tend to be rare in documents, sliding window approaches that cover a small number of terms tend to miss relevant entity relations that cross sentence boundaries. However, simply scaling up the window sizes leads to extremely large models (or dense graphs in the case of graph-of-word) that cannot be handled efficiently. The second issue is even more severe, since the use of directed edges makes an implicit assumption about the syntactic structure of the modelled language. While it is true that word order matters, especially in English, this is less true in different languages. Consider, for example, the case of Hungarian where the word order is very free and often used to denote emphasis. For the extraction of entity relations, using directed edges seems even more questionable, since there is no prescribed order in which arbitrary entities should occur. Consider, for example, the sentences *Copenhagen is the*

capital of Denmark and *The capital of Denmark is Copenhagen*, which are semantically identically and only syntactically different. Since natural languages are not linear languages, attempting to model the syntactic relation with directed cooccurrences seems ill-advised, especially if it comes at the cost of losing parts of the semantic relations. Therefore, the directed graph-of-word model is not suitable for modelling entity-centric relations.

Text topology graphs

A second work that uses similar concepts to our approaches for the extraction of implicit networks is presented by Blanco and Lioma, who investigate the integration of topological measures of the structure of a cooccurrence network for the derivation of node rankings. The work extends on decades of research into graph representations of text, and also gives a good review of this prior work [26]. In contrast to previous works, their model aims to include discourse properties by mapping them to the topological properties of the network, such as centrality measures, average path lengths, or clustering coefficients, and they analyze the fitness of structural graph metrics for the task of document ranking.

Similar to the graph-of-word approach, rankings of nodes are computed in this model by replacing the *tf* term of a *tf-idf* computation by network-based measures, but include the unchanged inverse document frequency for normalization. As we show in Chapter 3.2, it makes sense to go a step further and also replace the *idf* term by a network-based measure for the derivation of localized term rankings, when the focus is a local exploration of relations rather than a recommendation of documents.

2.2 Word Cooccurrence and Collocation Analysis

Many approaches that are related to our model originate from the fields of corpus analytics, Web science, or text mining, and are concerned with the analysis of word cooccurrences or word collocations. The difference between cooccurrence and collocation is subtle and may vary between sources. A commonly accepted interpretation of word *cooccurrence* is the joint occurrence of words, whereas a *collocation* is a cooccurrence of words within a short distance from each other that is recurring (significantly) more frequently than expected [170]. The concept of collocation is typically found in corpus linguistics, where it is based on the Firthian principle that words are defined by the words that frequently occur in their context, and serves as a basis for research into word associations. In contrast, many text- and data mining approaches consider the more general cooccurrences to detect

patterns or relations, often in a context of networks (for a brief introduction to complex networks and graphs, see Chapter 2.3). In the following, we first consider related work on cooccurrences in general, before discussing collocations.

2.2.1 Word cooccurrence networks

Modelling and analyzing the cooccurrences of words has a long tradition that dates back at least to Van Rijsbergen, who proposed to drop the assumption of word occurrence independence in favour of measuring word dependencies with non-linear weighting functions [204]. Since graphs are a natural and intuitive way of representing sets of connected things, many works in the analysis of word cooccurrences use such a representation for a variety of applications. Some recent works on modelling and exploring term cooccurrences use networks to describe term relationships in a context-oriented framework, and employ network analysis techniques to derive measures from the network or to compare language specific networks [38, 44, 118, 119]. Others exploit the properties of such networks to learn document representations and context-dependent relationships through embeddings [195]. However, these types of networks typically consider words or terms without associating them with additional semantics, and are therefore ill-suited to handling entity-centric tasks. Due to the potential density of word cooccurrence networks, these approaches often model term relations in very narrow cooccurrence windows.

More recently, typed cooccurrence networks have been introduced, which include cooccurrences of (named) entities that are detected and extracted from text documents. Nodes still represent terms, but may also be associated with entity types, such as *person*, or *organization*. Examples of such networks include entity graphs used for identifying entities that participate in trending events [160], time-term association graphs used to estimate the focus time of documents [103], or even the cross document co-reference resolution based on spectral graph clusterings of mentions [51]. A different approach, which focuses on the broader word category of concepts rather than on entities, is designed to determine the semantic relatedness of documents from extracted concept graphs [145]. As is evident from these descriptions, these works cover vastly different applications, for which they are specialized. As a result, the underlying network structures tend to be highly optimized for a singular purpose, and are not flexible enough to act as document models.

There are, of course, some approaches that take a more direct approach to cooccurrences and do not focus on a graph representation, but instead employ cooccurrence statistics directly. An example of such an approach is SigniTrend, which uses word cooccurrences to detect events in text streams as spikes in the timeline [164]. However, the relation to

graphs is still evident, since the relations between cooccurring words can be interpreted as weighted edges, and have even been applied directly to the visualization of significant cooccurrences in text streams as graph-like word clouds [163]. A downside of this approach is the minimalistic representation of cooccurrence counts that enables its high efficiency on text streams with a large volume, but also prevents its use for search or exploration applications on the history of the stream as a collection.

Finally, network-based word cooccurrence analysis also motivated some of our previous work that influenced the implicit network model we present here, which includes the extraction of social [69] and location [70] networks from Wikipedia. In these works, entity relations are counted as simple cooccurrences, and weighted directed edges are derived from cosine weights of the node-edge incidence matrix. In contrast to the implicit network model, these works are focused on analyzing the topological and community structure of such cooccurrence networks, instead of the contents of the documents.

A direct predecessor of the implicit network model that we present here is the analysis of term-date networks in the text of Wikipedia [185], which restricts the exploration of word relations to the bipartite case of dates and terms. Thus, it can be seen as a special case of the model that we introduce in Chapter 3.

2.2.2 Word collocations

In contrast to the cooccurrences of adjacent words, word collocations are more subtle. While there are multiple definitions of what constitutes a collocation and the label is used more freely in natural language processing than in linguistics or corpus linguistics, a popular interpretation defines them as composites of words whose semantic and syntactic properties exceed what can be predicted from their components [56]. While this interpretation is (intentionally) vague, it highlights a requirement that goes well beyond the analysis of cooccurrences, which can only serve as indicators but not evidence of collocations. Due to this variety of definitions, most of which are too removed from our work on implicit networks, we only focus on a few specific examples to motivate why collocations are literally too narrow to account for entity relations. For an in-depth discussion and historic background of research into collocations, we refer to the thesis by Evert [56].

For the analysis of collocations, a typically considered window size of cooccurrences lies in the range of two to five words. For the relation of entities, such a cooccurrence window size is too strict, since it is unlikely to find entity mentions at such close proximity. In fact, if the use of pronouns is considered, it is likely that two related entities do not even occur

in the same sentence. As a result, we expect collocations to play a more significant role for word embeddings due to their similarly window-based cooccurrence extraction, rather than for implicit networks of entities. There are, however, some works on collocations that are at least conceptually related to our model in Chapter 3, since they rely on graph representations of collocations [15, 35] or even on the notion of edge direction [78], similar to the graph-of-word approach. However, in contrast to the networks that we consider, these graphs are intended to function on a much smaller scale and a linguistic level that is much closer to the individual words or terms.

A noteworthy aspect of cooccurrence and especially collocation networks, such as the examples above, is the process behind their emergence in contrast to their construction. This contrast has raised the question if such networks should indeed be considered as complex networks [232]. While it is obvious on the surface that networks can be constructed from word adjacencies and word proximities, the underlying generation process (that is, natural language) is different from those of commonly considered complex networks in network analysis [144], which often include some notion of conceptual or physical flow along edges. As a result, the application of the typical metrics from network science may be lacking a formal basis, and should therefore be considered critically and in the context of the application. In our contributions, we thus focus on local relationships in the implicit networks, instead of flow-based global network metrics or indices.

2.3 Graphs and Heterogeneous Information Networks

Since our implicit network model is a graph structure, we consider some of the basic terminology for graphs and networks in the following, and discuss related work on knowledge graphs. Knowledge graphs in particular are interesting due to their dual overlap with our own work. On the one hand, they serve as resources for the annotation of entities as we discuss in Chapter 2.4. On the other hand, the underlying principles of relation and information extraction from unstructured text is conceptually similar to the intuition behind implicit networks, albeit different in methodology.

2.3.1 Foundations of graphs and networks

The terms *network* and *graph* are often used interchangeably, which we also do to some degree in the following. Formally, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical construct that consists of a set of *nodes* $\mathcal{V}$ (often also referred to as vertices), that are connected by a

set of *edges* $\mathcal{E}$ (sometimes also called links). Thus, edges are elements in the set of the Cartesian product of the set of nodes with itself, meaning that $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Strictly speaking, this results in edges being ordered tuples, such that for two nodes $v, w \in \mathcal{V}$, we have $(v, w) \neq (w, v)$. Such a graph is called directed since the order of nodes in the tuple matters and there are two reciprocal edges that could connect any two given nodes (one in either direction). Since many applications do not consider a direction of edges, undirected graphs can also be considered, in which edges are formally treated as sets of nodes of size two, instead of ordered tuples. On occasion, it can be useful to represent a graph as an adjacency matrix $\mathcal{M}$ of dimension $|\mathcal{V}| \times |\mathcal{V}|$, whose binary entries $\mathcal{M}_{vw}$ indicate if there exists an edge between nodes v and w.

Both nodes and edges may be assigned additional attributes such as weights, which we model as functions $f : \mathcal{V} \to \textsc{Att}_f$ or $g : \mathcal{E} \to \textsc{Att}_g$ that map the respective nodes or edges to some corresponding attribute space Att. Typical attributes of graphs that we use in our model are the *neighbourhood* and the *degree*. The neighbourhood of a node v is the set of all nodes that are adjacent to v, meaning that there exists an edge that connects the node to v. We denote the set of neighbours with $N(v)$ and let, for an undirected graph

$$N(v) := \{w \in \mathcal{V} : (v, w) \in \mathcal{E}\}. \tag{2.4}$$

The degree $deg(v) := |N(v)|$ is then the number of neighbours of node v. For directed graphs, the neighbourhood and degree are defined analogously, but we discern between the indegree and outdegree of nodes, depending on the direction of edges.

While graphs provide the formal representation, and graph theory is concerned with the mathematics of graphs, the term network is typically used in applications that focus on the data or on processes that happen inside the graph. Network analysis is then concerned with modelling the topological structure and the principles of the emergence of graphs that represent observed complex systems. In the following chapters, we utilize only few additional graph notations or tools from network analysis, which we introduce as they are needed. For a more in-depth background, we refer to the textbook by Newman [144].

As we have already touched upon in Chapter 2.2, the application of many established network-analytic measures should be approached with caution, since the implicit networks in our model constitute a type of cooccurrence network. Therefore, we do not go into detail on the metrics that are frequently applied in network analysis to uncover the topology and global characteristics of such networks, but instead focus on leveraging the local structures for the discovery and retrieval of term relations. Thus, our approach also covers applications that typically fall into the domain of knowledge graphs.

2.3.2 Information networks and knowledge graphs

While almost any system with interconnected components can be modelled as a complex network, an important type of such networks are *information networks*. In contrast to plain (complex) networks, information networks encode discrete labels as attributes of nodes and edges [191]. Typical examples include scientific collaboration or citation networks that contain authors, publications, and journals or conferences as nodes, and (co-)authorship relations or editorial positions as relations, as well as cinematic networks of movies and actors. From an analytic point of view, such information networks are especially interesting when they have a heterogeneous structure in which nodes and/or edges have multiple different attribute values. For an overview of heterogeneous information networks, we refer to the recent review by Shi et al. [168].

Knowledge graphs

An important type of heterogeneous network for natural language processing are *knowledge graphs* (or knowledge bases), both as a product of language processing and as a resource. While the term *knowledge base* stems from the difference to *data base* with the intention of highlighting the fact that it stores knowledge and not just data, the knowledge bases that are in widespread use have a graph structure, so the term knowledge graph is equally appropriate. This structure is also reflected in the used RDF storage format that represents the contained information as triples [92], which can be interpreted as two nodes and a connecting edge. Based on this structure, knowledge graphs then support a variety of tasks in which the added knowledge is helpful, or allow the inference of information that is not explicitly contained [172].

There currently are a number of knowledge bases that started as scientific projects and serve as major sources of structured knowledge in scientific publications (and beyond), as well as some more focused knowledge bases with a more narrow focus (for a thorough overview, see [190]). Of the large knowledge bases, DBpedia [14] deserves mention as one of the major representatives of knowledge bases that aim to extract structured knowledge from the semi-structured and unstructured content of Wikipedia. Over the years, it has grown to include knowledge from over 100 language editions of Wikipedia. Of the more specialized knowledge bases, EventKG [76] is related to our work in its focus on entities as the central components of events. EventKG is a temporal event knowledge graph and is itself composed of aggregated event-centric knowledge from larger and more general knowledge bases, and is designed to provide event data for timeline generation or question

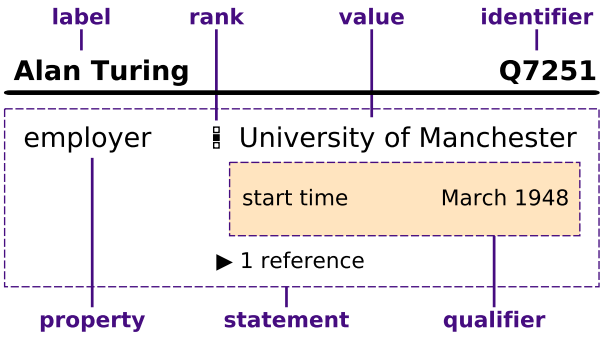

Figure 2.1: Example of an entry in Wikidata for the item that corresponds to Alan Turing. In contrast to traditional knowledge bases, Wikidata does not store facts but claims, which are backed by references and can be ranked by the community. The data model consists of only items and properties (classes do not exist explicitly), which entails that the entire class hierarchy can be edited by the users, causing it to gradually change according to the requirements of new data being added.

answering. In our work, we rely on two of the major knowledge bases, namely YAGO [124] and Wikidata [206], which we describe in more detail in the following.

YAGO. Similar to DBpedia, YAGO started as a project for the extraction of knowledge from the semi-structured content of Wikipedia, and its name is a tongue-in-cheek acronym for *Yet Another Great Ontology*. However, despite the apparent redundancy, YAGO includes a unique feature that makes it useful in the classification of entities: its taxonomy. Categories in YAGO are derived from a combination of classes that are extracted from Wikipedia categories with the WordNet ontology [133], which enables a much cleaner categorization of named entities than comparable knowledge bases. Several versions of YAGO have been published over the years, each with updated content and more included sources. Recently, the extraction of further contemporary versions of YAGO has been suspended in favour of making the extraction code available as open source code.

Wikidata. Unlike previous knowledge bases, Wikidata arose not from the need to extract knowledge from existing sources, but as a knowledge base behind Wikipedia to provide structured knowledge for handling the challenges of updating facts across the multilingual versions of Wikipedia. In contrast to previous approaches at constructing knowledge bases, Wikipedia is a collaborative knowledge base into which the users can enter statements manually, although it is increasingly expanded by automated bots [187]. Since its inception, Wikidata has been merged with the historically influential knowledge base Freebase [32] in a collaborative process that had users confirm imported statements [196], and has recently become a major source of structured knowledge well beyond Wikipedia. For our work, however, Wikidata is not interesting merely due to its size, but because of the

collaborative process behind it that ensures the addition of emerging entities in a timely fashion, which is relevant for the processing of news streams. Because of this collaborative process, however, Wikidata is using a hierarchy that is less rigid than those of extraction-based knowledge graphs, and features a more dynamic structure. As shown in Figure 2.1, statements can be assigned ranks and qualifiers to model their potentially limited validity periods. The perpetual change of the knowledge graph that this structure entails is both a blessing and a curse when Wikidata is used for linking entities, as we discuss in more detail in Chapter 2.4.

Knowledge extraction at a web scale

In the construction of knowledge graphs, semi-structured information such as tables or Wikipedia infoboxes are an invaluable source of information, but they are limited in scope. To include purely textual sources as well, the extraction of information from unstructured text is necessary. As a result, numerous approaches have been presented over the years towards such a procedure, which are too numerous and removed from our approach to cover here (for an overview, see [211]).

However, an especially important concept in this respect is *open information extraction*, which aims to identify novel entities and relations at a Web scale [16]. Conceptually, this can be achieved by repeatedly chaining the extraction of relations and entities. Newly identified relations are used to identify previously overlooked or emerging entities as one of their arguments, while entities in turn help to locate new relations that have the entities as arguments. As a result, this process is often pattern-based, although newer contributions also approach the problem by jointly embedding entities and relations [154]. In addition to providing knowledge for the extension of knowledge bases, open information extraction also serves to provide structured knowledge for *questions answering* (QA) from unstructured sources [109] as an information retrieval task.

A downside of these approaches is the rigid concept of relation, which must necessarily be qualifiable as a well-specified relation between two entities that is explicitly mentioned in the text. As a result, ambiguous relation information is often discarded, even though it might be sufficient to satisfy a user's information need in an exploration of the documents. This focus on qualifiable relations is the primary difference between knowledge graphs and implicit networks. Where knowledge graphs and relation extraction methods target explicit statements to extract discrete relations between entities from a local scope with a focus on precision, implicit networks are designed to extract quantifiable relation strengths between entities. Thus, the former process is more precise in the extraction

of relations from individual documents that can be described in an information network, whereas the latter emphasizes a higher recall to identify relevant relations across the document collection that are never explicitly stated in the documents.

2.4 Named Entity Annotation

With entities being a key concept in our document model, the recognition of entity mentions in documents is a central aspect of document preprocessing and annotation. For many applications, the size of the document collection can be expected to be too large to manually annotate the contained entities in the texts. Therefore, automated approaches are required, which we discuss in the following. This annotation of entities can be split into three steps, namely the recognition and classification of entities, the disambiguation of entities, and the linking of mentions to an entry in a knowledge base or gazetteer. While the latter two steps could be considered separate, they are typically performed jointly. For entity disambiguation and linking in particular, we also provide some details on using Wikipedia and YAGO as knowledge graphs.

2.4.1 Named entity recognition

The first question that one should address in regard to entities is *What is an entity?*, the answer to which is of course highly philosophical. In practice, entity annotation approaches have therefore typically focussed on named entities, which are somewhat easier to categorize, since names can be seen as rigid designators that uniquely identify the entity in question [110]. For example, the name of *Perth, Western Australia* is a unique identifier for the Australian city Perth. However, note that shortened forms of a name might be ambiguous (and therefore non-rigid), such as *Perth*, which could also refer to the city in Scotland. Such a rigid designator is the concept that enables the disambiguation of entities and the unique representation of entities with multiple names by one unique identifier. Historically, most emphasis has been put on named entities of the types person, location, and organization [79], although temporal expressions are often considered to be named entities as well. In the evaluation of our work in the following, we focus on these four types of entities, but note that this choice depends on the data sets that we use and not on the model itself, which is agnostic to the types of entities. In fact, the implicit network representation as introduced in Chapter 3 can be applied to any type of entity annotation that is appropriate in a given context, even if this definition of an entity were to be novel.

From a conceptual point of view, the recognition, annotation and classification of entities can then simply be described as the task of finding chunks in the text that represent the mention of an entity, determining the type of entity, and annotating it accordingly. In practice, however, this task is complex and the subject of ongoing research. For an overview and as a starting point, we recommend the survey by Nadeau and Sekine [139], which covers both the recognition of entities, as well as their classification. In our work, we use three tools in particular, namely the Stanford NLP toolkit, Ambiverse, and HeidelTime. However, since these tool also provide entity disambiguation and linking functionality, we discuss them only after we introduce these concepts.

2.4.2 Named entity disambiguation and linking

Once entity mentions in a document have been recognized (automatically), ambiguous mentions can be problematic. Consider, for example, a document that mentions *The President*, which could refer to numerous different individuals and is thus highly ambiguous. However, the context and metadata can often be helpful in resolving these ambiguities. For example, if the mention above occurs in a political document of the United States, it becomes more likely that it refers to the President of the United States. This, of course is still ambiguous since more than one person has held this office over the years, but metadata (such as the document date), or contained information (such as temporal expressions) may be used to resolve it. In our example, if the document can be dated to 1963, then the person in question could be John F. Kennedy or Lyndon B. Johnson. Even other ambiguous entity mentions in the text can be helpful in resolving the ambiguities. For example, if the text contains the phrase *The President and his wife Jacqueline*, it becomes very likely that the document is referring to the 35th president, John F. Kennedy.

In practice, the task of automated entity disambiguation can be modelled as the assignment of probability scores to possible target entities for each mention in the text, based on which a selection is performed. Typically, a list of candidate entities is generated from an ontology, knowledge base, or gazetteer for each mention. Therefore, once they are disambiguated, the entity mentions are also linked to an external resource, which motivated the term *entity linking* that is sometimes also referred to as *wikification* when Wikipedia is used as a target knowledge base. Approaches to solving this task are diverse and include, for example, the use of the context of mentions alongside knowledge bases to construct a coherence graph from which the best candidates for all mentions are determined jointly [97], or the computation of centrality scores on graphs that are derived from the similarity of word embeddings [233].

Of course, the phenomenon of ambiguity is not unique to entities, but affects all kinds of words, meaning that the task is a special case of word sense disambiguation [141], although there are some differences in practice [40]. Finally, the problem is not limited to explicit mentions of entities, but also includes the dereferencing of pronouns, and is thus linked to the task of anaphora resolution [91].

2.4.3 Entity annotation toolkits

For the evaluation and exploration of our entity-centric implicit networks and their applications, we require large collections of documents that are annotated for named entities. In the following, we therefore discuss the annotation tools that we later use.

HeidelTime. HeidelTime is a rule-based temporal tagger [188] that we use for the extraction of temporal expressions. In addition to the extraction of temporal expressions, HeidelTime also normalizes them to the TimeML standard [151], thereby providing entity disambiguation capabilities for temporal expressions. While the Stanford NLP toolkit also offers dates as one of the types of entities that it can extract, HeidelTime has the advantage of being domain sensitive [189], meaning that it can be adapted to the formatting in different text domains. In particular, it supports both the narrative domain (which includes Wikipedia texts) and the news domain (which we require for document stream analyses).

Stanford CoreNLP. The Stanford CoreNLP toolkit is a general-purpose natural language processing pipeline that includes entity recognition and classification functionality, but also provides sentence splitting, tokenization, and part-of-speech tagging [126]. Since it does not include entity linking, we use it in our experiments in applications where we focus on entities that are annotated and classified, but not linked.

Ambiverse. The Ambiverse natural language understanding suite is our primary tool for entity linking. It includes named entity recognition based on KnowNER [166], and provides named entity linking capability derived from AIDA [97]. Thus, it can be used as an end-to-end tool for annotating named entities in documents and linking them to Wikidata and YAGO identifiers. The last step is particularly important, since it is beneficial to rely on links to both knowledge bases, as we discuss in the following.

2.4.4 Linking and classifying entities with knowledge graphs

For the linking of entities, choosing the knowledge graph that is used for obtaining the entity names is not trivial. In addition to providing additional external knowledge about the linked entities (which differs from knowledge graph to knowledge graph), its class

hierarchy also serves as the primary source of information in the classification of entities. In this regard, both Wikidata and YAGO have some advantages and drawbacks.

The benefit of Wikidata, as mentioned in Chapter 2.3, is its inherently collaborative design, which enables almost immediate updates of the knowledge graph as new entities emerge and are being referenced in documents. While this is of prime relevance for the processing of document streams such as news articles, it also comes at the cost of a less stable class hierarchy. Since the Wikidata knowledge graph must be able to adapt to changing content to accommodate edits, its hierarchy also changes over time. Furthermore, the potential of vandalism [87] and the fact that errors are committed not systematically by one algorithm but randomly by a multitude of users, make the hierarchy of Wikidata less predictable than that of a traditional extracted knowledge base. As a result, using Wikidata class hierarchies even for the classification of the common entity types persons, locations, and organizations, is a difficult task (for a detailed discussion, see [176]).

In contrast, the class hierarchy of YAGO is partially derived from the WordNet ontology [132], which provides a slim hierarchy that is well suited for entity classification. This benefit, of course, comes at the cost of topicality, since YAGO is not updated incrementally but extracted in batch. As a result, keeping up to date with newly emerging entities is problematic when only YAGO is used.

To benefit from both approaches, we use a combination of the two knowledge bases in our work, which has the added advantage of potentially providing data from both knowledge bases to any linked entity. For the identification of entities, we use Wikidata identifiers. Since both Wikidata and YAGO are part of the Semantic Web, entities can in principle be matched through RDF relations. However, in our case, we found it simpler to use the Wikipedia page links that are contained in both Wikidata (due to the connection between the two projects) and YAGO (since it is extracted from Wikipedia content). We rely on this approach whenever we disambiguate entities for our evaluations and demonstrations in the following chapters.

2.5 Towards an Entity-centric Document Model

In this chapter, we have discussed numerous different approaches to the representation of documents, word and entity relations, and cooccurrences. However, when viewed through the lens of the requirements for entity-centric explorations of the documents and with a focus on entity relations within the context of their mentions, these approaches exhibit

one or several weaknesses. In the following, we summarize these issues and highlight what direction we are taking to propose a solution.

Vector space representations

In contrast to all other document models, the vector space representation has the advantage of an essentially unlimited window size. Any two terms that are contained in the same document are part of the same document vector. However, the model is focused on the occurrence of terms to model document content, but does not contain a fine-grained notion of *co*occurrence beyond document-level cooccurrences. As a result, while this model still provides a solid performance for document retrieval tasks despite its age, it is ill-suited for the representation and exploration of relations between terms inside the documents. By using a network approach in the following, we focus on the edges (that is, the relations between entities and terms), not on the nodes (that is, the terms). Thus, unlike the vector space model, weights in our network represent more flexible relation weights instead of term weights. In particular, relation weights can always be transformed to term weights if the localized scope around a finite set of entities is considered, while the reverse would be more difficult.

Word embeddings

While word embeddings are the de-facto standard model and current state-of-the-art for word representation and numerous retrieval applications, they suffer from three shortcomings. One is the limited size of the sliding window that limits the scope of observed cooccurrences. In many of the approaches, this window size is an artefact of the neural network architecture and cannot be increased without substantially increasing the training time of the models. The second problem concerns the issue of similarity and relatedness. Word embeddings are typically good at recovering word similarity due to the way in which they are trained, meaning that embeddings tend to be similar if the words they represent are (roughly speaking) somewhat interchangeable. However, they tend to perform worse when modelling the relatedness between words that occur in similar contexts but are not interchangeable. Therefore, despite the fact that embeddings are often used for such tasks, they are a poor choice when a measure of entity or term relatedness is required [8]. A third issue is the frequency of words, which embedding approaches typically address by limiting the vocabulary to the most frequent words. As a result, embeddings neglect potentially interesting parts of the document collection, which especially includes rare entities. While

such rare entities and their connection might be spurious, it is also well known that a substantial fraction of entities is in the long tail and thus rare [53, 194]. Therefore, disregarding them equates to disregarding a substantial amount of information in the document collection. However, precisely because these entities are so rare, even their global context is limited to few occurrence instances. Therefore, using a network representation of such entities with a very small neighbourhood size in the network may be more space efficient than relying on a relatively high-dimensional vector representation of the words. Thus, networks stand to offer a way of retaining the full relation information for rarer entities while requiring minimal training effort.

Collocations

Similar to embeddings, the analysis of collocations utilizes extremely narrow window sizes. However, in contrast to embeddings, this window size is narrow by design, since the focus lies on adjacent words that cooccur with significant frequency. For the exploration of entity relations, however, one finds that entities seldom occur in close proximity. Since entities are typically nouns, there is no syntactic basis for them to be adjacent (or even closely located in many cases). Thus, while the concept of considering cooccurrences for the sake of detecting word relations is similar to our proposed entity-centric network, the scope and scale of collocations do not match the task.

Relation extraction and knowledge graphs

The strength of knowledge graphs and relation extraction lies in identifying concrete relations between entities with high precision. However, due to this focus on precision, they are prone to low recall, and stand to miss relations that are a priori unknown or hard to specify in concise syntactic terms. Furthermore, while the edges of knowledge graphs are attributed, they are typically not weighted, which allows inference and reasoning, but complicates ranking and recommendation tasks. For the exploration of neighbourhoods in densely populated regions of the graph, they thus face difficulties. In contrast to embeddings, knowledge graphs are good sources of entity relation information, but less well suited to tasks that require a similarity of entity contexts, since much of the context information is lost during the extraction process. Thus, due to the focus on knowledge, the data that makes it into the knowledge base is a fraction of the content of the documents and too sparse to use as a model. However, knowledge bases can provide useful information not just for entity linking and disambiguation, but also as external knowledge in a suitable document model of entities, as we discuss in Chapter 7.

The way ahead

A final and central observation from our discussion of the related work is the lack of distance, as none of the presented approaches consider the distance of term and entity mentions in the text. However, proximity-based methods of representing cooccurrences have been shown to play a major role in the quality of the retrieved information [197]. Therefore, we conjecture that a model that includes not just cooccurrences but also cooccurrence distances is well situated to derive meaningful ranking scores from this information.

With regard to the representation of entity relations, we find similar room for improvement. On a spectrum of potential entity relations that ranges from continuous weights that are available for all pairs of words on the one side to discrete relations of only entities on the other, vector embeddings and knowledge graphs represent two extremes. While knowledge graphs contain only explicit relations between few entities, vector embeddings can be used to derive similarities between any two words in the document collection. In the following, we consider a middle ground in which edges in the network are continuously weighted but relations are not potentially omnipresent. Based on the implicit network model, we explore the context of entities and terms to distinguish those that *should* be semantically related, from those that *could* potentially be related.

Instead of extracting discrete relations between the terms and entities of a document collection, or learning vector embeddings that serve to later derive such relations, we investigate whether it might be more reasonable to represent the entire collection as a network in the first place.

3 Implicit Entity Networks

Based on the deliberations in the previous chapter and the clear lack of entity-centric document models, we introduce implicit entity networks in the following. To address the shortcomings of existing document models, we design the model to be both a conceptual representation of the data that supports a multitude of different retrieval tasks, as well as an efficient index that supports entity-centric retrieval operations on the underlying document collections. In doing so, we provide a basis for the more specialized applications that we introduce in the subsequent chapters.

Contributions. In this chapter, we thus make the following three contributions.

I We introduce and formalize entity-centric implicit networks as an efficient and versatile model of large document collections.

II We evaluate the performance of implicit networks for information retrieval tasks, and determine the scalability of such queries.

III We evaluate the performance of implicit networks against words embeddings as the current state-of-the-art in word representation.

References. Parts of this chapter are based on the following peer reviewed-publication and manuscript in preparation:

A. Spitz and M. 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 (SIGIR)*. 2016, pp. 503–512. DOI: 10.1145/2911451.2911529

G. Feher, A. Spitz and M. Gertz. "Evaluating Word Embeddings for the Prediction of Entity Participation in News Events". In preparation. 2019

3.1 Motivation

The textual description of phenomena in the real world, such as events, incidents, facts, or concepts, typically involves important entities as central components, which is reflected in mentions of (named) entities in the texts. Given a large collection of documents, however, such descriptions may be incomplete in a single context, or distributed across multiple documents. Consider, for example, the documentation of the Olympic Games and affiliated events in Wikipedia. For each iteration of the games, there are pages that go into great detail about the games themselves, along with a multitude of pages about individual athletes. However, not all information about everything that transpired at the games can be found on the central page, while the pages of individual athletes or sports events may be lacking important information, such as exact dates or even the location, and merely reference the Olympic Games of a given year. Thus, only the combination of data about multiple entities and from multiple contexts enables the full reconstruction of all pieces of a given iteration of the Olympic Games, including the place, the time, the involved participants, and their performances. As a result, numerous central tasks in information retrieval, such as event detection or summarization, are closely tied to named entity extraction and entity linking. For large document collections with a diverse set of entities, contexts, and concepts, a global approach that accounts for this distribution of mentions then also requires the incorporation of efficient indexing strategies across the documents.

To address entity-centric retrieval tasks in these cases, it can be beneficial to leverage partial information about the involved entities from multiple contexts to reconstitute the distributed relations. When we are investigating texts with a focus on the occurrences and cooccurrences of different types of entities across documents, we are therefore no longer bound by the concepts of *one sense per discourse* [63] or *one sense per collocation* [225], since we are not limited to one discourse or to one collocation. While these two interpretations have proven valid in many situations, there is room for improvement when large collections of documents are considered that contain partial and distributed relations. Since the task of entity disambiguation becomes less taxing when multiple entities of differing types are involved in a given context, merging the shared contexts of entity mentions stands to improve the retrieval performance. Thus, a model of the entire document collection that accounts for the context in which entities are mentioned appears sensible. This notion is further supported by recent research into web query evaluation, where the importance of a distinction between so-called *content words* and *intent words* has been highlighted [158]. By using a context-preserving representation of entity cooccurrences in the textual data,

it becomes possible to answer similarly structured queries, and leverage entity type information that would otherwise be lost.

Building on these observations, we introduce the implicit entity network model for representing and indexing entity cooccurrences across multiple documents within a document collection. The graph-based model then serves to represent the cooccurrences of (named) entities of arbitrary types in the context of the surrounding terms. While numerous systems and approaches have been proposed that enable the user to browse extracted information from textual sources based on some named entity dimension, such as timelines in the case of dates [9, 106, 165, 202], or a spatial dimension in the case of locations [2], there are no models that support arbitrary types of entities or enable a variety of downstream applications. However, these entity dimensions are often inextricably interwoven and retain their full semantic meaning only within their entire context, such as news event descriptions that involve, at the very least, locations and dates. Any method for the extraction and representation of entity cooccurrences should therefore be able to account for arbitrary types of entities with equal importance in the underlying model. To this end, we design the implicit network model to represent different types of entities equally, and include their cooccurring terms to preserve the context.

Structure. In Chapter 3.2, we formalize the concept of implicit entity networks and queries to this document model. We discuss implications for the application of the model in Chapter 3.3, and construct a network from the English Wikipedia that we use for an evaluation of the performance and scalability of queries in Chapter 3.4. Finally, in Chapter 3.5, we compare the performance of implicit networks to word embeddings and discuss the implications of the results.

3.2 The Implicit Network Model

Based on the importance of entities for the description of complex concepts or relations in almost any context, we focus on an entity-centric network representation of the documents. While the types of entities that are present in a document depend on the content and domain of the document, the underlying assumption of the importance of entity cooccurrences remains valid by generalizing it to the importance of word cooccurrences as discussed in Chapter 2. Regardless of the domain of the documents, if the most central concepts can be denoted as entities, then their cooccurrences are bound to indicate central relations. Since a graph is the most intuitive and natural formalization of a complex network of relations, we use this approach in the following.

3.2.1 A graph model for implicit networks

We begin by introducing the structural elements of the text that act as nodes of the graph, before introducing the edges and their weights.

Graph nodes

Let D be a collection of documents for which a network representation is to be extracted. Each such individual document $d \in D$ consists of a number of sentences $s \in d$. We denote the set of all sentences in all documents with S, that is, we let

$$S := \bigcup_{d \in D} d. \tag{3.1}$$

While two or more sentences may have identical content, note that we consider them to be separate sentences for the purpose of this model. That is, each sentence $s \in S$ occurs only once and in only one document. To account for this ordering of sentences inside a document, we define a mapping from sentences to their index.

Definition 3.1 (Sentence index ς). Let $\varsigma : S \to \mathbb{N}$ denote a mapping function from sentences to indices, such that each sentence s is mapped to its integer index $\varsigma(s)$ in the document that contains s.

Thus, ς provides an ordering of all sentences in a document in the order in which they occur when reading the text naturally. In the following, for the ease of notation, we assume that values of ς are consecutive and monotone increasing.

By making a bag-of-words assumption on the sentence level, we can treat each individual sentence as a set of words, which we then partition into entities and terms. For this purpose, recall that we do not make a distinction between words and multi-word expressions, as discussed in Chapter 2.1. Instead, we consider terms as chunks within a text that has already been tagged for entities. Specifically, we distinguish between two types of words, namely entities and terms. Entities E denote words or multi-word expressions that represent an entity mention in the text. For example, the expressions *Edgar Allan Poe* or even just *Poe* could refer to the American writer. What constitutes an entity may vary depending on the context and can range from persons, locations, or date mentions in an analysis of news articles, to organs, diseases, or drugs in medical documents. Conceptually, these entities can be regarded as a set of important terms that is given special weight in the context, which may be defined by the user when annotating the documents. With

regard to the identification of entities, note that entity mentions can either be only recognized and tagged as entities, or additionally be linked to some gazetteer or knowledge base. For the benefits and drawbacks of both approaches, see Chapter 2.4. In contrast, the set of terms T contains all remaining words that are not recognized as an entity. Formally,

$$T := \bigcup_{s \in S} \{w \in s \mid w \notin E\}. \tag{3.2}$$

A sentence can thus be considered to be a multiset of entities and terms $s \in (E \cup T)^*$. In the following, we define the graph representation of a document collection based on this intuition. Using these structural elements of the document collection, we can define the set of nodes for our network.

Definition 3.2 (Set of nodes $\mathcal{V}$). Given the four structural types documents D, sentences S, entities E, and terms T, we refer to their union $\mathcal{V} := T \cup E \cup D \cup S$ as the set of nodes of the implicit network.

To attribute the original type to each node, we use a mapping function that assigns nodes to the original subsets.

Definition 3.3 (Node type η). Let $\eta : \mathcal{V} \to \{T, E, D, S\}$ denote a function that maps each node $v \in \mathcal{V}$ to its original type $\eta(v)$.

Of course, we can also consider a more fine-grained version of such a function that maps to subsets within the set of terms or entities. For example, we could partition the set of terms by parts-of-speech, or entities according to their type, such as persons or locations. In the following, we use this notation where it is clear from context, without defining all possible subsets (since these largely depend on the application).

On a conceptual level, this model is agnostic to word ambiguities. Therefore, if a word has two meanings, we consider these meanings to be represented by two distinct elements of $\mathcal{V}$ that belong to different types. For example, consider mentions of *Washington*, which could be a person's name or a location, and should be mapped to two distinct nodes.

Edges and edge weights

To obtain a graph representation of the content of the documents, we also require a set of edges that connect the nodes in a way that reflects their relations in the document collection. To this end, we rely on cooccurrences within some window c that is measured in the number of sentences between the mentions of the entities or terms.

Definition 3.4 (Set of edges $\mathcal{E}$). Given a set of nodes $\mathcal{V}$ and some window size $c \in \mathbb{N}$, let $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denote the set of edges. For each $(v, w) \in \mathcal{E}$, we require that one of the following conditions is met. **(i)** We have $v \in S$, $w \in D$, and $v \in d$. **(ii)** We have $v \in E \cup T$, $w \in S$, and $v \in w$. **(iii)** We have $v, w \in E \cup T$ and there is at least one document in which v and w occur at most c sentences apart.

Due to the relatively free word order that is inherent to most languages (especially if sentences with multiple clauses are considered), modelling the cooccurrences of entities or terms as a directed graph that is based on word order does not seem reasonable, as discussed in Chapter 2.1. In the following, we thus focus on an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that represents the cooccurrences of the terms, entities, sentences, and documents as introduced above. Since the set of edges $\mathcal{E}$ of $\mathcal{G}$ is undirected, we do not distinguish between the edges (v, w) and (w, v) that connect two nodes v and w, and instead rely on the types of incident nodes to characterize edges as described in the following.

To generate weights for the edges of the graph that are derived from mention distances, we rely on the concept of *instances* I of entities or terms.

Definition 3.5 (Occurrence instance ι). Let an instance describe a distinct occurrence of an entity or term in a sentence (and thus in a document). Then, there exist surjective mappings $\iota_e : I \to E \cup T$ from instances to entities or terms, and $\iota_s : I \to S$ from instances to sentences that map each instance to the corresponding node in the graph.

For example, consider some entity v that occurs twice in a given document d. Then there exist instances i and j along with sentences $s_i, s_j \in d$ such that $\iota_s(i) = s_i$ and $\iota_s(j) = s_j$, as well as $\iota_e(i) = \iota_e(j) = v$. Let I_v denote the set of all instances of node v of type entity or term. That is, let $I_v := \{i \in I \mid \iota_e(i) = v\}$.

For the ease of notation in the following chapters, we sometimes also use the simplified notion of cooccurrence instances instead of occurrence instances. We denote with I_{vw} the set of instances in which terms or entities v and w occur jointly in a document. Then, we can also simplify the sentence based distance δ to directly write $\delta(v, w, i)$ for the distance between two nodes v and w in a given cooccurrence instance i. We use this notation where it is sufficient or expedient to consider cooccurrence instances.

To generate edge weights $\omega : \mathcal{E} \to \mathbb{R}$ from these instances and their cooccurrences, we then distinguish between two different types of relationships. The first type of relationship occurs between sentences and documents with entities and terms, and describes a sort of containment. That is, it describes how an entity or term is included in a sentence or

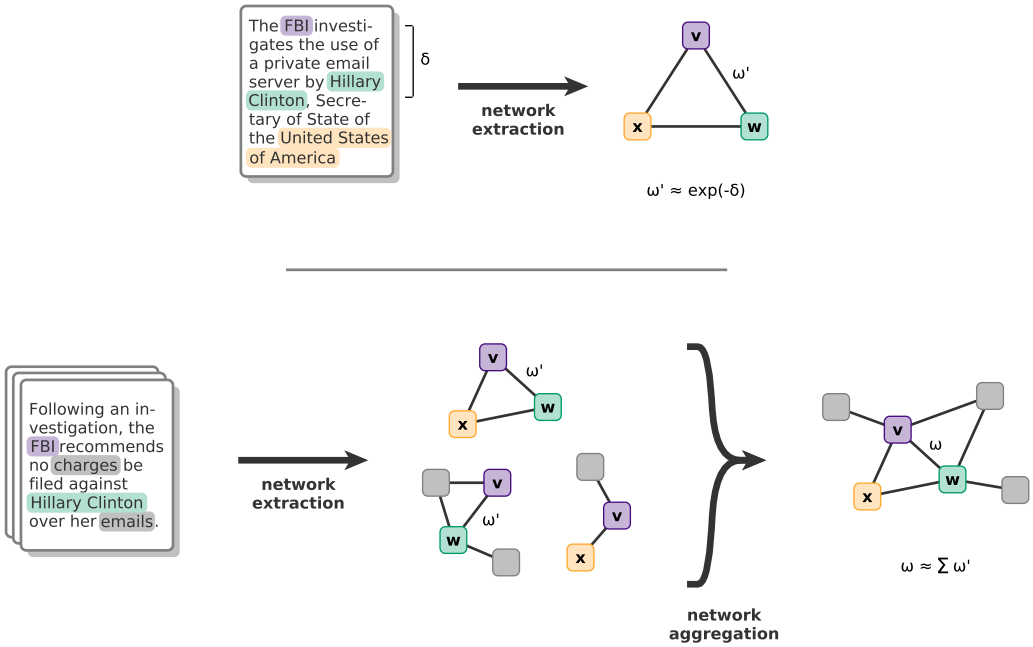

Figure 3.1: Schematic view of the implicit network construction process for a document collection. Top: within each individual document, the cooccurrence distances δ between mentions of entities and terms are measured. Entities and terms then constitute the nodes of a graph (along with sentences and documents) and are connected by edges, whose weights are generated from the distances. Bottom: the graphs that are generated from individual documents are aggregated by merging the weights of all edges. The resulting large graph is the implicit network representation of the entire document collection.

document. For this inclusion of entities or terms v in sentences s, we use the number of occurrences as a weight. That is, we set

$$\omega(v, s) := |\{i \in I \mid \iota_e(i) = v \wedge \iota_s(i) = s\}|. \tag{3.3}$$

We define the weights between sentences and documents analogously. However, note that this results in binary relations in which the weights equal 1 if and only if the document contains the sentence, and 0 if it does not. While it is possible to define a similar relationship between terms and documents or entities and documents, this can be omitted in practice since there exists a surjection from S to D, which allows us to reconstruct these relations as necessary without explicitly generating them.

The second type of relationship occurs between multiple entities or terms, and describes a cooccurrence within sentences and documents. To derive edge weights for this case, we first rely on the mention distance δ between two instances.

Definition 3.6 (Mention distance δ). Let i and j be two occurrence instances of entities or terms. Then their mention distance is defined as

$$\delta(i,j) := |\varsigma(\iota_s(i)) - \varsigma(\iota_s(j))|. \tag{3.4}$$

Thus, the distance between the two mentions equals the number of sentences between the instances, or 0 if they occur in the same sentence. If i and j do not occur in the same document, we set $\delta(i,j) := \infty$. Based on this distance, we can define the weight of edges between two entities or terms.

Definition 3.7 (Undirected edge importance ω). Let $v, w \in E \cup T$, then the importance weight of an edge $e = (v,w)$ between them is defined as

$$\omega(v,w) := \sum_{\substack{i \in I_v \\ j \in I_w}} \exp(-\delta(i,j)). \tag{3.5}$$

That is, we assign a weight to each cooccurrence of instances of v and w that diminishes exponentially with the distance between the two instances. This allows us to capture cross-sentence mentions of entities or terms that might otherwise only be captured by dereferencing pronouns. We then derive the total edge weight as the sum of weights of all individual instances. For a schematic view of the network extraction and construction process, see Figure 3.1.

Adjacency matrix

To obtain a matrix representation of the network that is based on the above weights, we can treat edges with a weight of 0 as non-existing to obtain a sparser and more efficient representation in subsequent analyses. Thus, the binary adjacency matrix $\mathcal{M}$ of graph $\mathcal{G}$ with dimension $|\mathcal{V}|^2$ can be defined as

$$\mathcal{M}_{vw} := \begin{cases} 1 & \text{if } \omega(v,w) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Recall that we refer to the number of edges that are incident on a node v as the degree of v and write $deg(v)$. Analogously, we define the type-specific degree $deg_x(v)$ of a node v as the number of edges from v to other nodes w for which $\eta(w) = x$. For example, $deg_S(v)$ denotes the number of sentences that contain a term or entity v.

3.2.2 Including hierarchical completeness conditions

For some types of entities, it may be reasonable to include further relationships, such as ontological information that is stored in an external knowledge base. We give a detailed description of an extended graph model that is designed to include such external hierarchies in Chapter 7. However, in some cases it may also be sensible to consider these relations during the construction of the network and include them directly in the graph representation, especially if entity mentions cannot be linked to a knowledge base. In particular when dealing with temporal expressions, an important type of hierarchy that directly influences subsequent exploration and retrieval tasks on the generated networks, is given by the granularity of dates, which includes years, months or days. To account for these differences in granularity, we can consider dates to be sets of time points for which an inclusion hierarchy exists. That is, each day is included in a month and the same relation holds between months and years. For example, if we are looking at the historic event of the declaration of independence of the United States on July 4, 1776, then it is easy to see that this event also happened in July of 1776, as well as in the year 1776. As a result, if dates $\textsc{Dat} \subset E$ are a subset of the set of entities, it is sensible to partition $\textsc{Dat}$ into three subsets of dates with precisions day, month, and year, such that

$$\textsc{Dat} = \textsc{Dat}_{day} \cup \textsc{Dat}_{month} \cup \textsc{Dat}_{year} \tag{3.7}$$

In a different domain, such as event descriptions from news articles that are published over a brief period of time, it may also be reasonable to consider hours as a fourth granularity level. In any case, a temporal hierarchy is integral to temporal retrieval operations on the network, especially if uncertain information is considered, and should be reflected in the graph representation. Thus, we may require that for all edges $(v_\tau, v) \in \mathcal{E}$ that include some date mention v_τ, if there exists a more coarse-grained date mention v'_τ such that v_τ is included in the time span covered by v'_τ, then we should also have $v'_\tau \in E$ and $(v'_\tau, v) \in \mathcal{E}$. Considering the example given above, if a mention of the U.S. cooccurs with the date July 4, 1776, then there should also be an edge between the U.S. and July 1776 in the network. Furthermore, we have to not only adjust the edge weight of the originally induced edge, but also the weight of the edge to the more coarse grained date. Thus, an edge between a day and another entity necessarily induces an edge between this entity and both the month and the year that include the day. An analogous, albeit higher-dimensional, hierarchical completeness condition can be formulated for locations if a geographic or spatial hierarchy of places is known. Similarly, persons could be structured in hierarchical organizations,

such as ambassadors and the countries that they represent. For a thorough discussion of network-centric relation refinement, we refer to our investigation into this issue [179].

A second kind of hierarchy can be defined for the names of persons, and is of particular interest when the mentions of these names are not normalized or linked to a knowledge base, but only recognized as string representations of person names. Here, we observe that persons are frequently referred to by only parts of their full name or variations thereof, such as the first or last name. As an example, Alan Turing may be referred to as Alan Turing, Turing, etc. However, all mentions refer to the same person and induce cooccurrence relationships. To better support the exploration of such networks, we may employ an approach that is reciprocal to the inclusion condition for dates as described above, and split a name n into components $n_1, ..., n_k$. We then require that if we extract an edge $(v_n, w) \in \mathcal{E}$ between the full name v_n and some other entity or term w, then we also include edges $(v_{n_1}, w), ..., (v_{n_k}, w)$ and adjust the edge weights accordingly.

In principle, while it is possible to apply this scheme to the names of organizations, these generally do not follow the same naming conventions as persons. Where a person might be referred to by their first and last name alone in different contexts, an organization will only rarely be referred to by parts of its name. However, an application to the names of locations is possible in cases where geographic hierarchy information is missing.

Of course, these methods for including the notion or hierarchy during the extraction process are heuristics. Therefore, they should be used with caution and in accordance with the entities that are contained in the data. It is obvious that these heuristics are likely to work for some cases, but perform poorly in others.

3.2.3 Node ranking functions

Based on the graph representation of the implicit network, we can now focus on graph nodes as the central components of retrieval tasks. Since nodes represent both content (entities and terms) as well as structure (sentences and documents), it is possible to formalize different retrieval operations as localized queries to the neighbourhood of nodes. To leverage the graph representation and extract information from the cooccurrences of entities in the document collection, we thus employ local ranking functions of node neighbourhoods. For any given input query entity, we can rank adjacent nodes in the graph representation by the importance of their cooccurrences to obtain the most relevant related entities. In other words, we obtain a ranking of nodes, depending on the strength of their connections to other nodes. The benefit of this approach is the possibility of computing

relevant neighbours efficiently due to the graph representation, despite the overall size of the document collection. Global rankings of relations are also possible, and we consider these for the extraction of network topics in Chapter 6.

We first consider a bivariate ranking function between two sets of nodes, where the sets correspond to either the type of node (term, entity, sentence, or document, for example), or the type of entity (such as persons, or locations). Thus, let X and Y be two sets of nodes. We can then define a bivariate relevance function as $\varrho_{XY} : X \to \mathbb{R}^{|Y|}$, that is, we map a node $v \in X$ to a vector of relevance scores such that each node in Y is assigned a relevance score with regard to v. Thus, we determine the most relevant neighbours of a query entity v by ranking adjacent nodes according to their relevance.

To instantiate the ranking function, we want to find a way of ranking relevant nodes efficiently on a local level, by using only the neighbourhood of that node. Therefore, we employ a graph adaptation of the *tf-idf* score [157], which we further modify to include the distance-based weight between entities and terms instead of simple cooccurrence counts. As a measure of the relevance of cooccurrences between two specific nodes, we build upon the analogy to terms that are contained in a document in the vector space model, by equating nodes $y \in Y$ with documents that "contain" the nodes $x \in X$ to which they are connected. For entities, this covers the edges to other entities or terms with which they cooccur. For documents or sentences, it equates to the terms and entities that they contain. Thus, we formalize both cooccurrence and containment relations in one unified representation. We then observe that the importance weight $\omega(x, y)$ in the graph corresponds to the overall strength of cooccurrences between x and y in the document collection, meaning that it represents a kind of term frequency (*tf*). Similarly, the class-specific degree of a node (that is, the number of adjacent edges that connect to nodes of the respective class) is equivalent to the frequency with which it appears alongside nodes of that particular class. As such, it is a kind of document frequency and can be used to compute an inverse document frequency (*idf*). By combining the two, we arrive at a version of the *tf-idf* score that is adapted to the graph representation. Since this process introduces edges weights that are asymmetric, we refer to this weight as the directed importance $\vec{\omega}$.

Definition 3.8 (Directed edge importance $\vec{\omega}$). Let $x, y \in E \cup T$ be two nodes, and let $\eta(y) = Y$ be the type of node y (that is, a subset of entities or terms containing node y). Then we define the directed edge weight between x and y as

$$\vec{\omega}(x, y) := \frac{1}{f}\omega(x, y) \log \frac{|Y|}{deg_Y(x)}, \tag{3.8}$$

where f is factor that can be selected to normalize the resulting weight.

To obtain a normalized ranking in the interval $[0, 1]$, we can use the normalization factor f to divide the resulting scores by the maximum local score:

$$f := \max_{\hat{y} \in Y}(\vec{\omega}(x, \hat{y})) \tag{3.9}$$

Thus, we effectively normalize the importance weights ω and induce an edge direction, since $\vec{\omega}(x, y)$ does not necessarily equal $\vec{\omega}(y, x)$. By ranking all nodes that are connected to a query node according to their normalized $\vec{\omega}$ scores, we obtain the desired relevance ranking ϱ. Since there is no difference between the partitions of the graph from a graph-theoretic point of view, we may select the classes as X and Y in any combination, and can even let $X = Y$. In practice, selecting entity types as classes is a sensible approach in most applications that have distinct types of entities.

We are, however, not limited to bivariate ranking functions, which are only feasible for single input query nodes and may thus be too limited in practical applications. For example, in the case of events that have a geographic and temporal aspect and pertain to actors and locations (or both), it may be necessary to obtain the relevance of nodes in one set with regard to nodes from multiple other sets. For example, we may want to rank dates in relation to both a location and a person to establish a visitation date. If we consider the inclusion of descriptive terms for entity relations, the importance of relevance functions that accept sets of input nodes is emphasized further. Thus, we need a relevance function that is of higher arity. That is, we require a function $\varrho : X^n \rightarrow \mathbb{R}^{|Y|}$, where each X_i denotes an input set. To instantiate such a function, we use a combination of the individual functions for each individual input node as defined in Equation 3.8.

Definition 3.9 (Edge scoring function ϱ). Let $x_1, ..., x_n \in X^n$ be a vector of input entities. Then we define the score of a potential output y in relation to all x_i as

$$\varrho(x_1, ..., x_n, y) := \frac{1}{n} coh(x_1, ...x_n, y) \sum_{i=1}^{n} \vec{\omega}(x_i, y), \tag{3.10}$$

where *coh* can be used as an optional parameter to enforce local cohesion.

For our experiments in the following, the function *coh* served to ensures that only those candidate entities count towards the relevance score that are connected to at least two query entities. Formally,

$$coh(x_1, ...x_n, y) := \begin{cases} 1 & \text{if } \sum_{i=1}^{n} \sum_{j>i}^{n} \mathcal{M}_{yx_i} \mathcal{M}_{yx_j} > 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

As an alternative to this binary interpretation of cohesion as a way of handling multi-node queries, a multi-component ranking scheme that is using an integer-based cohesion is conceivable, which we consider as a possible extension in Chapter 4.2. Conceptually, the model could also be parametrized to include different weights for different types of nodes. However, since it is not clear how values for such weights would be obtained in practice and they increase the complexity of the model, we do not pursue this further.

3.2.4 Document and sentence ranking

The ranking of sentence and document nodes in the graph enables (extractive) summarization and document retrieval, respectively. For summarization tasks, it would be possible to extract a combination of adjacent entities and terms to generate descriptions of some set of input entities. However, it may be more convenient to directly extract entire sentences that are relevant to the provided entities instead of just keywords. Similarly, if we consider entity linking as a task, it may be convenient to retrieve descriptive documents for a combination of entities or entity relations. Thus, the ranking of sentences and documents is also of interest and can be easily achieved based on the graph representation. Formally, this requires a function $\varrho : X^n \rightarrow \mathbb{R}^{|Y|}$, where the $X_i \in \{E, T\}$ and $Y \in \{S, D\}$. In the following, we employ a straightforward instantiation of this relevance function as a proof-of-concept by counting the number of nodes in the query set that are connected to a sentence (that is, the entities that are contained in the sentence) or to all sentences within a document, respectively. For sentences s, we thus arrive at the following instantiation of a relevance score:

$$\varrho(x_1, \ldots x_n, s) := \sum_{i=1}^{n} \mathcal{M}_{sx_i}. \tag{3.12}$$

For documents d, we can use the fact that the edges between sentences and documents are 0-1-valued to count the occurrences of all query terms in all sentences of a document and obtain

$$\varrho(x_1, \ldots x_n, d) := \sum_{s \in S} \sum_{i=1}^{n} \mathcal{M}_{sx_i} \mathcal{M}_{sd}. \tag{3.13}$$

We use rankings of sentences and documents that are based on these relevance scores in our examples of network exploration applications in Chapter 3.4. However, more in-depth relevance functions and ranking schemes for sentences are possible, as we discuss in Chapter 4.3, where we also evaluate them for the extraction of entity descriptions and the extractive summarization of entity relations.

3.2.5 Asymptotic complexity of queries

For large document collections, a key consideration and the main benefit of a localized graph model is the running time of queries. The scoring functions as defined in Equations 3.8 and 3.10 can be computed efficiently in $O(\langle deg_X Y \rangle \langle deg_Y X \rangle)$, where $\langle deg_X Y \rangle$ is the average set-specific degree of nodes in set X with respect to set Y. For sparse graphs, these average degrees are smaller than the number of nodes by orders of magnitude, thereby rendering these measures highly efficient for the analysis of large document collections. Given the size of graphs that are obtained for larger document collections, this is a critical feature as we show in the event-centric evaluation in Chapter 3.4, where these scoring functions allow for an ad-hoc ranking of neighbouring nodes. Beyond the presented scoring functions, the graph representation can serve as an index for further entity-centric approaches that we discuss throughout the subsequent chapters, where it benefits from a similar complexity due to the reliance on the localized network structure.

3.3 Implementation and Practical Considerations

In the following, we describe the technical details that are relevant to the implementation of the implicit network model as described above. We first introduce the algorithmic basis for implementing the network extraction, before highlighting the important steps in preprocessing the documents and discussing the benefits and drawbacks of entity annotation with and without entity linking.

3.3.1 Implicit network extraction algorithm

Based on the model presented in the previous section, the implementation of an implicit network extraction algorithm is fairly straightforward by extracting named entities from sentences, which are then connected in a graph structure with weights generated from their distances in the text. When implementing this model, it is important to note that fully connecting all entities and terms in a document produces an extremely dense graph in which each document is represented by a clique, and the combinatorial explosion in the number of edges results in a graph that would be prohibitively large for large document collections. However, the majority of these edges have little impact in practice. While the theory of the model is based on the assumption that entities in the same document share some connection regardless of their distance in the text, long-distance connections are weak and mostly negligible due to the exponential decay in edge weights. Therefore,

we include a cut-off parameter c in the algorithm that excludes edges if the distance between the two instances is too large. Similarly, while it is sensible to model cross-sentence relations between entities, terms are less likely to be related to entities outside of their containing sentence, so we limit the edges between terms and entities to those that occur in the same sentence. Based on these considerations, we arrive at the algorithm for constructing an implicit network representation of a document collection (see Algorithm 1). Conceptually, multiple ways of generating such a network are conceivable. In practice,

Algorithm 1 Creation of an implicit network based on the output of named entity annotations of a document collection. For the definition of the distance function δ, see Chapter 3.2. The *update* function adds a new edge if it is not yet contained in $\mathcal{E}$, or adds the value to the existing weight ω if it already is contained in $\mathcal{E}$. The cut-off parameter imposes an upper bound on the distance between cooccurrences.

Input: Documents D, cut-off parameter c

initialize $\mathcal{V} \leftarrow \emptyset$
initialize $\mathcal{E} \leftarrow \emptyset$
initialize $\omega(v, w) \leftarrow 0$ for all v, w
for $d \in D$ **do** ▷ iterate over all documents
    $S_d \leftarrow$ sentences in d
    $E_d \leftarrow$ entities in d
    $\mathcal{V} \leftarrow \mathcal{V} \cup S_d \cup E_d \cup \{d\}$ ▷ add sentence, document and entity nodes
    for $s \in S_d$ **do** ▷ iterate over all sentences in the document
        update $\mathcal{E}$ with $(s, d, 1)$ ▷ link the sentence to the document
        $E_s \leftarrow s \cap E_d$ ▷ get all entities in the current sentence
        for $e \in E_s$ **do** ▷ iterate over all entities in the sentence
            update $\mathcal{E}$ with $(e, s, 1)$ ▷ link the entity to the sentence
        $T_s \leftarrow s \setminus E_s$ ▷ get all terms in the current sentence
        $\mathcal{V} \leftarrow \mathcal{V} \cup T_s$ ▷ add term nodes
        for $t \in T_s$ **do** ▷ iterate over all terms in the sentence
            update $\mathcal{E}$ with $(t, s, 1)$ ▷ link the term to the sentence
            for $e \in E_s$ **do** ▷ iterate over all entities in the sentence
                update $\mathcal{E}$ with $(e, t, 1)$ ▷ link the term to the entity
    for $e_1 \in E_d$ **do** ▷ iterate over all entities in the document
        $E_d \leftarrow E_d \setminus \{e_1\}$
        for $e_2 \in E_d$ **do** ▷ iterate over all other entities in the document
            if $\delta(e_1, e_2) \leq c$ **then** ▷ if the distance is within the window
                $w \leftarrow \exp(-\delta(e_1, e_2))$ ▷ compute the weight contribution
                update $\mathcal{E}$ with (e_1, e_2, w) ▷ update the edge weight

Output: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \omega)$

iterating over all sentences in a document in sequence and extracting the local subgraph that is induced by the context window around this sentence is most sensible.

Asymptotic complexity of the network extraction

The complexity of this extraction procedure is quadratic in the size of the context window that is induced by the cut-off parameter c, but linear in the number of sentences and documents. The same holds for the number of generated edges in the local subgraphs. It is easy to see that the extraction process is trivially parallelizable by documents, which results in one graph representation of each document that are then aggregated into one combined graph structure. Aggregation of the individual document graphs can be achieved by sorting the generated edges, which adds a logarithmic factor to the time complexity. Overall, the asymptotic runtime requirements are then in $O(c^2|S|\log(c^2|S|))$, where c is typically a fixed and small parameter such that c^2 can be regarded as a constant, resulting in a loglinear runtime of $O(|S|\log|S|)$ in practice.

3.3.2 Document processing

A number of document-level preprocessing steps are necessary before and during the extraction of the network. The key factor in these decisions that is of practical importance to the implementation concerns the identical handling of the preprocessing steps during the generation of the network and during subsequent queries in order to ensure compatibility.

Character encoding

Unless the document collection that is used as input is very meticulously cleaned, special characters need to be removed. Typically, a thorough cleaning during preprocessing is not possible for large collections, since it is impossible to decide which characters are part of names, and which are artefacts (consider, as a simple example, title abbreviations that may or may not include periods, or the band name *U2*). Furthermore, accents that are present in the input texts may be difficult to input at query time (for example, due to querying the network from a mobile device with limited input capabilities, or from a laptop keyboard with a different language layout). Since we are extracting networks from English texts in the following, this is mostly an issue for foreign names, but since the approach is language agnostic, the issue needs to be addressed with the application scenario of the network in mind, especially for languages with more exotic characters or accents. For our experiments

in the following, we use UTF formatting and typically keep accents in the data, but use a lower case representation of all characters.

Term extraction

The extraction of terms is not as straightforward as it initially appears and requires a decision on what constitutes a word. Since terms are defined as the words of a sentence that are left after entities have been removed from the sentence, the removal of entities is key and can be performed in two ways, namely by part-of-speech tagging or by entity deletion. In an ideal scenario, part-of-speech tagging can be used to identify terms since the documents are tagged for parts-of-speech in preparation of the entity extraction step. Using these annotations, it is then possible to identify non-entity terms for the extraction or even filter them by type. However, this approach can be problematic in practice due to errors that occur during part-of-speech tagging, or due to some character sequences that might represent interesting terms not being recognized as words. Furthermore, while entities should not overlap in an ideal scenario, which would enable a structured removal of entities from a sentence, overlaps happen in practice, especially when several different annotation tools are used. Thus, a simpler solution to the extraction of terms is a bitmap for all characters of a sentence that is used to mark the covered text of entities for deletion. Afterwards, marked parts of the sentence are removed and the remainder is considered for term generation. While this prevents the overlap, it also stands to introduce word fragments or non-words as terms. Thus, setting filtering criteria and minimum word lengths for a word to be considered a term may be appropriate. Furthermore, words can be excluded from the list of terms based on frequency, such as frequent stop words, misspelled words, or word fragments that are extremely infrequent.

Stemming of terms

To reduce the size of the graph and to group related terms into a single node for improved recall in query answering, we recommend the use of a stemmer for term processing. This makes it much easier to match semantically similar words in a query on the resulting graph. Generally speaking, lemmatization would likely be preferable to stemming due to its better performance. However, lemmatization of terms only works properly during the document processing phase in which entire sentences are available, due to the need for part-of-speech tags that we discussed in Chapter 2.1 . In most application settings that intend the graph structure to be queried with individual query terms that are not contained

in a proper sentence structure, lemmatization would not work well due to missing context and syntactic information. Thus, it would not be possible to properly match query terms to lemmatized nodes in the graph, thereby making queries incompatible with the network.

Entity annotation and linking

In addition to the basic text preprocessing steps, a final point with strong practical relevance that is worth considering is the dependence of the implicit network model on the annotation of entities in the documents, which deserves attention due to the entity-centric focus of the model. In particular, the user is faced with a choice between only recognizing entities in the text, or also linking them to a gazetteer or knowledge base. On the surface, entity linking is the superior choice since it maps mentions of the same entity to the same node in the graph, even if there are differences in the surface forms of the mentions. Thus, entity linking takes care of disambiguating different or partial spellings of identical entities, which improves the performance of subsequent retrieval tasks due to a more complete representation of entity contexts. In practice, however, entity recognition requires an extensive NLP stack that includes, at the very least, sentence splitting, tokenization, chunking, and entity recognition itself. In cases where the data is annotated manually, this may not be problematic, but in cases of automatic annotation (which is the only viable option for large document collections), these automated steps are bound to incur a cumulative and propagating error that likely potentiates throughout the annotation pipeline. Thus, adding entity linking to the pipeline as a final step is likely to increase the overall error. In practice, this typically leads to problems in the recognition of infrequent or emerging entities that are mentioned very sparsely, while it has a more negligible effect on more frequent entities. Thus, the problem can be countered by calibrating the recognition and linking methods to optimize precision at the cost of recall, thereby emphasizing more common entities in the implicit network. In the following, we use both approaches, but primarily include entity linking for tasks that benefit from uniquely identifiable entities, such as politicians or places in the analysis of news articles in Chapters 5 and 6.

3.4 Event Completion on Wikipedia Data

To derive an evaluation task that fits the capabilities of implicit networks as entity-centric document models, we focus on the description of events as one of the core concepts of human communication. Since humans tend to not only speak about the way things are, but also about how they change, it is not surprising that the detection, extraction, and

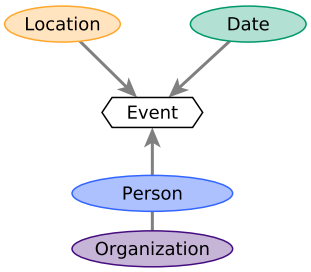

Figure 3.2: Conceptual visualization of an event as something that happes at a given time, to one or several actors, and at a given place, which highlights the importance of entities for the description of events.

analysis of events plays a pivotal role in natural language processing. As a result, a lot of research has already been devoted to events in general, with previous work focussing on the extraction of events and temporal facts from more or less unstructured textual sources, such as news articles [104, 112], social media [67], Twitter [9, 155], or Wikipedia articles [60, 113, 209] and even Wikipedia edits [71, 105]. Some authors have taken the opposite approach and linked Wikipedia events to historic news articles [137]. The breadth of these applications is reflected in the number of different definitions for the term *event* in the literature, so it makes sense to consider the precise definition of event that we use for our evaluation in the following.

For our evaluation, we view events in analogy to the definition given for the topic detection and tracking task [46] as *"something that happens at a given place and time between a group of actors."* We regard events as something at the intersection of *time*, *location* and involved *actors*, either individually or as *organizations* of multiple persons. Thus, we find that there is a strong emphasis on entities as components of events, as highlighted in Figure 3.2. In these cases, the void of structural information in unstructured textual sources is one of the greatest challenges, which can be ameliorated by focussing on the structure of the events themselves, and relating them to structures in the implicit network. On a language level, this concept is reflected in the original ACE definition of event detection [50], and has been utilized for tasks such as event threading [140] and incident threading [59]. On a higher level and across documents, however, with prior knowledge of already extracted named entities that are not necessarily contained within the sentences that correspond to the traditional definition of events, a local approach alone is not sufficient. Thus, it is reasonable to consider the representation of entity cooccurrences in an implicit network that is aggregated from individual cooccurrences across multiple documents, and use them to detect and complete event mentions.

3.4.1 An implicit network of Wikipedia

As a basis for our evaluation of events as an entity-centric extraction task that can be backed by an implicit network representation of the data, we extract such a network from Wikipedia. The benefits are comprehensiveness, since few text sources cover as broad a spectrum as Wikipedia, as well as scale, due to the size of the document collection.

Entity types for events: LOAD

With the main task being the identification and representation of events that emerge from the joint occurrence of named entities and temporal expressions, we base our model on the relations between distinct classes of such entities. The most prominent class of involved entities are the *actors* in an event. Generally, these correspond to the named entity type of *persons*, but we use the more general term to also include non-person actors in possible fictional settings, to which the model is equally applicable. The underlying assumption is that actors are singular individuals. In contrast, we consider groups of people to be *organizations*, meaning that in this model, an organization may describe a company, a political party or even a rock band (however disorganized the latter two might seem, at times). To represent the geographic component, we include *locations*, which in the most general interpretation are points or areas in space at which events take place. Finally, the temporal dimension is included by the mention of *dates* as temporal expressions. Here, we only consider dates of the granularity levels day, month, and year, but do not include time intervals. However, a refinement of the model that includes sets of temporally ordered, discrete dates as intervals is certainly possible. Since we are interested in modelling and extracting the relationship between these entities, we can use an implicit network that contains the four mentioned types of entities as a basis for information retrieval and event exploration tasks. Due to the structure of events as composites of entities with different types, we only extract edges between entities of different types. As a result, we obtain an implicit network that contains entities of type locations Loc, organizations Org, actors Act, and dates Dat, to which we refer as the *LOAD*-network. Thus, the set of entities can be regarded as a union of four sets $E = \text{Loc} \cup \text{Org} \cup \text{Act} \cup \text{Dat}$. A schematic visualization is shown in Figure 3.3.

Data selection and annotation

To test the model on a large-scale document collection, we use the entire dump of the English Wikipedia from June 2, 2015, as input. Since implicit networks are designed to

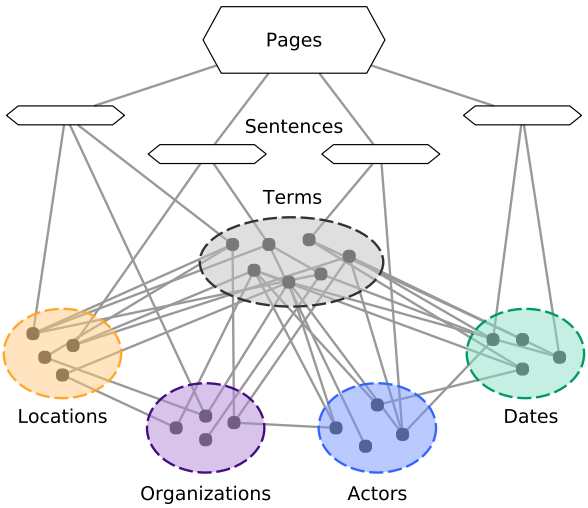

Figure 3.3: Schematic view of the implicit network extracted from Wikipedia articles (pages) for locations, organization, actors (persons), and dates.

represent the information that is contained in unstructured text, we exclude infoboxes, tables, references, and pages of lists, and therefore only use the raw text without any links or pre-existing annotations. For the tokenization, sentence-splitting, POS-tagging, lemmatization and named entity recognition, we use the Stanford Named Entity Recognizer [61]. We employ the 3-class model trained for CoNLL data to extract persons, organizations and locations as discussed above. For this first extraction of an implicit network, we only perform entity extraction without entity linking (we later also extract a network of Wikipedia that is constructed with entity linking in Chapter 4.2). As temporal tagger and for the normalization of temporal expressions, we use HeidelTime [188] instead of StanfordNER, since Wikipedia articles largely follow a narrative structure and we therefore need a domain sensitive temporal tagger that can be adapted to this narrative domain instead of the typical news domain [189]. For the stemming of terms, we use the Snowball stemmer that is an implementation of the Porter stemming algorithm [149]. We set the cut-off parameter for the distance between entities to $c = 5$, since this value allows the use of 32-bit single precision floating point numbers for storing the exponentially diminishing edge weights without the risk of numeric instability, and thus helps in keeping the graph size manageable. We use the hierarchical completeness condition as discussed in Chapter 3.2 for dates, and the inclusion by splitting for the names of persons and locations.

In the annotation phase, we find that there are about 4.6M English Wikipedia articles that contain at least one entity mention. The documents can be split into a total of 91.4M sentences, 53.5M of which contain at least one annotation. We find a total of 137.0M in-

	Loc	Org	Act	Dat	Ter	Sen	Doc
Loc	0						
Org	90.8	0					
Act	275.8	105.7	0				
Dat	83.0	45.5	127.6	0			
Ter	182.8	93.9	316.6	57.3	0		
Sen	71.3	20.9	84.4	38.3	412.2	0	
Doc	0	0	0	0	0	53.5	0
$\lvert\mathcal{V}\rvert$	2.7	3.4	7.1	0.2	4.9	53.5	4.5

Table 3.1: Number of edges (top) and nodes by entity type (bottom) of the implicit network constructed from the English Wikipedia (in millions).

stances of entities, which are divided into 27.0M of class date, 44.2M of class location, 44.4M of class person and 21.3M that are annotated as organizations. After the extraction of cooccurrences and the aggregation of edges, we obtain an implicit network, for which we give the metrics in Table 3.1. We observe that the number of distinct dates is by far the smallest due to the selected granularities year, month, and day. However, the coverage of dates in Wikipedia is above 50% for dates after the middle of the 16th century and perfect for dates after 1800 [185], meaning that each such date is mentioned at least once. The large number of terms can be explained by the presence of technical terms, mismatched names or locations, and numeric data, which we did not model as separate entities (although it is a possible extension of the model). While it would be possible to reduce this number by limiting the terms to a dictionary, this approach might be too restrictive in many applications, especially for a large collection such as Wikipedia.

3.4.2 Implementation architecture

Since the resulting graph representation of the Wikipedia implicit network is quite large, an adequate implementation architecture is necessary to obtain acceptable query speeds for subsequent tasks. Here, we consider two architectures. A purely memory-based approach is possible through the representation of the graph as a list of nodes with five types of adjacency lists. That is, one adjacency list is used per adjacent class (thus, six lists in the case of nodes representing sentences, and one list for nodes of type documents). Edge weights are stored in an identical structure. While such a representation is highly efficient with regard to answering neighbourhood queries, it also requires large maps for the lookup of entity names, which tend to take up large amounts of memory. Alternatively, an external database can be used to store most of the information. With efficient indexing,

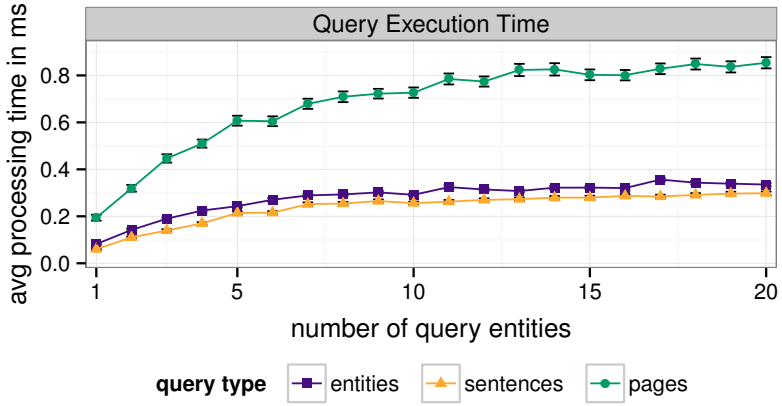

Figure 3.4: Processing speeds of queries for entities, sentences, and documents (Wikipedia pages), in respect to the number of input query entities. The results are averaged over 1,000 samples, and error bars denote the standard mean error.

the retrieval of node neighbourhoods is then slower but still possible. Hybrid approaches between the two architecture solutions are viable. For our experiments in the following, we store only the sentences in an external database, as these are equivalent in size to the entire original document collection. For the physical storage of the network itself, we use a custom Java implementation that is optimized for adjacency queries, and directly ties into our query interface. We later discuss an approach that is entirely based on using an database for storing the graph in Chapter 4.2.

Using the data, tools, and data structures as described above, the annotation of the data requires approximately 4 days on a dual Intel Xeon E5-2650 CPU with 20 physical cores and 256GB main memory, where the majority of the runtime is required for annotating entities. The initial extraction of the implicit network in our Java implementation then takes 46 hours using a single CPU core. The resulting graph has an uncompressed size of about 70GB on disk when edges are stored in one direction or 110GB if they are duplicated for faster initialization of the query interface, which takes about 20 minutes to load the data into an indexed in-memory representation. In our Java implementation of the entity query command-line interface, the graph requires about 130GB of memory, mostly due to the size requirements of hashmaps that contain the string representations of entities used to match input query strings to entity and term identifiers. Despite these expensive initial computations, the resulting graph allows for near real-time information retrieval and browsing of relations as shown in Figure 3.4. Even for higher counts of entities in a query, the processing takes less than a millisecond on average due to the representation of the sparse graph in adjacency list format (queries for multiple highly connected entities may take up to a second). Document recommendation queries for linking entities to

Wikipedia pages take slightly longer, since the connections between entities from all sentences in the documents are considered. For sentence queries, the shown runtime does not include the lookup of the actual sentences in the external database, but only the retrieval of the sentence identifier. In summary, we find that the processing time of information retrieval operations is insignificant given the size of the document collection, and is thus competitive with search engine speeds, especially since queries are processed only serially in this implementation. This is one of the primary advantages over alternative models, since the utilization of the sparse local neighbourhood structure of entities in the network is extremely fast for the majority of neighbourhoods.

3.4.3 Exploratory results

Based on the implicit network of Wikipedia as constructed above, we first investigate a number of example applications. Due to the versatility of the representation with regard to the types of entities that can be used in a query, we only consider a selection of possible application scenarios. However, all node-based rankings as described in Chapter 3.2 are viable, and are implemented in our query interface. For the explanation of the following examples and the subsequent evaluation, we introduce the concept of a *subquery*. Here, a subquery simply entails the splitting of input strings for entities of type *location* and *actor* into their components, which are then included in the query as well. The queries thus benefit from the completeness condition that is applied during the construction of the network. In the following, we use the syntax

$$\langle OS : (QS, value)^* \rangle$$

to describe queries, where $OS, QS \in \{\text{Loc}, \text{Org}, \text{Act}, \text{Dat}, T, S, D\}$ are the desired type of output set and the type of query entities, respectively, while *value* is the name of the query entity. Based on this syntax, we show a couple of results for example queries in the following, split into three primary use cases.

Browsing

The most straightforward application of the network is the ability to browse the connections between entities. In Table 3.2, we show the top-ranked results of three queries centred on the entity *Edward Snowden*. By including Snowden as the only query entity, we obtain a list of organizations that are closely tied to him, including the *NSA* and *USIS*, the company that vetted Snowden prior to his employment. Duplicate entries for the different

⟨Org : (Act, Edward Snowden)⟩

rank	organizations	score
1	nsa	1.000
2	national security agency	0.288
3	gchq	0.182
4	us national security agency	0.083
5	usis	0.043

⟨Org : (Act, Edward Snowden), (Act, Barack Obama)⟩

rank	organizations	score
1	nsa	0.546
2	senate	0.503
3	congress	0.340
4	republican	0.290
5	democratic party	0.283

⟨Ter : (Act, Edward Snowden)⟩

rank	terms	score
1	surveil	1.000
2	leak	0.985
3	document	0.610
4	whistleblow	0.532
5	contractor	0.496

Table 3.2: The five top-ranked results for three queries centred on *Edward Snowden*. Weights are given as the normalized directed importance weights $\vec{\omega}$, or their combination ϱ in queries with multiple entities. All terms are stemmed.

spellings of the NSA are artefacts from the named entity recognition step, which suggests that the implicit network representation can also be used as a tool for disambiguation or co-reference resolution. Moving beyond single-entity queries, when we include *Barack Obama* as a second query entity, the focus of the results shifts from security agencies to politics, where the Snowden incident was discussed. In contrast to querying for entities, we can also select the set of terms as output, as shown at the bottom of Table3.2). In this example, we find that the extracted terms provide a solid first impression of what made Snowden famous. Thus, we find that terms in the network can serve to describe entities or their relations. Overall, such browsing can be used as a tool for following connections and cooccurrences of entities through the data to explore events or relations, much like a knowledge graph. Due to the query speed, an interactive exploration is feasible.

Summarization

Based on the relationships of entities and terms in the graph, extracting descriptions for entities is no different from querying for any other type of target node. However, we can also query for sentences that contain the relevant entities directly to obtain extractive summaries. For example, the resulting top-ranked sentence for the query

$$\langle \text{Sen} : (\text{Act}, \text{Edward Snowden}), (\text{Org}, \text{NSA}) \rangle$$

is *"In early 2013, thousands of thousands of classified documents were disclosed by NSA contractor Edward Snowden"*, which summarizes the relationship nicely. While the current approach of locating sentences that contain the specified entities is straightforward, more intricate summarization metrics can be adapted to the existing graph structure, as we discuss in detail in Chapter 4.3.

Entity and concept linking

Since we constructed the graph from Wikipedia texts, we can also use it to recommend documents for entities, and effectively link entities to Wikipedia pages. The query

$$\langle \text{Doc} : (\text{Act}, \text{Edward Snowden}) \rangle$$

unsurprisingly yields Snowden's Wikipedia page as the top-ranked result. However, we can link more complex or combined concepts as well, for example with the query

$$\langle \text{Doc} : (\text{Act}, \text{Edward Snowden}), (\text{Ter}, \text{Surveillance}) \rangle.$$

For this query, the Wikipedia page for *Global surveillance disclosures since 2013* is placed ahead of Snowden's page in the ranking, since it lists the surveillance activities that Snowden uncovered in greater detail that the page describing the Snowden himself. In this context, the concept of subqueries is helpful, since persons are rarely referred to repeatedly by their full name on their own page. For example, the query

$$\langle \text{Doc} : (\text{Act}, \text{Albert Einstein}) \rangle$$

ranks the Wikipedia page for the *Albert Einstein Transfer Vehicle* highest, which was used to supply the International Space Station. The reason for this is that it is always referred to by the full name, while Albert Einstein himself is more often referred to as Einstein. Allowing

date	event description
1918-07-08	**Ernest Hemingway**, **Red Cross** volunteer, wounded in **Italy**
1960-10-12	**Nikita Khrushchev** pounded his desk with a shoe during a **UN** speech
1966-09-08	"Star Trek" debuted (**NBC**)
1973-10-06	**Israel** attacked by **Egypt** and **Syria**
1866-11-20	Bicycle with a rotary crank patented (**Pierre Lallemont**)
1960-08-19	**Francis Gary Powers**, U-2 spy plane pilot, convicted in a **Moscow** court

Table 3.3: Examples of events from the even participation evaluation data set that are annotated for actors, locations and organizations, together with the date on which the event took place. Note that these events are not necessarily represented by a single sentence in Wikipedia.

subqueries fixes this, and results in the physicist's page being top-ranked. While the transport vehicle never should have been tagged as a person during named entity recognition, such an error is common even with state-of-the-art tools. However, our findings indicate that implicit networks could be used to detect such inconsistencies. We encounter the task of document linking again for the recommendation of documents based on selected entities in Chapter 4.2, and for the linking of network topics to news articles in Chapter 6, where we provide additional use cases and examples.

3.4.4 Evaluation data

To evaluate the performance of the ranking of entity relations, we require entity-centric event descriptions for events that are contained in Wikipedia. Since we constructed the implicit network from Wikipedia data, we cannot turn to the Wikipedia event pages that are frequently used for evaluation in this context. To our knowledge, there are no data sets that have already been annotated for entities and are suitable for an evaluation of the content of Wikipedia. For this reason, we annotate a set of events by hand. Specifically, in order to avoid using Wikipedia data, we obtain a list of events from the World History Project, which provides trivia about events *On This Day in History* [82] and predates Wikipedia. This website contain lists of historic events for each day of the year, as well as dates of birth and dates of death of more or less famous persons. The site contains 6,558 dates of birth with a brief mention of the person's profession, and 1,483 dates of death along with a sentence about that person's accomplishments or circumstances of death. Furthermore, it includes descriptions of 5,805 diverse historic events ranging from scientific discoveries to well-known events. In the following, we describe the preparation and annotation of the data to obtain a ground truth of events.

Historic events

As historic events, we consider the set of dates that remain when dates of birth and dates of death are removed. For these events, we have a date (with a granularity of days), as well as one sentence that describes the event. We randomly select 500 such events and annotate them by hand for the mentions of persons, locations, organizations, and remaining terms. A few examples of such events are shown in Table 3.3. It is evident that this data is very diverse in both structure and content. Structurally, events consist of differing numbers of entities of different types, and have varying numbers of terms. Based on the content, the data can be categorized into various areas of life such as culture, politics, or war, and ranges from well-known world events to minor descriptions, such as the patent on a bicycle crank. We specifically choose this set of random events instead of manually selected events to minimize the bias and provide a more challenging evaluation data set than the dates of death baseline we introduce next.

Dates of death

The dates of death serve as a baseline since they are likely represented by a single sentence in Wikipedia. While the dates of death are not provided separately from the historic events in the data set, they are easy to distinguish from other events based on their descriptive sentences, which always end with the word *"died"*. Thus, we use pattern matching to separate them from the historic events. Afterwards, we manually annotate the data to strip titles from names, split the description from the name and separate multiple names and nicknames. For example, the sentence *"H(oward) P(hillips) Lovecraft, horror writer, died."* yields the information that Lovecraft was a horror writer, which we can use as terms to describe him, but it also provides the alias *H.P. Lovecraft* in addition to his full name. We store this information together with the provided dates for a total of 1,483 persons.

3.4.5 Event completion evaluation: date prediction

To evaluate the performance of entity-centric rankings that use the Wikipedia implicit network based on the two ground-truth event data sets, we use the description of events as query input. Specifically, we use all entities of type location, organization, and person as query input and evaluate the resulting ranking of dates with respect to the known ground truth date of the event. Thus, for each event, the query input is a set of entities. For querying strategies that include the context, this set may also contain terms. We focus on the prediction of dates since there exists exactly one date of granularity day per event in the

evaluation data set. Furthermore, the coverage of dates in Wikipedia is very comprehensive for the considered time frame [185], which reduces the risk of erroneously evaluating the completeness of Wikipedia or the performance of the named entity recognition tool. For the output ranking, we only consider dates with a granularity of days since this is the granularity in the evaluation data sets.

Ranking methods and baselines

Based on this evaluation task, we test several possible ranking methods that utilize the implicit network model. With LOAD, we denote the basic method of inputting all query entities and ranking dates according to the combined edge scoring function ϱ. For LOADsq, we use the same approach but also enable subqueries to split names into components. For LOADTsq, we use the descriptive terms of events as part of the query in addition to the participating entities. Similarly, LOADTsq+ is a special query for the dates of death evaluation set, in which we also include the terms *death* and *died* in addition to the entities and terms that are contained in the event description. Furthermore, we use two baseline approaches. For BASEr, we rank dates that are connected to any query entity at random (this process is repeated 1,000 times and the results are averaged). Conversely, BASEw ranks dates by their weighted connection $\vec{\omega}$ to any one entity in the query.

Evaluation results

In Table 3.4, we show the results for the dates of death and historic events evaluation data sets. Independent of the data set, we find that the LOAD methods are able to locate an overall larger number of correct dates at some position in their ranking. The number of identified dates rises as we include more entities and terms in the queries. In the case of dates of death, the LOAD approach outperforms the randomized baseline, but is almost equivalent in performance to the weighted baseline. Since we only have one class of input query entity in this case (namely persons), this result is expected. The overall best performing approach is LOADsq with its included subqueries, which retrieves the largest number of correct dates that are ranked at the top position. The results are similar for the historical event data, although the weighted baseline performs much poorer with the addition of further entities to the input query. Interestingly, the inclusion of terms in the query helps with the overall recall but not with the precision of the results, indicating that entities are better suited as query input than the overall more frequent terms. This observation is reinforced by the results of LOADTsq+, which ultimately has the best recall values at the cost of a low precision.

data	method	found	missing	cor@1	prc@1
dates of death	LOAD	869	613	122	0.082
	LOADsq	1326	156	**207**	**0.140**
	LOADTsq	1374	108	125	0.084
	LOADTsq+	**1443**	**39**	13	0.009
	BASEw	869	613	206	**0.140**
	BASEr	869	613	63.25	0.043
historic events	LOAD	290	210	39	0.078
	LOADsq	341	159	**40**	**0.080**
	LOADTsq	**414**	**86**	33	0.066
	BASEw	290	210	19	0.038
	BASEr	290	210	0.48	0.001

Table 3.4: Evaluation results for both event data sets. Shown are the used methods and the two baselines, the number of identified dates, the number of missing dates that are not predicted at any rank in the results, the number of top ranked dates that were in the evaluation data sets (cor@1), and the resulting precision at rank 1 (prc@1). The best achieved score for each metric and data set is highlighted in bold.

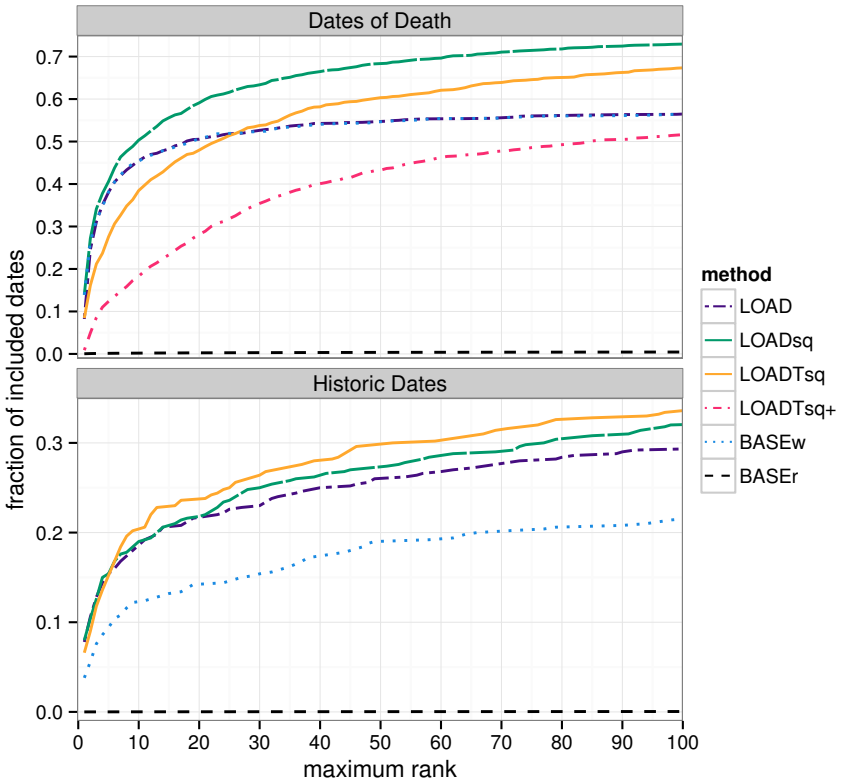

Figure 3.5: Prediction of dates based on the event data. Shown is the fraction of identified correct dates in the top k ranks (recall@k) versus the rank k.

In order to evaluate the browsing potential of the method when used as an entity-centric search engine, we also consider the number of correctly identified dates with progressing rank in the list, as shown in Figure 3.5. Here, we find that LOADsq performs best for the date of death data, for which it includes 50% of the correct dates in the first 10 positions of the ranking. In comparison, the performance of LOADTsq is worse. Even though LOADTsq and LOADTsq+ ultimately include the most correct dates, this happens at a point in the ranking where it is of little relevance, unless this were to be used as a pre-selection step for further analyses. For both data sets, the LOAD methods with subqueries perform significantly better than either baseline. For the historic event data, the inclusion of terms leads to a further improvement over the other LOAD methods, which emphasizes the need to include terms in the model, since they may be essential components or even the only known features of an event description that is contained in the texts of the document collection. We further explore this importance of terms (and the context that they provide) for event relations in the following chapters.

3.4.6 Related work beyond event completion

While there are no previously published baseline methods for the task of event completion or the entity-centric prediction of dates, there is some related work that indicates possible future directions and applications for implicit networks. In particular, the extraction of temporal events, event dating, and graph-centric entity retrieval are related fields, as we briefly discuss in the following.

Temporal information retrieval

Temporal information is prevalent in many documents across application domains, and the extraction of temporal facts and events provides a method of inducing structure in unstructured document collections, due to the ordering aspects of time. Thus, exploiting temporal information in documents has become an important part of information retrieval [37], which has recently seen numerous advances in this direction. Therefore, a number of approaches in the literature focus on linking temporal and entity information for event retrieval and description, similar to our date-based evaluation.

For example, Setty et al. generate timelines from news articles by highlighting important dates for a specific user query (for example, about persons or events) [165]. To this end, they exploit the document creation times of news articles under the assumption that the top-k time-travel query result for the topic of interest changes significantly at important

times. Huet et al. use structured data in the form of a knowledge base to mine temporal trends or assess the importance of entities [101]. In an approach that is also based on an external knowledge base of news articles, Gupta and Berberich identify time intervals of interest for given keyword queries based on pseudo-relevant documents [83]. They employ a probabilistic approach for the selection of suitable documents for a given query and generate a time interval from the contained temporal expressions. Kanhabua and Nejdl analyze temporal anchor texts extracted from Wikipedia's edit history to track and detect the evolution of entities and events [105]. Similar to the approaches described above, temporal metadata (in this case, the edit history) is used to discover time-related knowledge.

In contrast to implicit networks, all of the above methods use not only the document collection and the temporal information that is available in the unstructured content of the documents, but also the document creation time as external metadata. Thus, a possible extension of implicit networks could include such external temporal information and use it to enhance the model, for example in the aggregation of edges. We discuss this in further detail in Chapters 5 and 6, where we extend implicit networks to streaming data.

Entity-centric analyses

Articles that follow a more entity-centric focus include the analysis by Filannino and Nenadic, who extract temporal footprints of objects, persons, or historical periods from encyclopedic descriptions of Wikipedia articles [60]. Das Sarma et al. use dynamic relation graphs to identify the specific entities that participate in trending events [160]. Abujabal and Berberich also address the problem of extracting events from semantically annotated document collections. Based on methods from frequent itemset mining, they identify and rank events in relation to named entities [1]. In contrast to implicit networks, however, they consider events to be represented by sentences and therefore cannot extract events that are spread across multiple documents or sentences. Kanhabua et al. assess the importance of temporal expressions for events based on entities and features [106]. They extract events from the cooccurrences of entities and locations, but also restrict the cooccurrences to those within documents and single sentences.

While these approaches are similar to our evaluation of implicit networks in exploiting the entity information contained in the documents' texts, they focus on single documents. In contrast, we do not analyze single Wikipedia pages or use concept templates, but rather study the relationship between dates and content (for example, entities, events, or keywords) in a general and global way. For this, we consider the full document collection at

once without limiting our knowledge extraction to specific concepts. However, by focusing on sentence retrieval, our model could also be used to reduce the extracted information to the sentence level for a multi-way summarization approach of points in time, entities, and concepts.

Timeline and fact extraction

Finally, there are a number of systems for building knowledge bases of temporal information or timelines. One such system, CATE, extracts the context of entities and presents them in a timeline structure [202]. Kuzey and Weikum provide a framework for the extraction of temporal facts and events from the content and structure of Wikipedia articles by means of pattern recognition and rule-based extraction [113]. A similar approach is used to extend YAGO to include temporal knowledge [209].

Unlike these systems, we do not aim to provide a knowledge base in the traditional sense, but a method for exploring and browsing a previously unknown document collection of unstructured text in respect to not only time but all entities that participate in real-world events. In contrast to these knowledge bases, the entity-centric design of implicit networks encompasses many further options for browsing and retrieving entity relations and text snippets from the underlying document collection, as we show Chapter 4.2.

3.5 Evaluating Implicit Networks Versus Embeddings

In recent years, vector embeddings have become the predominantly used representation of words, as we have already discussed in Chapter 2.1.3, and are based on the intuition that the context of words in a corpus can be mapped to a continuous, low-dimensional vector space. Since these embeddings can be learned without supervision, their appeal is obvious. As a result, they have been used for a multitude of tasks in natural language processing and information retrieval, such as addressing the problem of vocabulary mismatch [65], named entity recognition [203], or improved query expansion [49]. In addition to their typically small and fixed dimension, one of the main advantages of embeddings that makes them easy to use in learning tasks, is the underlying notion of taking words that occur in similar contexts in the documents, and embedding them in close proximity in the vector space. Therefore, it seems reasonable to assume that embeddings could perform well for the event-based entity prediction tasks that we have used to evaluate the implicit network.

Conceptually, it is a simple matter of obtaining a similarity of terms or entities by comparing their vector embeddings with some vector-based similarity measure, and such approaches have been used in the past, for example in the disambiguation of entities [233]. However, since embeddings tend to encode a similarity of contexts, whereas implicit networks encode a relatedness of occurrences, it is not immediately obvious if embeddings are truly adequate for solving the task. In the following, we therefore investigate the performance of word embeddings for the task of predicting entity participation in events, and use implicit networks as a baseline for the comparison.

Attribution. This section describes joint work. The training of the word embeddings, and the implementation and evaluation of embedding-based neighbourhood predictions were performed by a collaborator.

3.5.1 Predicting event participation with embeddings

As evaluation task, we consider a similar event participation prediction of entities as in Chapter 3.4. That is, given an incomplete set of entities that participate in an event, the task is to predict the remaining entity. As we have already seen, this is not a complicated task when using an implicit network representation. However, when we use embeddings as a representation, it becomes more complex. Intuitively, we want to rank entities by their proximity to the query entities. For a single query entity, this task can be solved as a simple nearest neighbour search in the vector space, starting from the vector coordinates of the query entity, based on some suitable distance measure. For multiple input entities, we need to rank each candidate entity according to its proximity to multiple starting points in the vector space. In order to use neighbourhood search in a meaningful way for multiple query entities, we therefore need to either rank the target entities in relation to multiple input entities, or generate a centroid vector of the set of starting points that correspond to the input entities. In the following, we describe such combination approaches.

Vector-based scoring functions

Let $Q \subseteq E$ with size $|Q| = n$ denote the set of query entities, and let X be the set of possible target entities for which we wish to obtain a ranking. Each query entity $q \in Q$ is then associated with a vector embedding ε_q of this entity. Then, our aim is to derive a scoring function $\varrho : E^n \to \mathbb{R}^X$ that maps each entity in the target set to some score that determines how closely related it is to the target entities. Once we have such a score, we can rank all candidate entities by their score and treat the output of the embedding-based approach

in the same way as the implicit network approach. In the following, we consider three different approaches to the construction of a joint scoring function.

Sum of the distances. As a first combination method, we consider the sum of minimal distances to all query entities. Thus, we obtain

$$\varrho_{\text{SUM}}(Q, x) := \sum_{q \in Q} dist(\varepsilon_q, \varepsilon_x) \tag{3.14}$$

for some vector distance function $dist$. To find the best target entity for our prediction, we then rank all entities in the target set X in ascending order by their ϱ_{SUM} score.

Minimum maximum distance. In contrast to the sum of distances, we can also consider minimizing the maximum distance to any query entity. Formally, we let

$$\varrho_{\text{MINMAX}}(Q, x) := \max_{q \in Q} dist(\varepsilon_q, \varepsilon_x), \tag{3.15}$$

which we can then use to rank entities in the target set X in ascending order by their achieved ϱ_{MINMAX} score.

Distance to the average query vector. In contrast to the above two approaches, we can also create a centroid vector from the individual vector representations of the query entities, based on which we then identify the nearest neighbour. Therefore, we obtain

$$\varrho_{\text{AVG}}(Q, x) := dist(\langle \varepsilon_q \rangle, \varepsilon_x) \qquad \text{with} \qquad \langle \varepsilon_q \rangle := \frac{1}{|Q|} \sum_{q \in Q} \varepsilon_q \tag{3.16}$$

We again rank entities in the target set X in ascending order by their ϱ_{AVG} score.

For each of he three combination approaches, we obtain a ranking of all candidate entities. In each case, the first-ranked entity in the list can be regarded as our prediction, and we can evaluate further ranks of the list to obtain recall scores.

3.5.2 Evaluation data, task, and setup

Before we evaluate the embeddings against the implicit network rankings, we introduce the evaluation task and data set, as well as the word embeddings that we utilize.

Evaluation data set

One downside of embeddings tends to be the amount of time that is required to train the models. This is problematic since we cannot rely on pre-trained out-of-domain embed-

dings for this comparison, but rather have to train embeddings on the same document collection from which we extract the implicit network. In particular, the training effort that is necessary to train embeddings in the entire Wikipedia would be exceedingly large. Therefore, we use a smaller set of news articles that we later also use for the evaluation and exploration of dynamic implicit networks in Chapters 5 and 6. Since the nature of the data does not influence the results, and since the evaluation task works similar to the event completion task that we already considered, we do not introduce the data set or ground truth here in-depth. For details on the data and how it is collected, we refer to Chapter 5.4. In summary, the data is comprised of 127,485 English news articles that we annotate for named entities of the types actor (person), location, organization, and date, similar to the Wikipedia data set.

Event completion task

The event completion task on news articles is constructed in the same way as for the Wikipedia data. We consider extracted events to be sets of entities, from which we generate queries by removing one entitiy and using it as the target entity, while the remaining entities serve as the query set. For a detailed description of how the set of ground-truth events is generated in the case of news articles, we refer to Chapter 5.5. Overall, we obtain an evaluation set of 97 news events, from which we construct 293 distinct queries. However, due to the occurrence window sizes that are used to train word embeddings, which are typically much smaller than the windows we can use to extract implicit networks, not all queries can be answered by all approaches. After removing these queries, we are left with 263 distinct queries that we use for our evaluation. To generate predictions from the implicit network, we then proceed in the same way as for the evaluation on the Wikipedia implicit network. For the embeddings, we generate a ranking of all potential target entities with the proper type (that is, the same entity type as the true target entity) in the neighbourhood of the query entities.

Embedding approaches

While embeddings are popular for many tasks, there is no universally best embedding that performs well on every possible task [162]. Therefore, we use three different approaches for generating the word embeddings. Specifically, we use the continuous bag-of-words (cbow) and the skip-gram approach from word2vec [131], as well as global vector embeddings (GloVe) [146]. For a more detailed introduction of these methods, see Chapter 2.1.3.

To generate the training data for all approaches and make the results comparable to the implicit networks, we use the entity annotated version of the document collection. That is, we replace all mentions of named entities with the corresponding Wikidata identifiers. The resulting embedding vectors of the Wikidata identifiers then serve as the vector representations of the entities that we use in the evaluation.

Experimental setup

Like most unsupervised learning approaches, word embeddings require a number of hyperparameters to be tuned during the training to optimize their performance. Therefore, we conducted an extensive search of hyperparameter settings for each of the embedding methods. In the following, we only report the results of the settings that maximized the precision@1 scores as the primary performance metric. We train all three embeddings for 100 training epochs, in the dimensions 50, 100, and 200. Since the results of training the embeddings are nondeterministic, we repeat all experiments 10 times and report the average results. For cbow, the best performance is obtained for 15 negative samples, a context window size of 21, a minimum of 3 occurrences for a word to be included in the training data, and a down-sampling threshold of 10^{-5}. For skip-gram, the best down-sampling threshold is also 10^{-5} at a window size of 21. Glove also performs best at a context window size of 21, with values of $\alpha = 0.6$ and $x_{max} = 25$ for weighting the least squares objective during the generation of the embeddings.

As distance function for determining the neighbourhoods and predicting entity participation, we utilize the cosine distance.

3.5.3 Evaluation results

We begin the evaluation by considering the three different combination approaches for multi-entity queries in the vector embeddings. As shown in Figure 3.6, both SUM and AVG perform equally well in terms of recall, while the precision@1 (that is, the recall value for rank $k = 1$) is better for SUM. In contrast, MINMAX performs considerably worse and never reaches a recall value above 0.6 at rank 10. Therefore, we consider SUM to be the best combination approach for embeddings. In the following, we thus focus on SUM as a representative of the embedding-based prediction approaches, since it effectively serves as an upper bound for all observed results. In comparison to the implicit network, all embedding-based approaches perform much worse, as we can also see from the values in Table 3.5. Overall, we find that the implicit network has a significantly higher performance

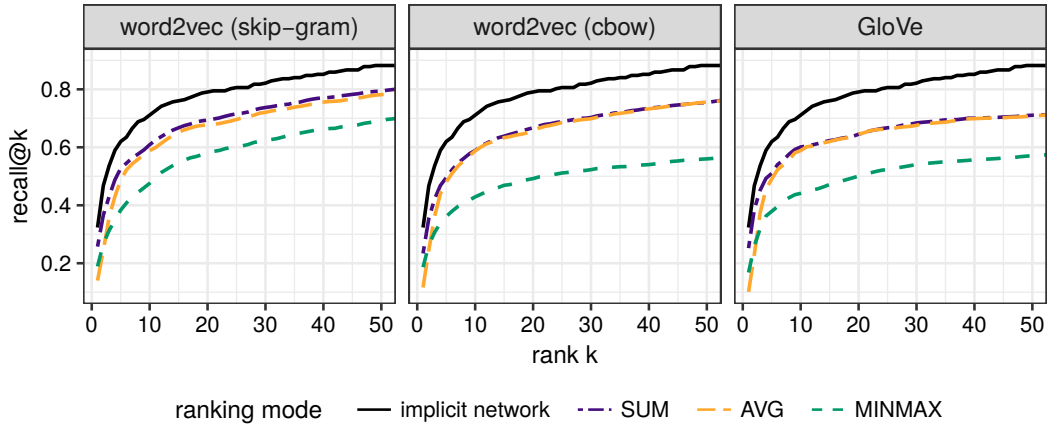

Figure 3.6: Comparison of the performance of embedding-based prediction in the event completion task for the word embedding models word2vec (cbow), word2vec (skip-gram), and GloVe. Shown is the recall@k for the three query embedding combination modes SUM, MINMAX, and AVG on embeddings of dimension 200. The performance of the implicit network is included as a baseline. Performance values are averaged over 10 iterations.

	prc@1	rec@10	rec@50	rec@100
word2vec (cbow)	0.221	0.589	0.754	0.819
word2vec (skip-gram)	0.229	0.609	0.794	0.875
GloVe	0.215	0.601	0.711	0.752
implicit network	**0.330**	**0.711**	**0.882**	**0.932**

Table 3.5: Average precision@1 and recall@k values for the event completion task and SUM combination mode and an embedding dimension of 200. Performance values are averaged over 10 iterations. The best performances for each metric are highlighted in bold.

in both precision@1 and recall@k for $k \leq 100$, indicating that word embeddings are not well suited for this entity-centric task in comparison to the implicit network.

In Figure 3.7, we consider the relative performance of the three embedding techniques for the SUM combination approach, as well as the effect of adjusting the dimension of the embeddings. We find that skip-gram tends to perform better than cbow and GloVe, but the overall results are fairly similar. In contrast, the embedding dimension has a significant impact, as the results of all embedding methods increase with the dimension of the embeddings. However, even given this trend, embedding dimensions that might lead to results that equal the implicit network are not feasible due to the increase in training time.

Given these results, an interesting aspect to consider is the influence of the frequency of entities in the training data on the obtained results. Intuitively, it should be harder to train embeddings for entities that occur very infrequently due to the lack of training data. In

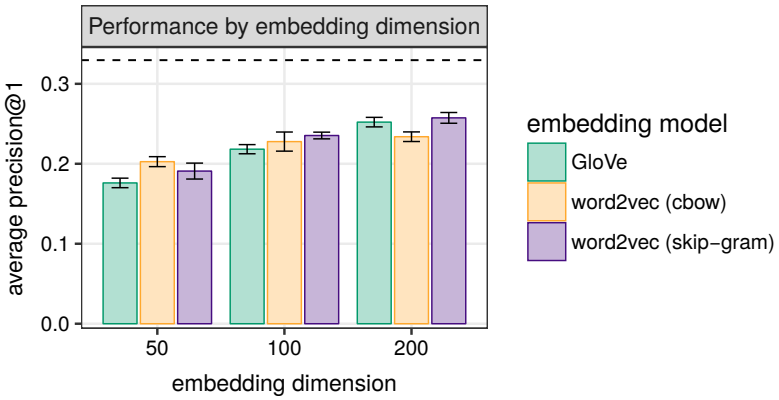

Figure 3.7: Evaluation of the dependence of the predictive performance of embeddings in the event completion task on the embedding dimension. Shown are the average precision@1 for the combination mode SUM, averaged over 10 iterations. Error bars denote one standard deviation. The dotted line denotes the performance of the implicit network.

contrast, implicit networks can be expected to perform well due to the local sparseness of the graph in the neighbourhood of infrequent entities. In Figure 3.8, we therefore show the average rank of target entities in dependence of their frequency in the training data. As expected, we find that the implicit network performs best overall. However, while all methods do not perform well for extremely rare entities (that is, entities with frequency ≤ 10), skip-gram actually performs best for these entities, and is the best-performing embedding overall. In contrast, the performance of GloVe is much more erratic, despite the fact that it does not perform significantly worse than the other embeddings overall. Therefore, it seems reasonable to assume that GloVe captures some of the entity relations exceedingly well, while failing for others. This indicates that ensemble methods between implicit networks and embeddings might be useful for capturing the context of entity relations, as we further explore in Chapter 5.

3.5.4 Comparing embeddings and implicit networks

While embeddings are a popular and widely used representation, not least due to their compactness and versatility, they are not universally usable for every task, as we have seen in this evaluation. This poor performance may in part be caused by the overall sparseness of entity mentions in the documents that starves the learning approaches for sufficient training data. An alternative explanation is the considered task. Embeddings tend to focus on the similarities between words by embedding words that occur in the same context close to each other in the vector space. As a result, they perform well on similarity-based tasks such as the identification of synonyms. However, they tend to perform less well

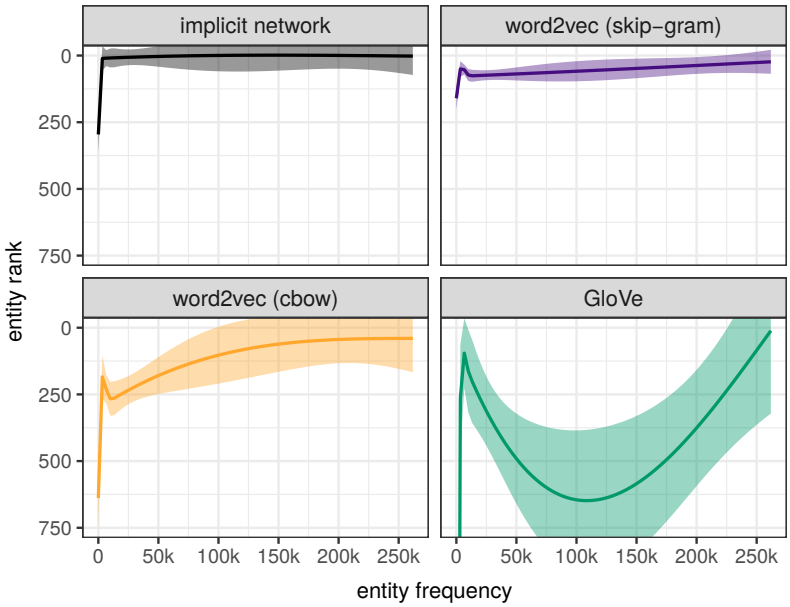

Figure 3.8: Comparison of the average predictive performance in the event participation task versus the frequency of entities in the training data. Shaded areas denote the 0.95 confidence intervals of the fit.

for tasks that are based on relatedness, such as the event participation that we consider here. Especially in the case of embeddings for entity-annotated texts, word embeddings tend to perform less well for relatedness-based tasks than they do for tasks that require similarity [8]. Given the notion that learned features typically show better performances than manually designed features when a variety of tasks is considered, this raises the question how word embeddings could be tweaked or combined with implicit networks to achieve a better performance.

In addition to the differences in *how* the context of entities is represented by word embeddings and implicit networks, there is also the question of *what* is embedded. For word embeddings, the context windows are typically kept very narrow and tend to cover little more than a sentence. In our experiments, we already increased the context window size and found that the performance improved substantially over the default settings. However, increasing the window size comes at the cost of increased training times, and diminishing returns arise for window sizes that are too large, likely due to more included noise. We found the context size of 21 to be at the upper limit of what is still feasible, with training times of up to 18 hours per model on a dual-core machine even for this smaller document collection. In contrast, implicit networks capture entity relations at much greater distances at a fraction of the computational effort due to their increased cross-sentence window sizes.

Here, we see the benefits of considering entities and terms separately. While an increased context window size is sensible for entities, it would introduce noise for terms that are less likely to be related across sentence boundaries. We see this as confirmation of the potential behind an approach that uses different window sizes for entities and terms.

Overall, as a result of this comparison, we find implicit networks to be well suited for the exploration of entity relations in document collections. In the following, we therefore consider this exploratory aspect on the network level, which embeddings cannot offer.

3.6 Summary and Discussion

In this chapter, we introduced the implicit network representation of large document collections as the foundation of subsequent, more application-driven contributions in the remainder of this thesis. We did not go into application-specific details or data, and instead applied the network model for generic tasks. Specifically, we evaluated and tested the approach on Wikipedia, primarily because it is an enormous and free source of unstructured text that can be cleaned to remove noise, and thereby allows for an assessment of the scalability of our model. However, note that the implicit network model itself can be used to represent an arbitrary document collection, since it makes no assumptions about the documents' structures or origins. In particular, any entity-centric approach is bound to perform better on a corpus or document collection with a narrower scope of topics due to the improved performance of named entity recognition tools that can capitalize on the reduced ambiguity. Therefore, the performance of retrieval tasks on the network also stands to improve in such settings, in contrast to the very diverse Wikipedia data.

Practical implications

Recall the journalists and data analysts from our introduction in Chapter 1 that are working on such data as the Panama Papers or the FBI's investigation into collusion during the U.S. presidential election of 2016. For them, implicit networks offer the potential of looking at their data from new angles. By modelling their unstructured textual data as an implicit networks, they can now extract potential relations between arbitrary entities in the documents and consider them in context. Unlike our experiments, where we used typical named entities and standard entity annotators, such practical applications can include any type of (named) entity and any type of annotation process. For small data sets, manual annotation might be viable. However, a more likely scenario is gazetteer-based annotation,

in which lists of entities are matched against the text to annotate their occurrences and construct a network. Considering the practical application scenarios, it becomes apparent how comprehensive the notion of *entity* can be, and how it can include anything from the traditional names of persons or locations, to financial concepts, individual groups of people or factions, or even to relevant parts of technical terminology.

The network-centric view on any type(s) of such entities then provides the analysts or journalists with the ability of uncovering latent connections between focal concepts. By focusing on some entities or terms as input queries, they can explore the implicit relations in their neighbourhood and discover relations between them. However, not only does the network allow them to uncover and visualize such non-obvious relations, the fact that implicit networks also include structural elements, such as sentences or documents, even provides provenance information for these mentions. As a result, it becomes possible to start at entities or terms of interest, and identify documents with relevant content that mention implicit relations and contexts of which these entities are a part. Thus, the network model reduces the insurmountable task of reading thousands of documents to browsing a handful of documents that serve as starting points for uncovering previously unknown relations and affiliations.

Outlook

Beyond the domain of predominantly static input data that we have considered so far, such as Wikipedia dumps, an application of the model in a streaming or online setting is possible. There, it stands to benefit from the underlying graph representation, which allows incremental updates to the graph as new documents arrive in the stream. The addition of new documents to the model is possible in real-time, simply by processing the document and adding the resulting subgraph to the main graph structure. Even edits, additions, and deletions in the individual sections of a document can easily be accounted for. Since the adjustment of edge weights happens locally within the graph structure, a change graph can be computed for a document before and after an edit, and then used to update the implicit network. Thus, the model is able to handle the processing of frequently edited document collections as well as streaming sources such as news feeds.

Based on these deliberations, we focus on applications and refinements of the model in the following chapters. We begin by presenting further applications for implicit networks in practice on the example of two use cases in Chapter 4. Afterwards, we discuss an extension of the model to dynamic document collections and consider possible applications for implicit network models of streams of news articles in Chapter 5, where we also

address partial edge aggregation techniques and document time stamps. In Chapter 6, we investigate how implicit networks can provide a novel angle on topic detection in dynamic document collections, and how they support an interactive exploration. Finally, in Chapter 7, we consider a generalization of entity and term cooccurrences to a hypergraph network model and provide a formal basis for expressing queries to such a representation of documents.

4 Applications of Implicit Networks

Using the implicit network representation of entities and terms in documents that we introduced in the previous chapter, we now consider applications of this document model. To this end, we focus on two application scenarios in particular. First, we demonstrate the efficiency and effectiveness of the representation as both an index structure and a data source for localized explorations around entities and terms in the network by showcasing an entity-centric search engine that works on top of the network representation. Second, we focus on the extraction of sentences for the task of extractive summarization, and show how such descriptive sentences can be used to identify complex relations that cannot be detected by current knowledge extraction and knowledge base population techniques.

Contributions. In this chapter, we thus make the following three contributions.

I We demonstrate how an implicit network representation can be used to construct an entity-centric search engine with an interactive user experience.

II We propose, expand, and evaluate sentence ranking schemes for the extractive summarization of descriptive sentences from implicit networks.

III We investigate the extraction of complex entity relations beyond the capabilities of traditional relation extraction approaches on the example of toponyms.

References. Parts of this chapter are based on the peer-reviewed publications:

A. Spitz, S. Almasian and M. Gertz. "EVELIN: Exploration of Event and Entity Links in Implicit Networks". In: *Proceedings of the 26th International Conference on World Wide Web (WWW), Companion Volume*. 2017, pp. 273–277. DOI: 10.1145/3041021.3054721

A. Spitz, G. Feher and M. Gertz. "Extracting Descriptions of Location Relations from Implicit Textual Networks". In: *Proceedings of the 11th Workshop on Geographic Information Retrieval (GIR)*. 2017, 1:1–1:9. DOI: 10.1145/3155902.3155909

4.1 Overview and Motivation

As we have seen in our exploration of the Wikipedia implicit network in Chapter 3.4, implicit networks can be used to describe and extract latent entity relations from large document collections. Thus, it stands to reason that they can support a variety of tasks in information retrieval, knowledge extraction, and search. While such implicit relations offer less insight into the types of connection between entities than traditional knowledge bases, they are much easier to extract from unstructured textual sources, and can therefore be constructed from a wider variety of texts. The derived relationship strengths between entities can then be used to identify and leverage important co-mentions, based on which complex constructs of semantically related entities can be assembled with ease. In this chapter, we therefore explore applications for such implicit networks beyond the examples that we have previously considered.

For the first application scenario, we focus on the typical task of searching a document collection. Here, the entity-centric focus of implicit networks provides us with a unique angle and perspective that we can use to extract information through entity relations. Thus, we describe and demonstrate an entity-centric search engine that enables us to search for any type of node (meaning entities or structural elements such as sentences or documents), and rank them according to the user's input. In particular, since we can link nodes in the network to the entries of a knowledge base, we consider the scenario in which the input also consists of typed entities in addition to plain terms, thereby providing the user with additional input opportunities for a faceted entity search, and additional feedback about his selection of query entities.

Based on these options for ranking arbitrary nodes in the neighbourhood of query entities or terms in the network, we then focus not just on the relations between entities, but on describing these relations. As our second application scenario, we therefore investigate the ranking and extraction of descriptive sentences from the network representation, which can serve to describe or summarize relations. Using these relation descriptions, we then perform a contrastive comparison to traditional knowledge graphs and the discrete relations that they contain.

Structure. In Chapter 4.2 we give a broad overview of the network model's capabilities by presenting a user interface that can be used to query an entity-annotated version of Wikipedia for different types of input and output entities. Following this general exploration, we focus on sentence nodes in the network in Chapter 4.3, where we consider the task of extractive summarization and descriptive sentence extraction.

4.2 Interactive Entity and Event Exploration

The primary example of an implicit network for a large document collection that we have considered in the previous chapter is the LOAD graph of Wikipedia, which encodes the textual proximity of location-, organization-, actor-, and date-mentions in Wikipedia. While we have provided some examples of possible exploration, identification and summarization tasks that were centred on events and entity relations, these static examples can only scratch the surface of possibilities. In the following, we thus present the implementation of a user search interface that supports these entity-centric retrieval tasks on the Wikipedia implicit network. We discuss the working principles and architecture of such an application, and demonstrate how it can be used for the exploration of Wikipedia or generalized to other data sets or types of entities.

Attribution. This section describes joint work. The Web-frontend of the user interface was implemented by a collaborator.

4.2.1 Entity-centric exploration of documents

The data structures that are typically used for entity-centric search tasks are knowledge graphs, which are invaluable sources of structured, discrete relations between entities that have numerous applications in information retrieval and natural language processing. Tasks such as search, disambiguation, or question answering can easily be augmented by structured entity relations stored in knowledge bases such as YAGO, DBpedia, Wikidata, or the Google Knowledge Graph, either by using the relations between entities for the task, or by displaying additional information to the user. However, the practical applicability of such resources depends heavily on the existence of structured knowledge for the domain under investigation. In cases where this information is unavailable, implicit entity networks can be extracted directly from unstructured text to fill the role of knowledge bases and support information retrieval from the underlying document collection, or add useful entity relations. Furthermore, the weighted network structure that is extracted from entity cooccurrences enables the use of established information retrieval approaches that would otherwise be incompatible with the discrete, unweighted graph representation of a traditional knowledge base.

As a result, implicit network approaches offer an important augmentation to entity-centric search that the inclusion of knowledge bases alone cannot provide. While the integration of knowledge bases in search and retrieval tasks introduces additional, *external* knowledge, implicit networks induce a graph structure exclusively from the *intrinsic*

information that is contained in the documents and thus support exploratory measures for arbitrary document collections. In combination with knowledge bases and entity linking techniques, implicit networks of entities can serve not only as indices and data structures for the occurrence of entity mentions in the underlying document collection, but also aid in the construction and exploration of complex entity structures that directly enable the identification of events in which these entities participate. Therefore, we can use an implicit network representation to augment search applications in document collections when (sets of) entities are both the input as well as the output of queries.

In the following, we discuss EVELIN as a comprehensive system for the Exploration of eVent and Entity Links in Implicit Networks, and showcase entity-centric search, ad-hoc summarization, and subgraph exploration on the example of the Wikipedia implicit network. In contrast to our previous exploration in Chapter 3.4, we now merge unstructured and structured data by linking entity mentions in the text of Wikipedia to Wikidata identifiers to augment the search experience[1].

4.2.2 Related work on entity-centric search applications

Over the last years, a number of applications have been proposed that cover similar use cases. One of the first such systems is Broccoli, which extends SPARQL queries with textual cooccurrences [18] and combines full-text search with knowledge base search on Wikipedia. With a focus on news articles, STICS uses entity auto-completion to suggest related entities for queries [95], but does not include temporal information beyond the publication date. The system has seen a couple of updates, including the visualization of trends in entity occurrences [94]. More recently, it was updated to include weighted cooccurrences of entity n-tuples for auto-completion suggestions [161]. Several other systems also focus on the search and visualization of streams or collections of news articles, often in relation to entities in a knowledge base. Exposé introduces a time-aware retrieval approach for organizing news articles and linking them to Wikipedia as an event repository [136]. This method is mostly focused on a temporal order of events, but includes entities as facets for filtering the results. In a similar approach, Contextualizer serves as an interface for browsing news documents, by retrieving candidate documents based on user-selected keywords that are then ranked by contextual similarity and temporal aspects, and linked to external Wikipedia information as a context [201]. Geared towards the large-scale aggregation of news events, Event Registry is focussed on news articles,

[1] EVELIN is available online as a Web tool for searching a 2016 dump of the English Wikipedia, along with a user's manual: http://evelin.ifi.uni-heidelberg.de

named entity annotation and date identification [114]. To this end, articles are clustered by the contained named entities and content to obtain aggregate events. NewsStand offers a geographic event-centric visualization and exploration of news topics, including spatial keyword distributions [199]. It has been extended by a large number of additional aspects such as CrimeStand for spatially tracking criminal activity. With a similar focus on the geographic aspect, Frankenplace is designed as a search interface for interactive thematic maps [2]. It provides geographic context for queries to an underlying document collection and utilizes topic modelling to leverage both themes and locations as dimensions for the exploration of its search results. Moving beyond Wikipedia and news articles, Expedition is a system with a focus on scholarly search [171] that filters and refines documents interactively on a timeline based on the user's entity selections. Similarly, DigitalHistorian is a stand-alone tool aimed at temporal and entity-centric corpus exploration in the Digital Humanities [84]. InfoScout is designed for an augmented, subject-centric investigation of documents and is thus focussed on person mentions [128]. Based on an underlying knowledge base, XKnowSearch infuses traditional keyword searches with structured information by introducing an intermediate query entity graph layer that maps keywords to entities, which are then used in the search [228]. Similarly, SemFacet combines search capabilities with knowledge base support for faceted search using document metadata [12].

Finally, two recent approaches are conceptually the most closely related to entity-centric implicit network search. Ceroni et al. introduce a system for detecting and pinpointing events in document collections based on the cooccurrences of named entities [39]. However, they focus on the task of validating event occurrences in the collection. Similarly, the Entity Relatedness Graph is designed for the exploration of entity relations [3]. It uses the concept of entity similarity for ranking adjacent entities, but unlike our approach, it relies on the Wikipedia link structure for learning semantic relations between entities instead of using cooccurrences inside the document collection.

The key component that we consider here, which is missing in all previous approaches, is the combination of fully structured information from knowledge bases and unstructured information from the underlying text collection in one single graph representation.

4.2.3 Graph-based entity ranking model

To better support an interactive browsing of the implicit network and the underlying document collection, it is necessary to slightly adapt the ranking functions for nodes in the network as introduced in Chapter 3.2. In the following, we motivate and introduce these updated ranking functions that we use to retrieve information from the graph.

Recall the definition of an implicit network structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with edges $\mathcal{E}$ and nodes $\mathcal{V} := T \cup E \cup D \cup S$ that are comprised of entities E, terms T, sentences S, and documents D. Also recall that $N(v)$ denotes the neighbourhood of a node v, that is, the set of all nodes in the graph that are connected to v. For two sets of nodes X and Y, edges between entities $x \in X$ and $y \in Y$ are then weighted by directed weights $\vec{\omega}$ that are derived from the textual distances of their mentions. For the entire derivation, see Chapter 3.2, but recall that the weight can be summarized as

$$\vec{\omega}(x, y) = \left(\log \frac{|Y|}{|N(x) \cap Y|}\right) \sum_{i \in I_{xy}} \exp(-\delta(x, y, i)), \tag{4.1}$$

where $\delta(x, y, i)$ denotes the distance in sentences between the occurrences of x and y in some cooccurrence instance i, and I_{xy} is the set of all such cooccurrence instances. Since these weights encode the directed importance of one entity for another, their relations can be used to rank nodes in the neighbourhood of one or several query nodes. In the following, we further refine these rankings depending on the type of target node.

We can represent the common core of all rankings for retrieving information from the graph as a query $\langle X|Q, n\rangle$, which consists of a target set $X \in \{T, S, D\}$ or $X \subseteq E$ (the latter for different types of entities), an integer n that specifies the number of nodes to retrieve from X, and a set of query entities $Q \subseteq \mathcal{V}$. To answer a query, the aim is then to order all entities in X based on some ranking function ϱ that measures the importance of their relations to query entities $q \in Q$ by using the network structure. The answer to a query is a set $X_n \subseteq X$ of the n top-ranked entities in X such that $\varrho(x_n) > \varrho(x)\ \forall x_n \in X_n, x \in X \setminus X_n$.

Ranking entities

A ranking of entities means that we equate the target set X with some subset of entities or terms. Thus, $X \subseteq E$ or $X = T$. For a subset of entities or terms as target set, we can distinguish between two different scenarios. First, if the set of query entities contains only a single entity q (that is, $|Q| = 1$), then we rank entities in X by the weights of edges starting at q in the graph, and let $\varrho_{\text{E}}(x) = \vec{\omega}(q, x)$. If q and x are not connected, then we simply set $\varrho_{\text{E}}(x) = 0$. This ranking directly retrieves the most important entities x in the neighbourhood of entity q and is identical to the entity ranking that we introduced in Chapter 3.2. To obtain a score in the interval $[0, 1]$, we normalize with the maximum observed score $\vec{\omega}_{max}$ to any entity in the result set,

$$\vec{\omega}_{max} := \max_{x \in N(q)} \vec{\omega}(q, x). \tag{4.2}$$

Second, if we have multiple query entities (that is, $|Q| > 1$), then we employ a two-tier ranking system, in which we decompose the ranking score into two components $\varrho_{\text{E}} = coh.sum$ such that coh denotes an integer and $0 \leq sum < 1$ (the dot thus denotes a decimal separator). In contrast to simply using the cohesion as a binary indicator as discussed in Chapter 3.2, this allows us to rank candidate neighbours first by the cohesion component coh and break ties according to the second component sum. Here, coh denotes the *cohesion* of the subgraph that is induced by the query entities. Formally, coh is the number of query entities that are connected to a target entity beyond the first, that is,

$$coh(x) = |N(x) \cap Q| - 1. \tag{4.3}$$

Thus, we have $coh \in \{0, \ldots, |Q| - 1\}$, with larger values indicating a greater connectedness of a target entity to the query entities. For the second component, we again use the normalized sum of edge weights

$$sum(x) = \frac{1}{sum_{max}} \sum_{q \in Q} \vec{\omega}(q, x). \tag{4.4}$$

where sum_{max} is the maximum obtained sum of scores such that $sum \in [0, 1]$. For the resulting ranking score in the case of multiple query entities, we can thus set $\varrho_{\text{E}} = coh + sum$. We observe that $\varrho_{\text{E}} \in [0, |Q|]$, where greater scores denote a greater importance. The score then effectively ranks entities in the target set by the cohesion as the main component, and breaks ties by the directed importance edge weights.

The case in which we have only a single entity query, and therefore $|Q| = 1$, then corresponds to the special case where $coh = 0$, as described above.

Ranking sentences

To rank sentences, we equate the target set with the set of sentences, meaning that $X = S$. As discussed in Chapter 3.2, obtaining a ranking for sentences is less direct than a ranking for entities, since the edge weights between entities and sentences are binary and there is no notion of weight for these edges as a result. Therefore, we employ a slightly different two-component ranking scheme $\varrho = coh.sum$ to improve upon the existing sentence ranking. The first component is identical to the case of entity ranking and denotes the cohesion. That is, coh here denotes the number of query entities that are contained in a sentence x. To obtain the second component, we consider the k most important terms for each query entity and assign to each sentence a score that indicates how many of these

terms it contains. Formally, let T_Q be the union of the k most important terms for all query entities in Q. Then we obtain *sum* as the fraction of important terms that are contained in the sentence

$$sum(x) = \frac{|N(x) \cap T_Q|}{|T_Q|}. \tag{4.5}$$

Similar to the case of entity rankings, the combined ranking score ϱ_S is then the sum of the individual scores *coh* and *sum*. Here, other sentence ranking schemes exist, in particular schemes that also normalize the results for the lengths of sentences. We address the construction and performance of these alternative rankings in more detail in Chapter 4.3 for the purpose of summarization.

Ranking documents

For documents, a ranking can be obtained in almost the same way as for sentences, but with a target set $X = D$. However, we observe that each sentence belongs to exactly one document. Thus, we easily arrive at a ranking of documents by computing a ranking of sentences according to the query entities, and then propagating the scores from the sentences to their respective documents. Using the same two component score as above, we define the cohesion of a document d as the maximum cohesion of any of its sentences

$$coh(d) = \max_{s \in d} coh(s). \tag{4.6}$$

For the second component, we sum over the contributions of all individual sentences and obtain

$$sum(d) = \sum_{s \in d} sum(s). \tag{4.7}$$

After normalizing with the maximum of all $sum(d)$ values, and combining $\varrho_D = coh + sum$, we again obtain a score $\varrho_D \in [0, |Q|]$. While this approach is intuitive, it is not straightforward to implement on the dense network structure due to the two-hop relation between entities or terms and documents. Thus, we subsequently consider a merging of document edges into sentence edges when discussing the application architecture.

Subgraph extraction

As a final extraction method that is valuable for an exploratory visualization tool, we also consider an implementation of subgraph extraction. With the aim of highlighting the immediate neighbourhood of a given set of query entities, we first include all query entities

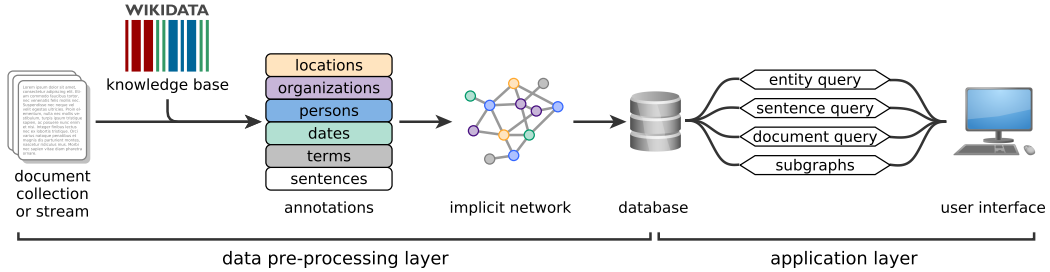

Figure 4.1: Schematic view of the data processing pipeline and system architecture of EVELIN. Here, we use Wikipedia as the input document collection and Wikidata as the knowledge base for entity linking, but the process can be applied to any document collection or document stream.

in the subgraph. To discover additional nodes, we rely on entity queries as defined above. Specifically, we rank entities in each of the entity type sets according to their importance for the query entities. Then, we select the three highest ranked entities in each set that have a cohesion of $coh \geq 1$ and include them in the graph. In a final step, we extract all edges between the selected nodes and include edge weights that can be used to visualize the importance of relations. In essence, this procedure equates to determining the highest ranked neighbours, and extracting the complete graph between all nodes that are selected in this way. Unfortunately, this extraction is comparably slow for dense local subnetworks, and may require a few seconds for highly connected nodes in the Wikipedia data. In Chapter 6, we consider a more efficient way of extracting descriptive subgraphs.

4.2.4 Application architecture

Based on the ranking functions discussed above, we now describe the data set that we use, and the system architecture for extracting the data and processing queries on the resulting implicit network of entities. For an overview of the system architecture, see Figure 4.1.

Data pre-processing

The extraction of an implicit network is possible from any document collection in which some type of entities can be identified. Here, in order to demonstrate the feasibility of the approach for large document collections and to provide comprehensive query choices, we again use the unstructured text of all English Wikipedia articles, from the dump of May 1, 2016. In contrast to the previous network in Chapter 3.4, we do not use automated named entity recognition but instead follow Wikipedia links to their Wikipedia pages, from which we can extract a Wikidata identifier. This largely avoids the imprecision that is inherent to

named entity recognition, since Wikipedia links are essentially manual annotations by the Wikipedia editors. One possible problem with this approach is the policy of only linking the first mention of an entity on a Wikipedia page, which we address by a subsequent string search of already mentioned surface forms of Wikipedia links and their Wikidata labels. We then link discovered entities to Wikidata identifiers, which disambiguates all mentions of any one entity to its common Wikidata entry. Using these identifiers, we then classify the Wikidata entities into the desired classes of location, organization, and actors. For the extraction and normalization of dates, we use HeidelTime [188]. We construct an implicit (LOAD) network from all discovered entities of types location, organization, actor, and date with a maximum window size of $c = 5$ sentences. The resulting graph is constructed from 4.5M Wikipedia articles with 43.6M sentences that have at least one entity, containing 2.0M named entities, 5.2M distinct terms, and 1.3B edges. In addition to the implicit network structure between the entities, terms, sentences, and documents, we now also have Wikidata information for all discovered entities. Specifically, we can use the Wikidata labels as unique entity labels, and retrieve short entity descriptions from the knowledge base to display entity information to the user.

Application layer

As discussed in Chapter 3.3, an in-memory representation of the implicit network of the entire English Wikipedia is possible and allows extremely fast queries in the order of milliseconds. However, due to the high memory requirements (around 200GB for full efficiency), this is infeasible for a long-running, non-commercial application. Conversely, query speeds in the order of few milliseconds are not necessary for an interactive user experience, where speeds of a few hundred milliseconds are sufficient. Therefore, we rely on a fully external storage architecture by storing the data in a MongoDB, with separate collections for entities, terms, sentences, documents, and edges. This architecture then runs on desktop-level hardware such as the Core i7 with 32GB main memory and an SSD drive that we use to run the Web demonstration. Since entities are enriched with Wikidata information to obtain entity descriptions and canonical labels, a text index on the English canonical label can be used for searching entities in the database by their label, and for compiling a list of entity suggestions to the user, based on input strings. An alternative solution is the use of prefix tries [94], which is faster than an index. However, it would have provided little improvement since entity retrieval by label is not a bottleneck even for millions of entities, and would have reduced the portability of the implementation. We thus rank entity suggestions by the text match score and break ties by the number of

connected sentences in the network (that is, by frequency). Graph edges are stored with precomputed directed importance weights $\vec{\omega}$. For edges that involve sentences, a collapsed storage format that contains both the sentence and the respective document of the entity or term allows faster retrieval speed at the cost of storing one additional integer per sentence edge. The query processing routines described above are implemented in Java, and enable query processing speeds in the order of a few seconds or less for all but the experimental subgraph queries. While individual queries are easy to parallelize by input entity for queries containing multiple query entities, we do not include a parallel implementation for single queries. Instead, to allow the application to serve queries from multiple users simultaneously, query processing allocates one thread per query per user. To avoid system overload in the case of multiple users that spam queries, an internal mapping of queries to browser fingerprints allows us to limit the number of active queries per user.

Presentation layer

The web interface of EVELIN is implemented via HTML and JavaScript, and serves two primary purposes: classifying input terms and entities according to their entity type, and visualizing the output of ranked entity lists and subgraphs. For handling entity input and sending queries to the application layer, we use jQuery and pass entity information in both directions as JSON objects. The Bootstrap library [33] and Mustache web templates [210] are used for the responsive layout and for displaying data tables. To recognize, classify and color input entities as they are entered, we use the tags-input and typeahead libraries of Bootstrap, which are extended to include the required functionality for color-coding nodes according to their (entity) type. The interactive visualization of subgraphs is handled by the D3 JavaScript library [34], which uses a combination of HTML, CSS, and SVG to display data. Graphs are visualized with a force-directed layout. The web server itself is realized on top of the Java Spark micro framework [213], and is directly integrated with the application layer into a single application. The Communication between the user interface and the server is built on AJAX and uses JSON for transmitting entity information in both directions, including input query entities, output entity rankings, and graph data. Due to the Bootstrap library, EVELIN is fully compatible with mobile devices. Examples of the interface for inputting query entities is shown in Figure 4.2.

4.2.5 Application usage scenarios

Due to the flexibility of the underlying implicit network representation, EVELIN as described above inherently supports various application scenarios for exploring the data. In

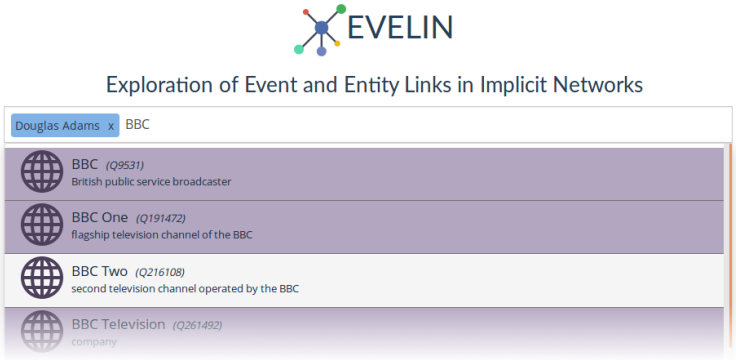

Figure 4.2: Query input interface for EVELIN. The query entity *Douglas Adams* is already entered and typed as a person, as indicated by the color. Upon typing *BBC*, the user is presented with a selection of entities that could be a match (in this case, all of them are organizations) that can be selected. Alternatively, any input can be entered as an untyped search term by not selecting a suggested entity.

the following, we describe these core features as they organically arise during the exploration of the network and documents by the user.

Entity exploration

To address the most fundamental task, beginning with a single input entity such as a person or location, EVELIN is capable of discovering related entities from the ranking of relevant nodes in the neighbourhood of the input entity. By adding such newly discovered entities to a query, or by removing less interesting entities, single-entity queries can be grown dynamically into the exploration of an event or an entity's timeline. Alternatively, input entities or sets of multiple input entities (for example, potential event candidates) can be described and understood through term recommendations provided by the system. In Figure 4.3 (top), we show the output of an entity query with the target type *actor*.

Entity summarization

Once an interesting entity combination is identified by the user, sentence rankings can be used to derive multi-entity summarizations from the implicit network. Based on any set of input entities, such as the one discovered during the entity exploration phase, EVELIN can retrieve a ranking of sentences from the entire corpus that best describes the selected entities as a composite. Thus, it generates extractive summaries of the selected entities, such as the example shown in Figure 4.3 (middle).

Figure 4.3: Query output of the EVELIN interface. Top: the result of a query of type ACT for the input entities *Douglas Adams* and *BBC*. All result entities are linked to their Wikidata entries for further exploration. The results of the three remaining entity types and terms have a similar design. Middle: the result of a query of type SEN, which ranks descriptive sentences. Bottom: a ranking of Wikipedia pages for the two given input entities.

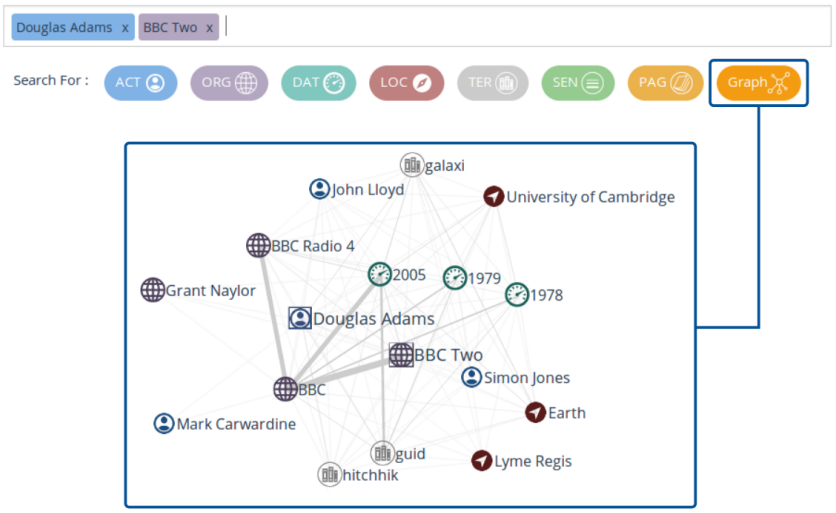

Figure 4.4: The result of a subgraph query for the two entities *Douglas Adams* and *BBC* in the EVELIN interface. The thickness of edges denotes edge weights, and node icons are selected according to the type of entity.

Entity linking

As a more coarse aggregation step that provides more context, the system can also be used to obtain provenance information for further studies. Effectively, this is achieved by linking individual entities or even a set of entities to documents from the underlying document collection that provide evidence for the entity cooccurrences. For example, consider the description of the relation between Douglas Adams and the BBC in Figure 4.3 (bottom). Here, it becomes apparent how individual entity occurrences and cooccurrences can serve to locate relevant mentions across multiple documents. Thus, these entities act as stitching points between the event descriptions spread across documents. We investigate this further in the analysis of dynamic implicit networks for news streams in Chapter 5.

Subgraph exploration

As an alternative to the ranking-based approaches above, we can also consider the extraction of subgraphs around entities. While this process is less precise than the entity-centric searches, a subgraph exploration can help the user to obtain a first impression of an entity's relations, and provide a starting point for subsequent searches. Thus, a subgraph extraction can serve as an augmentation and visualization of the search process where appropriate. An example of such a subgraph visualization is shown in Figure 4.4. While the subgraph extraction of EVELIN is limited in regard to its scalability due to the density

of local graph neighbourhoods, we consider a global ranking approach to the extraction of subgraph as descriptive network topics in Chapter 6.

4.2.6 Summary

Overall, EVELIN represents a novel and comprehensive way of searching for entity-related information in unstructured, large-scale document collections by leveraging the intrinsic relations of entities in the documents. Instead of relying on term-, entity-, or structural linguistic information alone, it combines all three data types organically in a single graph structure that can be used to navigate and explore the underlying document collection. As a result, we are able to retrieve entity rankings as well as descriptive sentences and documents, which enables us to induce and query a knowledge graph-like structure derived from otherwise unstructured document collections. For the purpose of demonstrating the functionality, we have relied on event-centric data in general and the related entity types location, organization, person, and date, in particular. However, as we recall from the general implicit network model, this is not a necessary restriction, and the model can apply in any setting with entity types. For example, EVELIN could be applied in a medical setting where entities might be symptoms, while sets of entities correspond to diseases. Ultimately, the restriction to certain entity types is not model-based and depends only on the data set and annotations that are used.

Despite this generality and agnosticism towards entity types, a central type of nodes in the network are sentences, since they are a structural element of the texts and not a part of the content. In the following, we therefore focus on the extraction of sentence nodes, which leads us to the task of summarization.

4.3 Entity-centric Summarization

For the retrieval of concise entity relation information from large collections or streams of documents, existing approaches can be grouped into the categories of (multi-document) summarization and knowledge extraction. The former tend to fall short for this task due to the involved amount of information that cannot be easily condensed, while knowledge extraction approaches are often too discriminative for exploratory purposes. As a result, knowledge bases are typically populated with only short relationship descriptors, such as *capital of* or *born in*, but lack complex relations between the contained entities. However, as we have seen in Chapter 3 and 4.2, implicit networks can be used to extract descriptive sentences for arbitrary sets of entities. In the following, we therefore investigate the

extraction of descriptive sentences for sets of entities from implicit networks to further support an interactive exploration. We introduce and compare efficient scoring functions for sentence extraction that address this entity-centric summarization task, and apply them to the extraction of complex entity relations. Based on the example of relations between locations that are neither hierarchical nor containment-based in nature, we demonstrate the extraction of novel location relations that are not contained in knowledge bases.

Attribution. This section describes joint work. The comparison between relations that are extracted from the implicit network and those that are contained in the knowledge base, was implemented by a collaborator.

4.3.1 Why entity-centric summarization?

The steadily growing amount of available textual information with entity-centric content has long passed the threshold for unassisted human analysis in many areas. Frequently, such an analysis is focused on certain aspects of the content that can be interpreted as a class of entities, such as person or location mentions, as well as their relations. In those cases, the exploration of new (or newly digitized) document collections inherently benefits from automated and interactive exploration tools that support an entity-centric focus. Consider, for example, journalistic approaches in which huge collections of financial or legal documents become available that need to be examined for relations between central actors to uncover fraud or corruption. It is therefore not surprising that automated summarization and knowledge extraction approaches are among the most popular topics in information extraction and retrieval. Given a set of documents, extractive summarizers are aimed at composing a comprehensive, readable, and short overview of the contained key points [142]. Entity-centric summarization techniques in particular enable the extraction of brief descriptions of entities or sequences of entities from a text [43]. However, in settings where the document collection is large, such approaches are problematic due to the sheer amount of information, which calls for the preparatory selection of relevant information with a focus on some aspect of the collection. Useful tools in this regard are searching and indexing approaches, external knowledge bases, or question answering systems [109]. However, such systems are not designed for the retrieval of succinct descriptions of aspects, entities, or relations within groups of entities in the collection. Furthermore, as we have discussed in Chapter 2.3, due to their predominantly pattern-based design, fact extraction approaches stand to miss information that is not well structured but potentially suitable to human understanding in an interactive exploration of the documents.

For the extraction of geographic mentions and location relations in particular, this results in an extremely limited scope of attributes and relations that can be retrieved from the texts, due to the fact that more complex relations beyond co-location and containment are often not expressed in simple patterns (or cannot even be expressed in such patterns). Consider, for example, trade relations between countries or cities. While it would be possible to observe a sentence that explicitly states such a relation (for example, *China is the largest trading partner of the European Union*), it is much more likely that this relation is never explicitly stated and can only be inferred from the frequent mention of individual trades. Similarly, the wealth of potential social interactions between people is hardly covered by the small subset of family and professional relations that are typically encoded in knowledge bases. How, for example, would one succinctly describe the relation between Ada Lovelace and Charles Babbage with two to three words in a way that does it justice?

For these types of relations, a pattern-based extraction is ill-suited. Significant cooccurrences of entity mentions in a specific context, on the other hand, can be helpful in determining pairs of related entities and in subsequently identifying sentences that describe their relation. As a result, such complex relations are largely missing from established knowledge bases like YAGO [124] and DBpedia [115], which are populated through pattern-based knowledge extraction, and even from the community edited Wikidata [206], which is increasingly populated through automated approaches as well [187]. Thus, even knowledge bases (and their subsequent applications) stand to benefit from the extraction of descriptive sentences for complex entity relations.

In the following, we formalize and then strive to satisfy this information need by addressing the task of *descriptive sentence extraction* for entities and entity relations. To this end, we build on the implicit network model, which enables us to efficiently extract relevant sentences for sets of entities, as we have seen in the previous section. Based on the intuitions and first approaches discussed there, we investigate more finely tuned scoring functions for sentence nodes, and how they can be used to extract descriptive and explanatory sentences that contain more than just simple relationship patterns. We evaluate our approach on the extractive summarization of descriptions for Wikipedia glossary entries, and the extraction of non-hierarchical location relations.

4.3.2 Related work

The existing work on descriptive sentence extraction is fairly limited, so we give a brief overview of related areas and the works that are conceptually the most similar. Notably, the task is also related to knowledge extraction for question answering, as discussed in

Chapter 2.3. We begin by discussing the basics of summarization, as well as question answering for geographic locations, which relates to our evaluation.

Document summarization

The summarization of entire documents (or sets of documents) has been well researched in the past, and several recent surveys exist on this matter [121, 142]. Graph-based approaches to coherence and composition are particularly successful, which were pioneered by contributions such as LexRank [52] and TextRank [130] that use graph centrality to identify relevant components of a summary. These methods have been continuously improved, and more recent additions include novel techniques such as topic signatures [5], vector embeddings [129], or word associations relative to a background corpus [80]. Overall, graph-based document summarization tends to focus on sentences and representations of topics as nodes. In contrast, by using implicit networks, we put the focus on entities instead of topics. As a result, we can retrieve and explore entity information from the network representation of large document collections on a scale for which traditional multi-document summarization approaches are ill-suited due to their runtime complexities.

Summarization and question answering for geographic IR

Since we focus on the relations between locations for part of our evaluation, some summarization and question answering (QA) approaches with a focus on geographic information can also be considered related. Text summarization and question answering for geographic concepts have first been extensively addressed in GeoCLEF [74] as part of the Cross Language Evaluation Forum (CLEF). Among the main tasks, the focus has been put on NLP-based geographic search. Thus, it established a hybrid approach between text summarization and QA, although neither the extraction of relationships between locations (or geographic entities in general) nor location summarization are a particular focus. Similarly, a recent work by Chen et al. [42] also focuses on geographic QA but does not address location summarization or relations between locations. In addition to geographic QA, the more recent NTCIR GeoTime track [72, 73] included a temporal dimension for determining answers to NLP-based geographic search queries. Different aspects of summaries for geographic IR have been analyzed and studied by Perea-Ortega et al. [147], who focus on sentences in a document containing geographic entities, but do not specifically study location summaries or extracted location relations. Thus, while there has been substantial research into geographic question answering, there is no previous work on the extraction of descriptive sentences for complex non-hierarchical relations between locations.

Descriptive sentence extraction

Finally, there are a number of more general approaches that focus on extracting sentences or tags as descriptions. Probably the most popular works on enriching location information are based on a method proposed by Rattenbury and Naaman [152], in which image tags in Flickr are used to derive semantically rich geospatial image and location descriptions. Similarly, Tardi et al. [198] outline an approach for identifying the characteristics of locations from tags that are associated with photographs. However, these frameworks neither utilize large text collections to further improve location descriptions, nor do they investigate descriptive relations between entities.

In contrast, some works in other domains focus on similar, entity-centric tasks and sentence extraction. Kim et al. consider the extraction and ranking of sentences based on their usefulness for understanding the reasons behind the sentiments that are expressed in a document for opinion summarization [108]. To this end, they rank sentences in opinionated texts based on their usefulness for the reader to better understand the reasons behind the sentiments. Biadsy et al. extract summaries targeted at the creation of person biographies by focusing on sentences that are biographical in nature [25]. They distinguish between sentences that are biographical in style and those that are not. Since we intend to describe entities in general, not just persons, such a focus would be too narrow in scope. Amitay and Paris summarize websites from external descriptions by looking at sentences that contain hyperlinks to these websites [11], which is conceptually similar to our description of entities, but they utilize patterns that are specific to hyperlink anchors that differ from entity mentions. The above approaches are focussed on a specific type of entity or concept (sentiments, persons, or links). In contrast, we consider a broader approach that is useful for entity-centric explanation in general.

An approach for the extraction of support sentences that is compatible with general entities is given by Blanco and Zaragoza for the extraction of support sentences. To this end, they identify descriptive sentences from a document collection that describe the entities contained in a textual query [27]. A downside of their solutions is the reliance on textual queries as input. Furthermore, they exclude sentences from the output that do not contain any of the input entities. However, since this is likely to occur whenever multiple entities are used as input, this renders their algorithms incompatible with sets of query entities.

To follow a more general approach, we use the entity-centric representation of implicit networks that enables us to rank sentences for multiple input entities or terms, and to extract descriptive sentences for the relations between these entities.

4.3.3 Network-based sentence retrieval

We begin by giving an intuitive definition of entity-centric sentence extraction, before formalizing the extraction operations based on the implicit network model. For the construction of the implicit network, we consider the same components as in Chapter 3.2.

Descriptive sentence extraction

To introduce the extraction tasks, we begin with the special case of single-entity sentence extraction, which we then extend to the more general multi-entity sentence extraction.

Single-entity sentence extraction. Given a collection of documents D that consist of sentences S, a set of entities E, and a query entity $q \in E$, we let

$$S_q := \{s \in S \mid q \in s\} \tag{4.8}$$

be the set of sentences in which entity q is mentioned. The aim is then to identify the sentence $s_q \in S_q$ that generally best describes q. Extending this notion, we can also consider the summarization of relationships between entities by focusing on sentences that jointly describe the occurrences of a set of entities.

Multi-entity sentence extraction. Given a collection of documents D, sentences S, entities E, and a subset of query entities $Q \subseteq E$, we let

$$S_Q := \bigcup_{q \in Q} \{s \in S \mid q \in s\} \tag{4.9}$$

be the set of sentences in which at least one of the entities is mentioned. A descriptive sentence for the set of entities Q then is the sentence $s_q \in S_Q$ that best describes the joint occurrences of entities from Q in D. From this definition, it is clear that single-entity sentence extraction is a special case of multi-entity sentence extraction for $|Q| = 1$. In the following, we thus focus on the more general task.

Implicit network construction

For the underlying graph representation, we follow the definitions in Chapter 3.2. That is, we use a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which the set of nodes corresponds to $\mathcal{V} := T \cup E \cup D \cup S$. The set of edges $\mathcal{E}$ again represents the containment or cooccurrence of terms or entities in sentences, and each of these edges is weighted with asymmetric weights $\vec{\omega}$

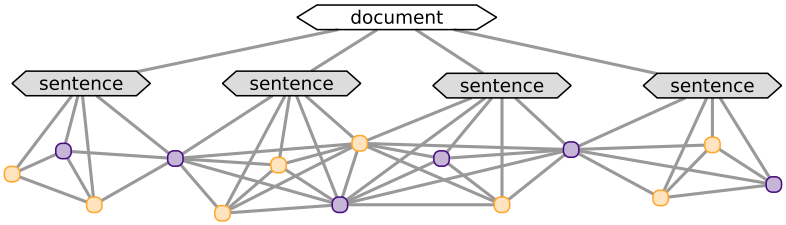

Figure 4.5: Schematic view of an implicit network for one document with four sentences. Entities (purple) and terms (yellow) as the main components of sentences are highlighted.

that we can use as input for the scoring functions. A schematic view of the graph that visualizes the sentence-centric focus is shown in Figure 4.5. Finally, recall that we denote the neighbourhood of a node v in the graph with $N(v)$.

Scoring functions for sentences

Based on the implicit entity network, we now introduce realizations of sentence extraction methods. We treat the task as a sentence ranking problem, in which we rank sentences according to their relevance for a set of input query entities $Q \subseteq E$, and then select the top-ranked sentence(s). Formally, we use scoring functions $\varrho : S \rightarrow \mathbb{R}$ that allow a ranking of sentences in the collection by their descriptiveness for the input entities. The answer to a query then is a sentence $s \in S$ such that $\varrho(s) \geq \varrho(s') \; \forall s' \in S \setminus \{s\}$. In the following, we describe four different scoring methods in a sequential manner. Thus, with the exception of the last method, each subsequent method includes and builds upon the previous methods.

Entity count (ENCO). As a baseline ϱ_{ENCO}, we use the method that we proposed in Chapter 3.2 and employed for initial exploration there. Recall that it counts the number of entities from the query set that occur in a sentence. Using set notation for the neighbourhoods, we obtain the score of a sentence s as

$$\varrho_{\text{ENCO}}(s, Q) := |N(s) \cap Q|. \tag{4.10}$$

For descriptive sentence extraction, this method can serve as a solid baseline, but suffers from two shortcomings. First, it performs poorly in the extraction of descriptive sentences for single query entities, since it assigns equal score to all sentences that contain the entity. Second, it does not consider the context of a sentence beyond the contained entities. Thus, ties between sentences with the same number of entities cannot be broken, which is especially of interest if no sentence exists that contains all query entities Q.

Term influence (TERI). Based on the above observations, we suggest an improved two-component scoring function that corresponds to the sentence ranking that we used in our implementation of EVELIN in Chapter 4.2. The number of entities in the sentence is kept as the first component, while we derive the second component from the set of terms that are most relevant to the query entities. To this end, we consider a ranking of terms in the neighbourhood of a query entity q by the directed edge importance $\vec{\omega}$ of the connecting edge, and let t_n be the n-th ranked such term. Let

$$T_n(q) := \{t \in T \mid \vec{\omega}(t|q) \geq \vec{\omega}(t_n|q)\}, \tag{4.11}$$

then the set $T_n(q)$ contains the n top-ranked terms in the graph with regard to q. From this, we obtain the most relevant terms for a set of query entities Q as

$$T_n(Q) := \bigcup_{q \in Q} T_n(q). \tag{4.12}$$

We then use these terms to represent the context of query entities and act as possible placeholders for query entities in a sentence that does not contain all query entities directly. That is, we let the context of entities act as their placeholder for ranking the sentence. We still rank by the number of query entities in the sentence first, but break ties by using the n most relevant terms for each entity. Formally, we let

$$\varrho_{\text{TERI}}(s, Q, n) := |N(s) \cap Q| + \frac{|N(s) \cap T_n(Q)|}{|T_n(Q)| + 1}. \tag{4.13}$$

Since the first component is an integer and the second component is strictly less than 1, we obtain the score based on the two-component intuition discussed above.

We find that identifying and using such relevant terms in addition to the query entities works well for sentence selection, but suffers from sentence length. While short and compact sentences are preferable descriptions in practice, both ϱ_{ENCO} and ϱ_{TERI} assign a higher weight to sentences that contain more entities (and terms), and thus favour longer sentences. In the following, we consider two normalization schemes.

Normalization by length (NORL). One possible way of normalizing with the length of a sentence s is to directly use the length in characters $len(s)$. Thus, we introduce the normalized score ϱ_{NORL} as

$$\varrho_{\text{NORL}}(s, Q, n) := \frac{1}{\log l(s)} \left[|N(s) \cap Q| + \frac{|N(s) \cap T_n(Q)|}{|T_n(Q)| + 1} \right]. \tag{4.14}$$

The addition of 1 in the denominator is due to the border case of sentences that contain no terms. Since we found in our empirical evaluation that we would otherwise give preference to extremely short sentence fragments that contain little more than the entity itself, we use the logarithm of the length (an alternative would be the length of a sentence in words). While this scheme normalizes based on the length of a sentence, it does not account for the number of entities and terms in the sentence overall and does not distinguish between terms and entities in the sentence.

Normalization by count (NORC). As a final method, we thus include a two-factor normalization that is based on the number of entities and terms per sentence. We define ϱ_{NORC} as

$$\varrho_{\text{NORC}}(s, Q, n) := \frac{|N(s) \cap Q|}{|N(s) \cap E|} + \frac{|N(s) \cap T_n(Q)|}{|T_n(Q)| \cdot (|N(s) \cap T| + 1)} \tag{4.15}$$

by normalizing the two components of the scoring function separately. The resulting function effectively measures the fraction of relevant entities and relevant terms that occur in a sentence. The factor $|T_n(Q)|$ in the second term also ensures that the contribution of entities to the final score is larger than the contribution of relevant terms.

In the following, we utilize and compare these four methods for the description of entity relations in general and relations between geographic locations in particular.

4.3.4 Sentence extraction evaluation

Based on these four scoring functions, we can now move to the extraction of descriptive sentences and its evaluation. We begin by discussing the construction of the large implicit network of entities on which we run the evaluation queries, and the extraction of the ground truth data.

Evaluation data and network extraction

The evaluation performance of entity-centric approaches is always influenced by the quality of entity annotations in the available data. Since we rely on both the recognition and disambiguation of named entities for a large document collection, manual annotations are prohibitively expensive. Thus, we again use Wikipedia as an evaluation resource, which allows us to recognize entity mentions due to embedded links between pages and disambiguate them through connections to the underlying knowledge base Wikidata. However, in contrast to the Wikipedia network used in Chapter 4.2, we do not limited the set of entities to a subset, but construct the network from all available entities.

Entity network extraction. We use the English Wikipedia dump of December 1, 2016, as a document collection, and restrict the content to the unstructured text (that is, we exclude lists, references, and infoboxes). As an entity, we consider any surface string that covers an embedded link to another Wikipedia page. Thus, we use embedded links to identify entities and directly link them to Wikidata identifiers. According to Wikipedia rules, entities are linked only once per page, so we use a string search of cover texts and Wikidata entity labels to tag subsequent mentions. We exclude all links that have no associated Wikidata identifier (that is, links that lead to Wikipedia pages with no associated Wikidata entry). To generate the network and exclude word fragments, we split the documents into sentences that are then tokenized. We restrict the terms to a minimum length of 4 characters before stemming and removing stop words. The resulting implicit network then contains 4.9M documents with 53.2M sentences, 3.6M entities, 5.8M terms, and 2.8B edges.

Query response time. Similar to the architecture used for EVELIN, we store the network in a MongoDB instance. Since the network data has a massive size of 400GB in JSON format, supporting efficient query processing is a valid concern. However, despite the size of the data, we achieve average query response times in the order of seconds when using such a secondary storage architecture. As we have discussed and shown in Chapter 3.3 and 3.4, this could be further reduced to the order of milliseconds when using an optimized, in-memory representation with collapsed edge attributes. However, for the task of evaluation, the performance is entirely sufficient.

Ground truth data. We evaluate the performance of all four scoring functions on single-entity sentence extraction since data from Wikipedia is available that can be used to evaluate this task. While an additional multi-entity evaluation would be beneficial, we are unaware of any labelled data that is suitable for such an evaluation. To obtain single-sentence descriptions of a variety of entities, we use the Wikipedia glossary pages for astronomy [215], biology [216], chemistry [217], and geology [218]. All four pages have a list structure that consists of items denoting the entity, and brief explanations that can be automatically extracted. For some examples from the glossary page of geology, see Table 4.1. To link the entities to nodes in the network, we rely on embedded Wikipedia links in the same way that we used during the network construction. We extract all items from these lists that have an associated Wikidata identifier and use only single-sentence descriptions for the evaluation. The sizes of the resulting evaluation sets are given in Table 4.2.

It is important to note that the glossaries are formatted as lists and thus excluded during the extraction of the network. Therefore, the exact sentences that we use as ground truth do not occur in the collection of documents that we use for the network construction, and we evaluate the extraction of descriptive sentences, not the retrieval of exact matches.

entity	Wikidata	description
archipelago	Q33837	a chain or cluster of islands
mineralization	Q6864409	hydrothermal deposition of economically important metals in the formation of ore bodies or lodes
tectonics	Q193343	large-scale processes affecting the structure of the Earth's crust

Table 4.1: Example of ground truth entities with Wikidata identifiers and descriptions from the Wikipedia glossary of geology.

Evaluation

Before discussing the results, we briefly describe the evaluation setup and introduce the employed evaluation metric.

Evaluation metric. The evaluation of extractive summarization and single-sentence descriptions is a notoriously difficult endeavour due to the lack of suitable measures with a genuine semantic comparison for short texts. However, since we are only interested in the relative performance of the four methods, this problem is less severe. To obtain some measure of comparison between the four ranking methods, we thus employ the standard evaluation metric ROUGE [120]. We use the RxNLP Java implementation [64] for our evaluation, with enabled stemming and stop word removal. Due to the limited size of sentences in comparison to summaries, higher-order n-grams do not occur with meaningful frequency, and we thus focus on ROUGE-1 as a measure of performance.

Evaluation setup. For each entity in each of the four ground truth data sets, we identify the corresponding node in the implicit entity network by matching the Wikidata identifiers. We then perform a ranking of all sentences that contain the entity and extract the top-ranked sentence according to each of the four methods. We compute the ROUGE-1 scores and calculate the macro-averages for each of the four data sets, as well as for the entire ground truth.

Evaluation results. In Table 4.2, we show the average ROUGE-1 precision, recall, and F_1-scores of the ranking methods ENCO, TERI, NORL, and NORC on the four evaluation sets and on the combined set of all entities. We use the $n = 5$ most relevant terms for selecting the best context of entities in a sentence for the methods TERI, NORL, and NORC. The overall performance is expectedly low due to the strictness of the evaluation, but we find several clear patterns. The scoring by term influence consistently results in the best recall across all data sets, which we attribute to the fact that it is lacking a normalization by sentence length. As a result, TERI favours longer sentences that are thus more likely

set	#entities	ENCO			TERI		
		prec	rec	F_1	prec	rec	F_1
astronomy	18	0.069	0.207	0.099	0.064	**0.248**	0.096
biology	167	0.086	0.181	0.105	0.075	**0.302**	0.106
chemistry	177	0.039	0.180	0.062	0.044	**0.316**	0.074
geology	225	0.053	0.144	0.072	0.061	**0.215**	0.090
all	587	0.059	0.167	0.079	0.060	**0.271**	0.090

set	#entities	NORL			NORC		
		prec	rec	F_1	prec	rec	F_1
astronomy	18	0.078	0.184	0.097	**0.084**	0.199	**0.109**
biology	167	**0.212**	0.133	0.127	0.160	0.179	**0.151**
chemistry	177	0.082	0.149	0.093	**0.084**	0.187	**0.107**
geology	225	**0.114**	0.129	0.100	0.105	0.150	**0.111**
all	587	**0.131**	0.138	0.105	0.113	0.171	**0.121**

Table 4.2: ROUGE-1 precision, recall, and F_1-scores of all four sentence ranking methods: entity count (ENCO), term influence (TERI), normalization by length (NORL), and normalization by count (NORC). We use a relevant term count of $n = 5$ for all evaluation sets. The best values for each metric and set are highlighted in bold.

to contain the key terms from the evaluation descriptions. The best precision is split between the two methods with normalization, NORL and NORC, depending on the data set. However, the difference in performance is fairly small in cases where the normalization by entity count performs better, and more pronounced in cases where the normalization by sentence length performs better, which indicates that NORL has a slightly higher performance with regard to recall for these data sets. Using the F_1-score as an overall measure, both normalized scoring methods consistently outperform the other two methods without normalization, with NORC performing best overall, as is evident from its superior recall in comparison to NORL. Despite the difficulty of the task, the relative gain in performance that is achieved by a normalization by length is noteworthy, as we find a 33% performance increase for a normalization by entity count over the non-normalized version of the measure (TERI). Given that Wikipedia contains some extremely long sentences, and that overly long sentences may occur in our data due to errors in the sentence splitting step, using such a normalization is clearly favourable to obtain readable results.

To analyze the difference in performance between the four methods more closely, we also consider precision, recall, and F_1-score as we vary the number of relevant terms n (see Figure 4.6). Note that the performance of the entity count baseline ENCO is constant as it

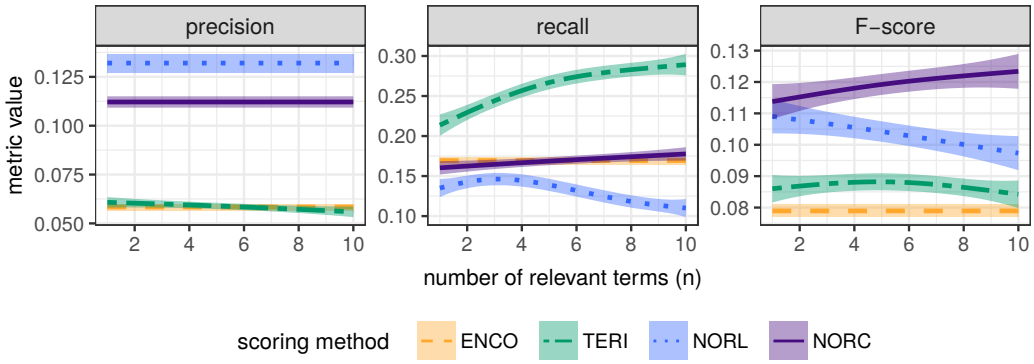

Figure 4.6: Average ROUGE-1 performances of the four scoring methods for sentences. Shown are the methods entity count (ENCO), term influence (TERI), normalization by length (NORL) and normalization by count (NORC) for varying relevant term counts n. Shaded areas denote 0.95 confidence intervals.

does not account for the occurrence of relevant terms in sentences. While the normalized scoring methods initially have similar F_1-scores for low values of n, NORC benefits more from using additional relevant terms. For recall, the performance of the term influence scoring method visibly exceeds all other methods by a large margin, which we attribute to the limitation of the sentence lengths by the methods that incorporate normalization. In contrast, the precision of the normalized methods is higher since the shorter sentences contain less noise. We find that increasing the number of relevant terms has no visible effect on the precision of NORL and NORC, and even decreases the precision of TERI. For recall, on the other hand, term influence scoring and the scoring that is normalized by count benefit from additional relevant terms, while the performance of the scoring that is normalized by length decreases. As a result, we find that NORC is best suited to for obtaining, descriptive sentences of moderate length, and that it benefits the most from adding context through additional relevant terms. However, the trade-off between the methods can be used to select an appropriate scoring function in practical applications that is tailored to the preferred result.

4.3.5 Extracting descriptions of location relations

Having evaluated the sentence ranking methods, we now apply the findings to the extraction and description of novel relations between entities in comparison to the typical relations in knowledge bases. Specifically, we investigate relations between pairs of locations that are significant in the implicit network of Wikipedia, but not contained in

Wikidata. Before we evaluate the coverage of discovered location relations with regard to the knowledge base, we first provide exploratory results as described in the following.

Exploration of location relations

The focus on entities of the type location requires an adjustment of the used data set, which we describe first before presenting and discussing the exploratory results.

Data preparation. As data set for this investigation, we utilize the same implicit network of entities from Wikipedia, but also consider a subset of entities of the type location. Since the classification of entities in Wikidata can be difficult due to the ever-changing hierarchies as discussed in Chapter 2.4, we instead use YAGO classes. To this end, we first merge the YAGO class hierarchy to the set of entities by matching the corresponding Wikipedia page URLs of items in both knowledge bases, and subsequently use it to assign entities to classes. For locations in particular, we utilize all entities that either have the class `yagoGeoEntity`, or are located in its subtree. The resulting data set has 242.8K entities of type location.

Exploration results. In Table 4.3, we show examples of descriptive sentences for pairs of European cities that we extracted with ranking method NORC as the overall best performing method. Since we do not aim to recreate relations that are already available in knowledge bases, we restrict these examples to pairs of cities that are not connected by a relation in Wikidata. For each pair of cities, we show only the two highest ranked sentences (in the case of ties, we randomly selected two sentences among the top-ranked sentences). We find that the sentences describe interesting relations between the cities that would be difficult to qualify as discrete relations in a knowledge base, but are well captured by descriptive sentences. For example, we find a variety of cultural relations, such as the importance of Berlin and Vienna for operetta, or Rome and Milan as centres of Italian fashion. The examples also include hierarchical similarities, such as the city states Hamburg and Berlin (note, however, that the sentence as contained in Wikipedia is not entirely correct since Bremen is commonly considered to be the third city state of Germany). Interestingly, the approach also identifies historic relations that are unlikely to be covered in contemporary knowledge bases, such as the relations between ancient Greek city states in the Peloponnesian War.

A common occurrence is the extraction of sentences that pertain to persons moving between two cities (for example, the second-ranked sentence for Berlin and Vienna). These sentences are artefacts of the predominance of biographical data in Wikipedia, which puts

Berlin (Q64)	Hamburg (Q1055)
1.	Berlin being the largest and Hamburg being the second largest city in Germany, they are also German states in their own right, having made both Wowereit and von Beust also state premiers.
2.	Two cities in Germany, namely Berlin and Hamburg, are considered city-states (German: “Stadtstaaten”).

Berlin (Q64)	Vienna (Q1741)
1.	In the same way that Vienna was the center of Austrian operetta, Berlin was the center of German operetta.
2.	Robert Bodanzky, also known as Danton (born 18 March 1879 in Vienna, Austria-Hungary as Isidor Bodanskie, died 2 November 1923 in Berlin, Germany), was an Austrian journalist, playwright, poet and artist.

Athens (Q1524)	Sparta (Q5690)
1.	Although Thebes had traditionally been antagonistic to whichever state led the Greek world, siding with the Persians when they invaded against the Athenian-Spartan alliance, siding with Sparta when Athens seemed omnipotent, and famously derailing the Spartan invasion of Persia by Agesilaus.
2.	The Greek historian Thucydides wrote in his History of the Peloponnesian War of how, in 416 BC, Athens attacked Milos for refusing to submit tribute and refusing to join Athens’ alliance against Sparta.

Athens (Q1524)	Corinth (Q103011)
1.	During the first centuries of the city’s existence, imported Greek articles predominated: pottery (see Kerch Style), terracottas, and metal objects, probably from workshops in Rhodes, Corinth, Samos, and Athens.
2.	In the wake of this battle, Athens, Thebes, Corinth, and Argos joined together to form an anti-Spartan alliance, with its forces commanded by a council at Corinth.

Rome (Q220)	Milan (Q490)
1.	It was set up in 1958 in Rome and now is settled in Milan and represents all the highest cultural values of Italian Fashion.
2.	Italian fashion is dominated by Milan, Rome, and to a lesser extent, Florence, with the former two being included in the top 30 fashion capitals of the world.

Table 4.3: Example of descriptive sentences for pairs of locations that have no connecting relation in Wikidata, extracted from the text of the English Wikipedia. Shown are the two highest ranked sentences for five pairs of European cities.

an emphasis on terms that are used in the descriptions of the births, deaths, and movements of persons in relation to their locations of residence. We discuss possible avenues to addressing or even utilizing this phenomenon after presenting the evaluation results.

Evaluation of relation extraction coverage

To substantiate our claim that we can extract descriptive sentences for novel relations that are outside of the scope of knowledge bases, we have to investigate the performance of the implicit entity network in identifying and ranking such relations. To this end, we extract ranked lists of locations in the network neighbourhood of given input locations by applying entity ranking methods. We then compare the ranked list of related locations for each input location to its knowledge base relations.

Evaluation data. To obtain evaluation data for this task, we again turn to Wikipedia lists, which we annotate through embedded links as described above. For this evaluation, we use lists of locations as seeds, for which we then rank adjacent locations in the network to identify relations that can be matched against Wikidata. Specifically, we consider the list of largest German cities [219] (79 locations) and the list of international capitals [220] (250 locations) as input locations. We extract each of the mentioned cities along with its Wikidata identifier and map it to the corresponding node in the implicit network.

Evaluation setup. Since our aim is a comparison of related locations in the implicit network to related locations in the knowledge base, we require a ranking function that extracts and ranks such locations from the network for a given input location. Here, we use the two-tiered ranking approach that we introduced for entity ranking in EVELIN in Chapter 4.2, and apply it to entities of type location. We also restrict the set of output entities to the type location to obtain a ranked list of locations that share a contextual relation with the input location.

Based on these rankings, we then perform the comparison to Wikidata relations. That is, we label each extracted location in the ranked list as *positive* if the corresponding location is linked to the input location in Wikidata, and we label it as *negative* if no such link exists. From these labels, we then compute precision and recall values for different positions in the ranked list that indicate how similar or dissimilar the top-ranked items of the list are in comparison to the content of the knowledge base. Thus, a method with high precision and recall would identify exactly the same relations as those that are contained in the knowledge base, whereas a method with lower precision and recall would identify different relations. For the following evaluation, we limit the set of relations to pairs of locations that are mentioned at least once in a common context in the documents.

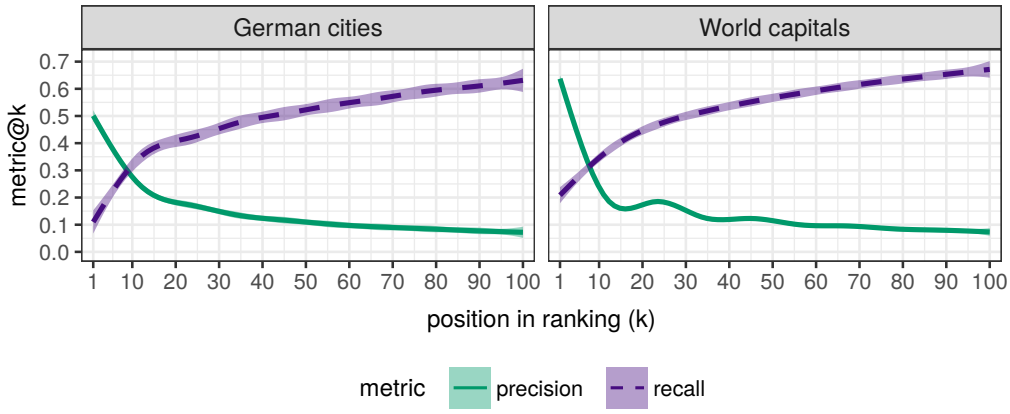

Figure 4.7: Average precision@k and recall@k for the ranking of location relations extracted from the implicit Wikipedia network in comparison to the location relations stored in Wikidata. Shaded areas denote 0.95 confidence intervals.

Evaluation results. We show the resulting precision and recall for the two evaluation data sets in Figure 4.7. In both cases, we find that the top-ranked locations from the implicit network differ strongly from the contained relations in the knowledge base. While the initial precision for the top ranks in the output list (that is, low values of k) shows that 50% to 60% of the top-ranked relations from the implicit network also occur in the knowledge base, this rapidly drops to 10% at $k = 50$, thus indicating a different type of relation. However, the relations that are contained in the knowledge base are also contained in the ranked lists, but appear further down the list, as is evident from the recall values. Since the implicit network is based on textual cooccurrences of location mentions, this is a reasonable observation. Hierarchical and containment relations that are typically stored in knowledge bases are much less likely to be mentioned frequently.

As a result, we can conclude that the observed relations in the implicit network describe different, potentially more complex relations. Thus, location rankings that are extracted from implicit networks can be used to identify pairs of locations that share novel relations. These pairs of locations can thus serve as input for the extraction of descriptive sentences, which stands to increase the number of different relations that can be considered. Beyond the direct use of such discovered relations in an exploration scenario, they could also be useful in discovering new facts and extending the coverage of knowledge bases.

4.3.6 Implications and Outlook

Two issues that we uncovered in our evaluation above deserve further investigation, which we discuss in the following.

Effects of the length of sentences

As we described in the evaluation of the sentence extraction of the four scoring functions, a normalization of the sentence scores by the length of the sentence directly influences the performance, regardless of whether this normalization is based on character or term counts. Specifically, a normalization by length increases the precision, but decreases the recall. While the overall benefit outweighs the drawbacks, as is evident from the obtained F_1-scores, such a restriction may not always be desirable. For example, the extraction of concise sentences is paramount for extractive summarizers, but such limitations may not apply to a composition technique for summarization that first extracts multiple sentences covering different aspects in relation to a set of focus entities and then combines them. As a result, a less restrictive extraction may be favourable at times, depending on the application. Thus, the selection of the proper sentence ranking method should be based on the given task at hand and on subsequent uses of the extracted sentences.

Interplay between entity types

In our extraction of example sentences for complex location relations, we found that a substantial number of location relations pertains to the movement of people between places of residence. While we see this largely as an artefact of the amount of biographical data that is stored in Wikipedia and our focus on cities as examples, our previous experience with EVELIN suggests that this can be exploited with the inclusion of additional entity types. The most direct approach would be to exclude or limit sentences that mention person movements from the results, where such movements can be identified. However, additional entities could also be used to focus on certain aspects of relations for a subset of locations. For example, the exchange of scientists between universities could be considered through the extraction of substantiating sentences that explain the transfers. The addition of dates as entities would then even allow an analysis of the flow of persons from one place to another, given an appropriate document collection. Such an inclusion of temporal data would be a step towards an analysis of the evolution of relations between locations in documents with spatio-temporal content.

Finally, we note that terms have so far played a minor role in the extraction of sentences, and only served to represent the context of entities indirectly, without being given a primary role in the graph structure. However, we further investigate the possibilities that arise from using descriptive terms as first class citizens in Chapter 6, where we use a similar intuition for the extraction of network topics around entities.

4.4 Summary and Discussion

In this chapter, we have taken a closer look at the capabilities of implicit networks for entity retrieval tasks. We have considered several ranking tasks in the context of an entity-centric search engine that supports the exploration of a static document collection, represented by an annotated dump of Wikipedia. Motivated by the efficiency of computing localized rankings in the neighbourhood of nodes, we have demonstrated that implicit networks can be used to support interactive and ad-hoc retrieval operations on large document collections, even if the data is stored on secondary storage. While we have approached the extraction of descriptive subgraphs for the visual display of the context around query nodes, these subgraphs are static and the efficiency of their extraction is limited in dense regions of the graph. In Chapter 6, we build on this concept to extract subgraphs to not only to describe static relations, but also to extract dynamic topics.

Based on the potential behind describing entities or sets of entities by extracting sentences, we then addressed the task of extractive summarization and focussed on sentence nodes in the network. We introduced further ranking functions to account for the length of sentences. Our empirical tests indicate that a normalization by sentence length improves the perceived ranking results due to the elimination of overly long sentences, as well as the overall quality of the top-ranked sentences. In addition to the extraction of descriptive sentences, there could be a future application in the multi-sentence summarization of entities, where a balance of recall and brevity stands to be investigated for an optimal coverage of contexts. Given the fast response times, such an extension could even enable near real-time document summarization techniques.

Practical implications

Consider the journalists and data analysts from our running example, and the possibilities that the methods discussed in this chapter open up for them. In contrast to the basic implicit network that is constructed from annotated but not disambiguated entities, an implicit network that is linked to a knowledge base provides new functionalities. In practice, this not only allows them to investigate the unstructured data in a more focused manner, but also provides additional information on entities from an external knowledge base such as Wikidata or even from internal knowledge repositories that are maintained within the investigative team. On top of this representation, an entity-centric search engine can then help in locating implicit entity relations comfortably, and visualizing them as graph structures. Through easily included links to the document repository and the knowledge base

used during the creation of the network, following up on information and obtaining further details on specific entities becomes a simple matter of identifying the corresponding nodes in the network.

However, a central aspect is not the retrieval of information about known entities, but the extraction of definitions and descriptions for previously unknown entities. Here, the investigators can directly benefit from an extractive summarization that provides them with descriptions of previously unknown entities from within the document collection itself. Furthermore, they can even retrieve descriptions of relations to give them a starting point for an investigation into the relations between two entities. Since these relations are derived from their cooccurrences across all document, they would be difficult or impossible to find in a manual analysis or by traditional search algorithms.

Outlook

Beyond the selection of applications that we presented here, implicit networks also have the potential to enhance further tasks in NLP or (social) network analysis. For example, the extraction of implicit social networks from document collections stands to be a useful tool in uncovering latent social relations [69]. Similarly, as we have demonstrated in our comparison to Wikidata, novel location relations beyond the typical hierarchical structures can be uncovered [70]. Networks of such latent relations could then also serve as semi-structured knowledge for the support of NLP tasks, similar to knowledge bases. There are already examples of applications in which such networks can improve the disambiguation of person mentions [68] or toponyms [178].

One limitation of the implicit networks that we have considered so far is their static nature. However, this is not an inherent feature, but an artefact of their construction from static document collections. In the following chapter, we therefore investigate the adaptation of implicit networks to a dynamic setting, in which they can be used to model not just document collections, but streams of documents.

5 Dynamic Implicit Entity Networks

In Chapter 3, we introduced implicit entity networks for the representation of large static document collections. In practice, however, many collections of documents are actually streams with a temporal dimension, such as news articles or blog posts. In this chapter, we therefore address how the implicit network model can be adapted to a streaming setting. In particular, since the context of entity occurrences and cooccurrences is more likely to change in new documents if a temporal dimension is considered, we also include new aggregation strategies for entity relations that go beyond a complete aggregation strategy like the one we used in Chapter 3.2.

Contributions. In this chapter, we thus make the following four contributions.

I We extend the implicit entity network model from static document collections to dynamic document streams, including publication dates as a temporal dimension.

II We include the context of entity mentions to enable a context-based aggregation of entity relations instead of a content-agnostic, complete aggregation, and propose a dynamic and a static partial aggregation approach.

III We evaluate the performance of dynamic implicit networks for information retrieval tasks on a large collection of entangled online news streams.

IV Based on the context of entity relations, we explore the extraction of dynamically evolving topics over time.

References. Parts of this chapter are based on the peer-reviewed publication:

A. Spitz and M. Gertz. "Exploring Entity-centric Networks in Entangled News Streams". In: *Proceedings of the 27th International Conference on World Wide Web (WWW), Companion Volume.* 2018, pp. 555–563. DOI: 10.1145/3184558.3188726

5.1 Motivation

Reading it in the paper in the morning is a common idiom for catching up with the news that is becoming increasingly less applicable. Putting aside the obvious departure from printed news, both the temporal aspect of *the morning* and the grammatical singular of *the paper* are less and less accurate. News are not reported and consumed in the morning, but in a constant news cycle throughout the day, published by a multitude of news outlets with varying degrees of reliability, political bias, and overlapping content. It is these *entangled streams of news* that the reader of news has to wade through to stay informed. As a result, the ever increasing number of news outlets and the frequency of the news cycle have made it all but impossible to obtain the full picture from online news. Consolidating news from different sources has thus become a necessity in online news processing. Despite similarities between the news cycle and streams of microblogs, and despite the abundance of research into extracting insights from online social networks, social media cannot take on the mantle of investigative journalism, which relies on argumentative texts and is less focused on the instant than it is on the evolution of stories. In this context, the so called *Five Ws* of *Who?*, *When?*, *Where?*, *What?*, and *Why?* are questions of central importance that serve the journalist and the reader in uncovering news. Naturally, these questions put an emphasis on entities as pivotal components of news. In information retrieval, this is reflected in the definition of an event as *something that happens at a given place and time between a group of actors* [6] that we have already applied to the modelling of events described in Wikipedia. Thus, it stands to reason that entities play a similarly (or even more) important role in inducing structure in the unstructured texts of news articles.

In contrast to Wikipedia, however, far more than one news article tends to be required to retrieve the full picture in large entangled news streams [192], thereby making it necessary to consolidate information snippets from different sources. On the other hand, a lot of information is replicated between or even within individual news streams and thus redundant. Intuitively, this motivates two major subtasks in automated news analysis: identifying event mentions in unstructured texts, and aggregating them across documents. These tasks are referred to as *new event detection* and *event tracking* [7], and can be augmented by detecting *topics* [28] that put individual documents into context. To make identified events accessible to users, a central step is thus their aggregation into threads of events along some dimension(s). Many different approaches have been proposed to this end. Some focus on a geographic aggregation and visualization of news sources [199], while others focus on the temporal aggregation [83], or both [229]. Alternative approaches use the participating entities directly [39, 111]. In the case of a temporal aggregation, different tem-

poral dimensions can be considered, such as the dates in the documents [135], or external information like the publication date [10] and edit histories [106]. With regard to timelines, another important aspect is then the temporal order, as the SemEval-2015 task for cross-document event ordering shows [134]. Beyond the above dimensions, more recent approaches include aggregation on a topic level [4] or based on word embeddings [138].

When we consider the above approaches for a contrastive exploration of the content of a news stream, we find that they suffer from two critical drawbacks: the limited number of *aggregation dimensions* and the *aggregation granularity* level. None of the approaches covers the entirety of available dimensions and it is indeed questionable whether an aggregation along all dimensions at once is realistically possible. Perhaps even more critically, the results are always coarse structures due to an aggregation either on the document, event, or topic level. However, if we consider incidents or events to be composite mentions of (named) entities, then they constitute the stitching points between individual news streams and can be used for a fine-grained consolidation. After all, we consume news about people, organizations, or locations of interest and follow them over time and in different contexts. Is it then not a more reasonable approach to retain this entity-centric structure of news in a suitable document model for subsequent analyses, and aggregate only where necessary and in exactly the dimensions that fit the exploratory task?

To address these shortcomings, we can adapt the implicit entity network model for document collections to a model for document streams. While we do so with the context of news streams as a primary application in mind, it should be easy to see how this approach is applicable to any other type of document stream that includes entity mentions, such as blog posts or even scientific publications. Although this adaptation is conceptually simple, it is not entirely straightforward. In particular, we have to include a temporal dimension beyond temporal expressions that are (potentially) included in the documents, and also consider document time stamps. Furthermore, taking into account the context of entity mentions is necessary to properly consolidate mentions from multiple sources and obtain a comprehensive framework for entity-centric analyses.

On the technical side, our model then serves to address the inherent scaling issues of multiple entangled news streams by utilizing efficient entity-centric queries to localized graph substructures. The streaming graph updates can take advantage of incremental adjustments to relevance measures for queries against the data [36, 223]. Furthermore, the implicit representation serves as an (inverse) index for retrieval tasks without requiring the storage of the proprietary content of news articles, which is an increasingly important aspect of news aggregation services.

On the application side, our model stands to provide a more fine-grained and versatile representation of entangled news streams than previous approaches, by relying on the entity-centric representation of implicit networks. Instead of utilizing document- or event-centric indexing, we focus on the level of entities and contexts, and use them as stitching points between individual news threads. The model then supports a wide range of tasks, including entity-centric topic and event extraction and tracking, contextual search, contrastive source comparison, and exploratory visualizations of the underlying streams, as we discuss towards the end of the chapter.

Structure. In Chapter 5.2, we discuss previous work that is related to the analysis of document streams with a focus on news, and give a brief background into the entity-centric exploration of such documents. Afterwards, we introduce the dynamic implicit network model in Chapter 5.3, and use it for the exploration of evolving contextual topics around the edges of such a network that we construct from news articles in Chapter 5.4. Finally, in Chapter 5.5, we evaluate the improved network model on a set of news events for the task of event completion.

5.2 Related Work

To the best of our knowledge, no previous work supports a comprehensive, entity-centric exploration of entangled news streams. However, there are some conceptually related approaches with a similar (albeit more limited) focus on entities as central components of news. In the following, we give a brief overview of these works.

5.2.1 Entity-centric exploration and analysis

A fundamental and historically important result in entity-centric document analysis with a strong emphasis on the detection of event descriptions is by Feng and Allan, who formalize the concepts of *incident threading* and *event threading* [59]. While event threading captures the internal structure of news topics by adding causal or temporal relations, incident threading merges mentions of identical entity cooccurrences. Later works utilize similar concepts. For example, Kanhabua et al. assess the importance of temporal expressions based on the cooccurrences of entities and temporal anchor texts within sentences [106]. Gupta et al. present EventMiner, a framework for extracting events from collections of documents [85] that is comprehensive in its use of temporal expressions and named entities, but does not scale to large document collections. Mishra and Berberich link coarse-grained

events from news articles to corresponding Wikipedia pages [135]. Similarly, Ceroni et al. use entity mentions and temporal information to confirm the occurrence of events in a document collection [39]. Although the above works focus on entities, they do not cover the breadth of entities that is inherent to news, and focus on static document collections.

5.2.2 Analysis of articles in news streams

In contrast to the entity-centric approaches, a number of well-known frameworks offer a comprehensive analysis of streaming news. Lydia is a large-scale aggregation tool for news articles [122] with numerous subsequent publications that build on the initial idea. The European Media Monitor builds and processes a repository of European news articles with a strong focus on the diversity of multilingual news content that is available in Europe [13]. News Stand monitors and retrieves RSS feeds to extract geographic content from articles for spatial clustering and visualization [199]. The enBlogue system allows the identification of emerging topics from news streams in real time [10], but has so far been applied with a focus on blogs and microblogs. A shortcoming of all the above approaches is the lacking support for a fine-grained and entity-centric analysis.

However, some more entity-centric approaches have been proposed for the analysis of news. Ahmed et al. combine topic modelling, clustering and named entity recognition to distinguish topics, storylines, and entities in streaming news articles [4], but do not include the effects of entity cooccurrences. To support an ad-hoc tracing of news streams, Vuurens et al. utilize the clustering and qualification of titles and sentences in news articles [207]. Moran et al. introduce the use of word embeddings to enhance first story detection in microblogs [138]. Yang et al. classify news documents into topics and measure topic novelty by using both keywords and named entities with relative weighting for event-level novelty detection [224]. For a similar purpose, Das Sarma et al. build entity dynamic relation graphs to identify entities participating in trending events, but exclude locations [160]. Many further approaches to streaming news analysis exist, but few of them consider temporal information, and none of them include temporal information at the content level, alongside entities, terms, and topics inside the documents as equally important components.

Thus, we focus on implicit networks as a more comprehensive representation that supports a multitude of subsequent analyses and dimensions of exploration. In contrast to the basic implicit network model from Chapter 3, we also include support for efficient streaming updates, an exploration along the dimension of publication dates, and the context of entities in this improved model. In the following, we thus describe how such a dynamic implicit network model can be realized in a streaming environment.

5.3 Stream Compatible and Context Sensitive Implicit Networks

Conceptually, we can approach the construction of a dynamic implicit network in much the same way as the static implicit network by deriving entity relations from joint entity mentions. However, a number of changes are necessary for an adaptation to a dynamic model that is compatible with streams of documents. In particular, we improve the static model by adding **(i)** term embeddings to encode the context of entity mentions, **(ii)** an adaptation to document streams by considering the document publication time as a distinct temporal dimension that is independent of temporal expressions in the documents, and **(iii)** an adaptation to streams of concurrent documents by using a multigraph model with (partial) edge aggregation schemes. We introduce these changes in detail in the following.

5.3.1 Entity multigraph model

Let D be a collection of documents or a document stream, such as news articles. As in the case of static implicit networks, each document $d \in D$ then consists of sentences $s \in d$. Recall that we denote the set of all sentences in all documents as $S := \bigcup_{d \in D} \{s \in d\}$, and that $\varsigma : S \to \mathbb{N}$ is a consecutive mapping of sentences to their index in the document in which they occur. Each sentence is then considered to be a collection of words, which we partition into terms and entities (or multiple entity classes) as needed.

Graph nodes

The set of nodes does not require any changes from the static model. That is, we can use the same approach as introduced in Chapter 3.2 and define the set of nodes $\mathcal{V}$ as the union $\mathcal{V} := T \cup E \cup D \cup S$ of documents D, sentences S, entities E, and terms T. Recall that the function $\eta : \mathcal{V} \to \{T, E, D, S\}$ then designates the node type $\eta(v)$ of a node $v \in \mathcal{V}$. Alternatively, the domain of η can be extended to include more entity types such as locations, dates, or persons if this is of interest for the modelled data.

Graph edges

In contrast to the nodes, the generation of edges needs to be approached in a different manner, since these capture the dynamics of the document stream. However, not all edges are affected in the same way. Similar to the static model, we construct the set of edges

$\mathcal{E} := \mathcal{E}_C \cup \mathcal{E}_P$ based on two criteria: *containment* and *proximity*. Containment represents edges $\mathcal{E}_C$ between entities and sets, such that an entity is connected to a set that contains the entity. This type of edge provides provenance and context information for entities, but is not affected by the dynamics of the stream. Proximity edges $\mathcal{E}_P$, however, encode the cooccurrence of entities within at least one common document and represent implicit entity relations. Edges of the proximity type introduce parallel edges in the graph since one edge is induced whenever two entities cooccur, and thus are influenced by the dynamics of the stream. To distinguish between parallel edges, we rely on *instances* $I \subseteq \mathbb{N}$ of entity cooccurrences, along with an injective mapping $\iota : \mathcal{V} \times \mathcal{V} \to I$ as introduced in Chapter 3.2. Recall that for cooccurrence instances, $i = \iota(v, w) \in I$ represents an instance of the cooccurrence of two entities v and w in some unique ordering, such that the tuple $e = (v, w, i)$ denotes an edge between v and w. For example, if entities v and w cooccur in a document, this induces an edge (v, w, i). If they later cooccur again, we obtain a new edge (v, w, j). Note that a document may contain multiple instances of the same entities and thus multiple cooccurrence instances. Formally, we obtain

$$\mathcal{E}_C := \{(v, w, i) \mid v \in w \wedge i = \iota(v, w)\} \tag{5.1}$$

$$\mathcal{E}_P := \{(v, w, i) \mid \exists d \in D : v, w \in d \wedge i = \iota(v, w)\} \tag{5.2}$$

Thus, containment edges occur between entities and sentences, or between sentences and documents, while proximity edges connect entities or terms and entities. Since there is a surjection from S to D, edges between entities and documents can again be reconstructed from edges between entities and sentences. The resulting graph is undirected and we therefore do not distinguish between edges (v, w, i) and (w, v, i) where this is clear from context. However, in contrast to the static model, the set of edges is now a multiset that contains parallel edges, meaning that $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a multigraph.

For the following description of the model, also recall that we use $N(v)$ to denote the neighbourhood of a node v, that is, it denotes the set of nodes that are connected to v by at least one edge in the graph.

5.3.2 Edge weights and edge attributes

To assign weights and attributes to the edges, we distinguish between edges of the containment type $\mathcal{E}_C$ and edges of the proximity type $\mathcal{E}_P$. Since the construction of the two types is conceptually different, we discuss the addition of attributes separately.

Containment edges

Edges of the containment type are binary relations at their core. Therefore, the resulting edges are essentially unweighted, although parallel edges may occur in rare cases such as multiple stop words occurring in the same sentence. To simplify the subsequent notation, we define the distance function $\delta : \mathcal{E}_c \to \mathbb{N}$ for edges between an entity v and a sentence s as $\delta(v, s, i) := 0$ if $v \in s$, and $\delta(v, s, i) := \infty$ if $v \notin s$. The distance between sentences and documents is defined analogously. Note that these distances work in the same way as the distances for the static implicit network.

Occurrence proximity edges

Edges of the proximity type are more complex due to more finely nuanced distances between entities and terms, and due to parallel edges caused by multiple cooccurrences. Since it is this entity cooccurrence information that encodes the relevant dynamic information that is necessary for later analyses, we want to preserve these multiple edges and enrich them with additional information prior to their aggregation (see Chapter 5.3.4). As edge attributes for occurrence proximity edges, we consider three fundamental concepts, namely **(i)** the publication time, **(ii)** the textual distance between the mentions of two entities, and **(iii)** the context of the mentions.

Publication time

For a stream of documents such as news articles, we can assume that a publication time or retrieval date is known for each document, which we use to derive an edge attribute.

Definition 5.1 (Publication time τ). Let $\tau : D \to \mathbb{N}$ be a function that maps each document $d \in D$ to its publication time $\tau(d)$. Let d_i be the document that contains a cooccurrence instance i inducing an edge $e = (v, w, i)$ between two nodes. Then $\tau(e) := \tau(d_i)$ is an attribute that assigns the publication time of the corresponding document to edge e.

Thus, if we observe multiple parallel edges between two specific nodes, each edge is assigned a time stamp that denotes its occurrence time in the stream.

Textual distance

The textual distance of proximity edges works similar to the static case. Formally, by overloading the function ς, we can map each entity of an instance to the index of the

sentence in which this entity occurs. Thus, let $\varsigma(v, i)$ denote the number of the sentence in which entity v occurs in instance i. For example, if entity v in instance i occurs in the first sentence of a document, then we have $\varsigma(v, i) = 1$. In analogy to Chapter 3.2, the textual sentence distance $\delta : \mathcal{E}_P \to \mathbb{N}$ of two entities can then be written as

$$\delta(v, w, i) := |\varsigma(v, i) - \varsigma(w, i)|. \tag{5.3}$$

For example, if entities v and w cooccur in a document such that v is contained in the first sentence, and w is contained in the fourth sentence, then $\delta(v, w, i) = 3$. Thus, if two entities occur in the same sentence, their distance is 0. If v and w never occur together in the same document, we set $\delta(v, w) := \infty$. To include the distance of entity cooccurrences in the graph, we assign to each edge $e = (v, w, i)$ the corresponding distance $\delta(v, w, i)$ as an edge attribute $\delta(e)$.

Context embeddings

To conserve the context of joint entity mentions and model the context in which an edge originally occurred, we use a vector embedding of terms in the context window of two entities. Formally, an embedding is a function $\varepsilon : T \to \mathbb{R}^k$ that maps a term to a point in a k-dimensional vector space (see Chapter 2.1). Without loss of generality, we assume that pre-trained embeddings are available for all terms (either trained on previous documents from the stream or on out-of-domain sources). To obtain the context of two entities in a cooccurrence instance, we have to consider the occurrences of terms around and between the entities or terms that correspond to the nodes of the edge. To this end, we first define a context window around a cooccurrence instance as a function of those entities.

Definition 5.2 (Context window *win*). Let $e = (v, w, i)$ be an edge. Then we define $win : \mathcal{E}_P \to 2^S$ as a function that maps an edge to a set of sentences. Specifically, let d_i be the document containing the edge-inducing instance i, then

$$win(v, w, i) := \{s \in S \mid s \in d_i \wedge \varsigma(v, i) \leq \varsigma(s) \leq \varsigma(w, i)\}, \tag{5.4}$$

where $\varsigma(v, i) \leq \varsigma(w, i)$ without loss of generality.

Thus, $win(v, w, i)$ consists of the sentences containing v and w, and all sentences in between. Based on this context window, we can define the context of an edge as the normalized sum of all embeddings of the terms in the context window.

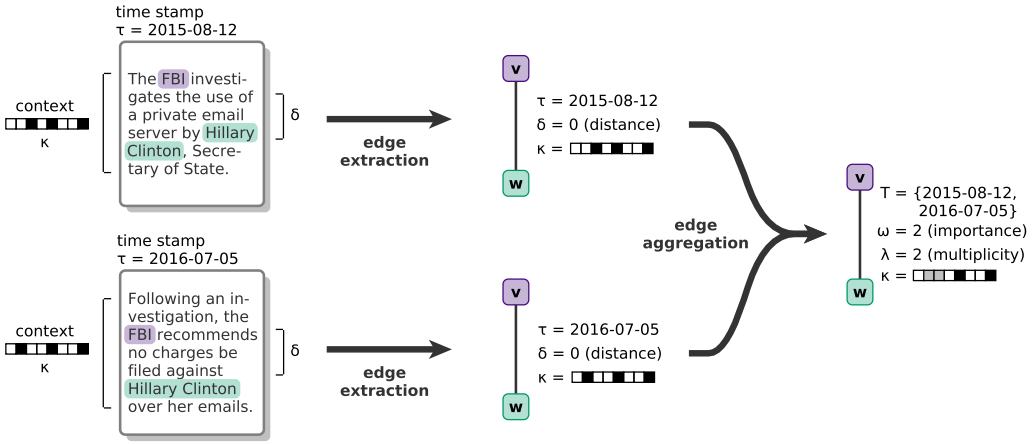

Figure 5.1: Schematic view of edge extraction and aggregation in the dynamic implicit network model. From an input document, edges between entities v and w are extracted with the vector-valued context embedding κ, the cooccurrence distance δ and the document timestamp τ as attributes. If edges between the same entities re-occur in a similar context, these edges are then merged and the attributes are combined to obtain the aggregated attributes.

Definition 5.3 (Context embedding κ). Let $e = (v, w, i)$ be an edge, and let $win(v, w, i)$ be the corresponding context window. Then the context function $\kappa : \mathcal{E}_P \to \mathbb{R}^k$ is defined as

$$\kappa(v, w, i) := \sum_{s \in win(v,w,i)} \sum_{t \in s} \frac{\varepsilon(t)}{|win(v, w, i)|}, \tag{5.5}$$

where $|win(v, w, i)|$ denotes the number of terms in the context window.

The removal of stop words and the restriction to content words is feasible in this step to reduce noise. For each edge $e = (v, w, i)$, we store $\kappa(e)$ as an attribute that can later be used to identify pairs of entities that appear in similar contexts. For a schematic overview of the model and the differences to the static approach, see Figure 5.1.

5.3.3 Aggregated graph attributes

Based on the edge attributes as introduced above, we now lay the foundations for an entity-centric exploration of document streams that is sensitive to the context of entity relations. A shortcoming of the static implicit network model is the aggregation of all parallel edges to obtain a simple graph. While such an aggregation makes the extraction of graph representations from large document collections feasible, it does not distinguish between mentions in different contexts. In many streaming applications such as news, however, the

number of contexts in which two entities cooccur is likely much more limited than in a comprehensive dictionary like Wikipedia. Thus, an aggregation of edges by context still results in a stark reduction of the number of edges, while also preserving the context of entity cooccurrences for later analyses. Here, it seems reasonable to design such an approach to be flexible enough to handle arbitrary and varying numbers of contexts for a given edge. Furthermore, an aggregation by context is very likely to partially preserve the multiplicity of edges (albeit with reduced multiplicity), while simultaneously collapsing unjustifiably duplicate edges to enable a more focused extraction of relations from the resulting graph. In particular for entangled streams of news articles with redundant information, such an approach is clearly beneficial.

In the following, we thus discuss how we can obtain an aggregated graph $\mathcal{G}_A = (\mathcal{V}, \mathcal{A})$ from the original multigraph $\mathcal{G}$. While the set of nodes remains unchanged, we obviously require a new set of aggregated edges $\mathcal{A}$ with aggregated attributes. Let v and w denote two entities, and let I_a denote a set of instances that induce parallel edges $\mathcal{E}_a$ with

$$\mathcal{E}_a := \{(v, w, i) \in \mathcal{E} \mid i \in I_a\} \tag{5.6}$$

between the two nodes v and w. In the following, we consider the derivation of edge attributes when the set of parallel edges $\mathcal{E}_a$ is aggregated to a single edge a. We begin by deriving the aggregated edge attributes.

Aggregated edge importance

This weight derives an overall strength of the relation between two entities from the sentence distances of individual edges, and corresponds to the importance weight of the static implicit network that we introduced in Chapter 3.2. Thus, the dissimilarity of individual sentence distances is transformed into similarities by a decaying exponentiation. The individual similarities are then added over all aggregated edges in a process that corresponds to the edge weighting in the basic implicit network. Thus, we compute an importance weight for the aggregated edge $a = (v, w, j)$ as

$$\omega(a) := \sum_{e \in \mathcal{E}_a} \exp(-\delta(e)) \tag{5.7}$$

Aggregated publication dates

For a temporal analysis, we store the set of all publication dates, which we assume to be distinct, as long as the granularity of time is fine enough. For lower granularities, this

attribute is effectively a multiset of dates, since multiple documents with identical dates are likely to occur. Formally, we thus have

$$\mathcal{T}(a) := \bigcup_{e \in \mathcal{E}_a} \{\tau(e)\} \tag{5.8}$$

Aggregated context

The context is the primary component of the edge aggregation (as we discuss in Chapter 5.3.4). However, once edges are aggregated, a single context vector is sufficient to represent an edge and facilitate context-sensitive queries. For reasons of storage efficiency, it is also sensible to not store more vector-valued edge attributes than necessary. Therefore, the contexts of individual edges is aggregated as the mean of the context vectors. Since the context of two entities whose mentions are separated by a couple of sentences is likely less important than two mentions within the same sentence, we normalize individual contributions by the distance of the mentions δ.

Definition 5.4 (Aggregated context embedding κ). Let $\mathcal{E}_a$ denote a set of parallel edges that are to be aggregated into a single aggregated edge a. Then the context for the aggregated edge is defined as

$$\kappa(a) := \frac{1}{|\mathcal{E}_a|} \sum_{e \in \mathcal{E}_a} \frac{\kappa(e)}{\delta(e) + 1}, \tag{5.9}$$

where the addition of 1 prevents a division by zero for intra-sentence occurrences.

Multiplicity of aggregated edges

To maintain the context centroid in the streaming aggregation model, we also have to store the number of individual edges that were aggregated.

Definition 5.5 (Multiplicity λ). Let $\mathcal{E}_a$ denote a set of parallel edges that are aggregated into a single aggregated edge a. Then the multiplicity $\lambda : \mathcal{A} \to \mathbb{N}$ is defined as $\lambda(a) := |\mathcal{E}_a|$.

Based on these four attributes, we can combine parallel edges in $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to create the aggregated graph $\mathcal{G}_A = (\mathcal{V}, \mathcal{A})$ as we describe in the following. For containment edges, only the importance and the number of aggregated edges are meaningful. For the importance of containment edges, note that the exponentiation turns the distances into a value of 1 for existing edges and 0 for missing edges. An overview of the edge attributes is shown in Table 5.1.

τ	publication time	win	context window
δ	textual sentence distance	κ	context embedding
ς	sentence index	λ	# aggregated edges
ι	instance of cooccurrence	ω	edge importance
ε	term embedding	η	node type

Table 5.1: Overview of edge and node attributes in the network.

5.3.4 Edge aggregation schemes

For the aggregation of edges, two settings should be considered. If real-time queries on streaming data are of interest, a streaming aggregation can be used to process documents as they come in, and merge new edges to existing ones. Conceptually, this resembles the streaming first story detection that has been proposed for microblogs [148], but retains the entire contextual information. In this case, extracted edges are treated as information fragments that can be merged with existing edges (if the context is sufficiently similar) or treated as new edges (if the context is sufficiently different). We refer to this as the *streaming approach*. Alternatively, all edges of all articles can be stored to retain the unaggregated information, similar to the basic implicit network approach, but with parallel edges. In this case, the edges are then aggregated locally between pairs of nodes at query time or during an exploration of the data. We refer to this as the *clustering approach*.

Streaming edge aggregation

A streaming aggregation supports the (near) real-time analysis of documents from the stream as they become available, and utilizes a *similarity threshold* parameter th. As new documents d are added to the collection, multigraph representations $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$ are constructed. Each edge in $\mathcal{G}_d$ is inserted into the collection graph $\mathcal{G}_A$ by aggregating it with existing edges based on context similarity. To measure the context similarity, any suitable vector similarity measure can be used to compare the embeddings. We then distinguish between three cases for a new edge $e = (v, w, i) \in \mathcal{E}_d$.

(i) If e is a containment edge, it is added to the set of aggregated edges $\mathcal{A}$.

(ii) If v and w are disconnected in $\mathcal{G}_A$, then e is added to $\mathcal{A}$.

(iii) Otherwise, if $\mathcal{G}_A$ already contains edges between v and w, we check if e is sufficiently similar to the context centroid vector of an existing edge. We aggregate e with the existing edge $a \in \mathcal{A}$ that is the most similar, and update the edge attributes accordingly. If no existing edge is similar enough, e is inserted into $\mathcal{A}$.

For a more detailed description of the streaming aggregation that includes all aggregation steps based on the notation introduced above, see Algorithm 2.

Clustering edge aggregation

In contrast to the dynamic aggregation that occurs as new documents are available, the clustering aggregation of edges is a post-hoc processing of the collected document stream, in which parallel edges are first collected and then clustered. To implement a clustering aggregation effectively, a clustering algorithm is required that does not rely on a fixed number of clusters as a predetermined input parameter, since the optimal number of aggregated edges per pair of nodes is a priori unknown and highly varying for different pairs of nodes. Additionally, outliers and noise should be kept separate from the clusters, since many news articles do not belong to major news stories. Therefore, density-based clustering approaches are likely a good choice. Once the edges between each pair of nodes are assigned to clusters, edges within each cluster are then simply aggregated into a single edge. For a more detailed description of the process, see Algorithm 3.

5.3.5 Complexity and stability of the aggregation

Given the differences between the two approaches, it is worth considering their asymptotic complexity to assess the viability of using them on document streams with large volume or frequent updates. Furthermore, since streams of documents with time stamps can be considered to have a natural ordering, the stability of the produced results is of interest.

Asymptotic complexity

The complexity of the streaming approach is dominated by the number of generated parallel edges, which is given by the number of instances I. As a result, the complexity is in $O(I \cdot \langle p \rangle)$, where $\langle p \rangle$ is the average multiplicity of parallel *aggregated* edges between node pairs. The number of instances I scales linearly with the number of articles N for a given cooccurrence window size, which is identical to our observation for the static model. In contrast, $\langle p \rangle$ is a new factor that arises from the parallel edges even after the aggregation. However, in practice it is small enough to support similar edge detection by linear scans, as we show in our evaluation in Chapter 5.5.4. The complexity of the clustering approach also depends on the number of instances with $O(I \cdot C)$, where C is the complexity of the selected clustering algorithm. As a result, the complexity is likely higher, since the complexity of most clustering algorithms is likely at least quadratic in the edge multiplicity

Algorithm 2 Addition of edges to the network in the streaming approach.

Input: $\mathcal{G}_A = (\mathcal{V}, \mathcal{A})$, document graph $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$, threshold th

$\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{V}_n$
for $e = (v, w, i) \in \mathcal{E}_d$ **do** ▷ for all new edges
  $\mathcal{E}_a \leftarrow \{(v', w', i) \in \mathcal{E}_d \mid v = v' \wedge w = w'\}$
  if $\mathcal{E}_a = \emptyset$ **then** ▷ if this is the first edge betw. v and w
    $a \leftarrow (v, w, i)$ ▷ create new aggregated edge
    $\lambda(a) \leftarrow 1$ ▷ set multiplicity to 1
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ ▷ insert as a new edge
  else ▷ otherwise, find candidates for merging
    $a \leftarrow \arg\max_{a' \in \mathcal{E}_a} \{sim(e, a')\}$ ▷ find most similar edge
    if $sim(e, a) \leq th$ **then** ▷ if similarity below threshold
      $\lambda(a) \leftarrow 1$ ▷ set multiplicity to 1
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ ▷ insert as a new edge
    else
      $\omega(a) \leftarrow \omega(a) + \omega(e)$ ▷ update importance
      $\mathcal{T}(a) \leftarrow \mathcal{T}(a) \cup \{\tau(e)\}$ ▷ merge date sets
      $\kappa(a) \leftarrow \frac{1}{\lambda(a)+1}(\kappa(a)\lambda(a) + \kappa(e))$ ▷ update context
      $\lambda(a) \leftarrow \lambda(a) + 1$ ▷ increase multiplicity

Output: $\mathcal{G}_A$

Algorithm 3 Aggregation of edges in the clustering approach.

Input: Multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, clustering algorithm

Initialize $\mathcal{A} \leftarrow \emptyset$ and $l \leftarrow 0$
for $(v, w) \in \mathcal{V} \times \mathcal{V}, v < w$ **do** ▷ for all pairs of nodes
  $\mathcal{E}' \leftarrow \{(v', w', i) \in \mathcal{E} \mid v = v' \wedge w = w'\}$ ▷ select edges
  $C \leftarrow \text{cluster}(\mathcal{E}')$ ▷ cluster edges by context similarity
  for $\mathcal{E}_a \in C$ **do** ▷ for each cluster of edges
    $l \leftarrow l + 1$ ▷ increase edge index
    $a \leftarrow (v, w, l)$ ▷ create new aggregated edge
    $\lambda(a) \leftarrow |E_a|$ ▷ set multiplicity
    $\omega(a) \leftarrow \sum_{e \in E_a} \exp(-\delta(e))$ ▷ aggregate importance
    $\kappa(a) \leftarrow \frac{1}{|E_a|} \sum_{e \in \mathcal{E}_a} \frac{\kappa(e)}{\delta(e)}$ ▷ combine contexts
    $\mathcal{T}(a) \leftarrow \bigcup_{e \in \mathcal{E}_a} \{\tau(e)\}$ ▷ merge date sets
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ ▷ insert as a new edge

Output: Aggregated graph $\mathcal{G}_A = (\mathcal{V}, \mathcal{A})$

of *unaggregated* edges. Thus, streaming aggregation tends to be faster than the clustering aggregation, which we also observe in practice. However, due to the localized clustering of edges between two nodes, the clustering approach is parallelizable by node pairs.

Stability

In regard to the stability of results, we find that both approaches produce deterministic results in principle. However, in the case of the streaming approach, this depends on the temporal order of documents in the stream. If documents with an identical time stamp are processed in a different order, the results are not guaranteed to be identical. On the other hand, the clustering aggregation approach is stable if and only if the used clustering algorithm is stable. Obviously, the aggregated graphs that are produced differ between the two approaches, and we compare their efficacy in Chapter 5.5.

Finally, it is worth noting that the clustering approach can also be applied in a streaming setting in practice, as long as sufficient memory is available to cluster edges locally at query time. Thus, both approaches are in principle usable in a streaming setting, and both account for the dynamics of entity relations. In the following, we investigate their application for retrieval and exploration tasks.

5.4 Entity Context Exploration

As we have demonstrated in the previous chapters, tasks that can be formulated as an entity or term rankings, or as the extraction of weighted entity graph patterns, can be addressed by using an implicit network model. In particular, events as dyadic or triadic structures of entities can be queried efficiently as we showed in Chapter 3. Due to the transitivity of edge aggregation (edges can always be aggregated further), all entity-centric exploration methods designed for the static implicit network also work with the context-enriched dynamic model. We thus focus on novel exploration methods that utilize temporal data and the context of entity mentions to extract evolving entity-centric topics from document streams. To this end, we show exploratory results on a large stream of entangled news articles from several news outlets.

5.4.1 Entangled news stream data

Since news streams are a typical example of complex, entangled document streams from multiple sources, we use them as data for our exploration. We first describe the acquisition and preparation of the news data, as well as the construction of the graph representation.

Data collection

Since we want to analyze the model on entangled news streams from multiple outlets, standard corpora from a single outlet such as the New York Times corpus [159] cannot be used. Instead, we collect articles from the RSS feeds of international outlets with a focus on high-quality news. To extract the content, we use manually created rules, since these enable a clean extraction of article contents (including multi-page articles) at a level that automatic boilerplate removal does not support [180].

Specifically, we use articles from 14 English speaking news outlets located in the U.S. (CNN, Los Angeles Times, New York Times, USA Today, CBS News, The Washington Post, International Business Times), Great Britain (BBC, The Independent, Reuters, Sky News, The Telegraph, The Guardian), and Australia (Sydney Morning Herald). The RSS feeds of these outlets differ, but we focus on feeds that cover political news. The time frame for our data collection is June 1 to November 30, 2016. We remove articles that have less than 200 or over 20,000 characters (due to limitations of the named entity recognition framework). We also remove articles that contain more than 100 disambiguated entities per article (typically, these are not articles but lists of real estate in weekend editions of newspapers). The final collection of articles then contains 127,485 time-stamped documents over a period of six months, with a total of 5.4M sentences.

Data preparation

Similar to the Wikipedia implicit network, we again focus on named entities of the types location, organization, person (actor), and date, since these correspond well to the central entities of news events. Data preparation then consists of five steps: recognition of named entities, entity linking, entity classification, part-of-speech and sentence tagging, and temporal tagging. For the recognition and disambiguation of named entities to Wikidata IDs, we use the Ambiverse natural language understanding suite [93]. To classify named entities into actors, locations, and organizations, it would be possible to use Wikidata hierarchies directly, but this can be problematic due to their constantly evolving structure [176]. Therefore, we map Wikidata IDs to YAGO3 entities [124] and classify them according to the YAGO hierarchy, since it is derived from WordNet hierarchies and easier to handle (see Chapter 2.4). For actors, we use the class `wordnet_person_100007846`, and for organizations `wordnet_social_group_107950920`. For locations, no comprehensive WordNet class exists, so we use `yagoGeoEntity`, which was designed specifically for this

purpose [96]. For the extraction and normalization of temporal expressions, we run HeidelTime in the news domain setting [188]. Finally, for sentence splitting and part-of-speech tagging, we use the Stanford POS tagger [200].

Implicit network construction

To construct the network, we proceed as described in Chapter 5.3. We again use stemming instead of lemmatization, with the same reasoning as in the case of Wikipedia in Chapter 3.3, and rely on the Porter stemming algorithm [149]. We impose a minimum word length of 4 characters for terms, and set the window size for the extraction of entity cooccurrences to $c = 5$. For the term embeddings that encode the cooccurrence context, we use Google's pre-trained 300-dimensional word2vec [131] word embeddings as an out-of-domain source that is trained on a much larger corpus of news articles.

The resulting network then has 5.7K dates, 27.7K locations, 72.0K actors, 19.6K organizations, and 351K terms, which are connected by 83.4M parallel edges (prior to aggregation). While this data is therefore substantially smaller than the Wikipedia implicit network, it contains six months of news, which is a reasonable time frame for our analyses.

5.4.2 Contextual topic evolution

To highlight the exploratory possibilities of the model, we demonstrate the extraction of evolving contextual topics. To this end, we extract topics that best describe the individual contexts in which two entities are mentioned together and consider their evolution over time. Naturally, multiple such contexts may exist for any given pair of entities, which is reflected by the multiple parallel edges.

Contextual topics

We first provide a description of our approach to the extraction of contextual topics, before we consider their evolution. Recall that a context vector $\kappa(a)$ is associated with each aggregated edge $a = (v, w)$. We define a *contextual topic* of edge a as a weighted list of terms that describe the context in which entities v and w occur in instances included in a. To extract the contextual topics for each aggregated edge between these entities, we retrieve all terms $T_x = N(v) \cap N(w) \cap T$ in the joint neighbourhood of the two nodes, along with all edges that connect them to v or w. We aggregate these edges such that each term x is connected to both v and w by exactly one edge, which we denote with a_v and

a_w. Based on these triangular structures, we obtain a ranking score for each term $x \in T_x$ in relation to edge a as

$$\varrho_t(x|a = (v, w)) := \min\{\text{sim}(\kappa(a), \kappa(a_v)), \text{sim}(\kappa(a), \kappa(a_w))\} \tag{5.10}$$

Intuitively, we are ranking terms by how closely the context in which they occur with an entity matches the context in which the entities occur together. We create such a ranking of terms for all aggregated edges between v and w. For each such edge, we select the k top-ranked adjacent terms to describe the topic. Thus, we obtain a natural language description for each of the edges between the two entities. Since edges are aggregated based on context similarity, the assumption is that the terms then describe the context of an aggregated edge, and that each aggregated edge in turn represents a topic. Furthermore, since edges are also attributed with temporal information in the form of publication dates of the articles that induced these edges, we can consider the evolution of edge-centric topics over time, which we do in the following.

Exploration results

To demonstrate the expressiveness of contextual topics, we show a timeline visualization of contextual topics for pairs of entities. To extract these topics, we use a cosine similarity of the context vectors and rank the adjacent terms as described above. Then, we assign to each aggregated edge between the two entities the $k = 5$ top-ranked terms as topic descriptors. We select the three top edges by multiplicity (that is, the aggregated edges with the highest λ values). Since each such edge is associated with a set of publication times, we can plot the evolution of the topics over time. The results for two entity pairs are shown in Figure 5.2. At the top, we see the evolution of contextual topics for Brazil and the IOC (that is, the International Olympic Committee, which is the organization to which the Olympic Games are linked in this data set). One can easily identify contexts as dealing with corruption, sports, and the awarding of medals. Specifically, the award topic spikes precisely at the date of the games. The second example shows the relation of Prime Minister David Cameron to the United Kingdom during the beginning of the Brexit crisis. While all three topics are related to this issue, the referendum topic spikes at the proper date, and the drastic shift of the remaining topics towards Cameron's resignation occurs only after the result of the referendum is announced.

Overall, the intuitive notion of aggregated edges as contexts in which entities are mentioned corresponds well with our observations. Thus, term-based topic descriptors can

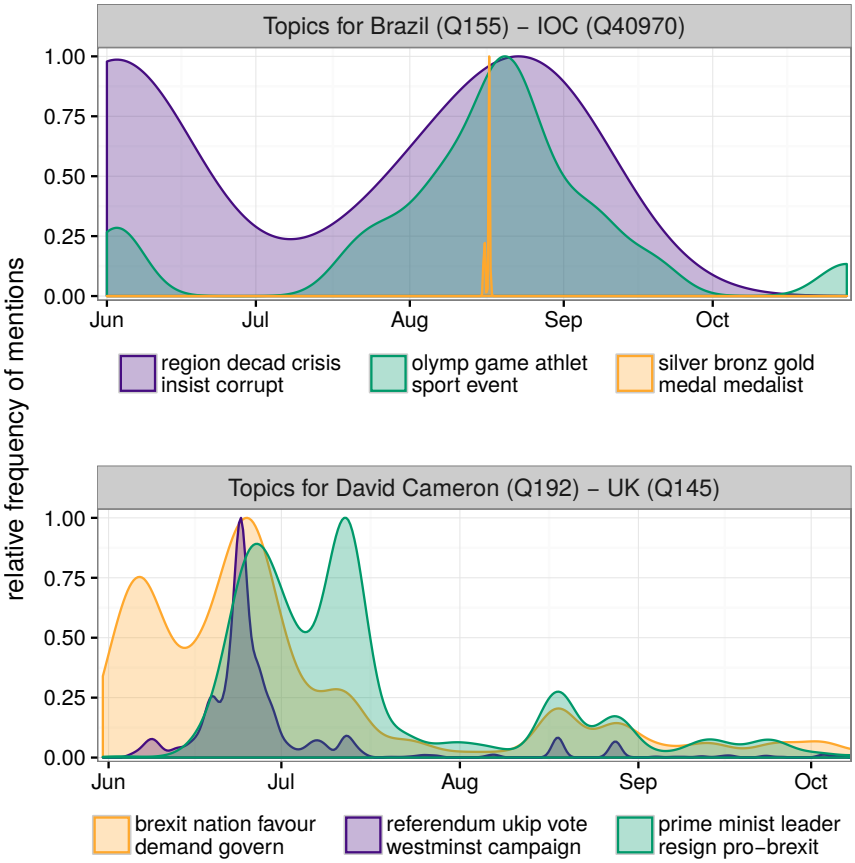

Figure 5.2: Evolution of the context of edges as contextual topics for two selected entity pairs with streaming aggregation (using cosine similarity and an aggregation threshold of $th = 0.65$). Shown is the relative aggregated frequency of publication dates for the three edges with the highest multiplicity λ. Contexts are derived from the joint neighbourhood of both entities by selecting the $k = 5$ terms whose context is most similar to the edge context. Q identifiers denote Wikidata IDs. Top: relation between Brazil and the International Olympic Committee. Bottom: relation between David Cameron and the United Kingdom.

be used to assign meaning to such edges, and their extraction serves to facilitate an exploratory analysis of the news stream. Since the extraction utilizes only a localized substructure of the network around the focus entities, the process is efficient and enables a near real-time exploration of the entire entangled news stream. Alternatively, a subset of news outlets and focus entities can be selected by the user for a contrastive analysis between outlets, or context terms can be employed as additional input to quantify *how* a given news outlet reports about a specific group of entities.

However, we also note that this definition of a contextual topic is just one possible interpretation of an entity-centric topic, which is limited in its scope to one edge between exactly two entities. We go into more detail regarding the extraction of entire subgraphs as entity-centric network topics in Chapter 6.

5.5 News Event Completion Evaluation

To demonstrate the validity of the dynamic network model beyond exploration scenarios, we evaluate the streaming and clustering aggregation approach on a set of news events. As data set, we use the same collection of news articles as in the exploration above.

5.5.1 News event completion task

We evaluate versus the static implicit network as a baseline and use an event participation prediction task, similar to the date prediction that we used to evaluate the implicit Wikipedia network in Chapter 3.4. As an extension of the date prediction task, the event completion task can be defined as follows: Given $k-1$ out of k entities participating in an event as query entities, the goal is to predict the remaining hold-out entity based on the data. Thus, the evaluation works in the same way as before, by ranking entities in the target set based on a set of query entities, and comparing the predicted entity to the ground truth. However, the type of target entities is not limited to dates in this case. Thus, both the implicit network baseline and the dynamic implicit networks are used to rank entities x in the target set $X \in \{\textsc{Loc}, \textsc{Org}, \textsc{Act}, \textsc{Dat}\}$ based on a set of query entities $Q \subseteq \textsc{Loc} \cup \textsc{Org} \cup \textsc{Act} \cup \textsc{Dat}$.

Implicit network ranking (baseline)

Recall the *tf-idf*-like normalization of edges of the graph that we introduced in Chapter 3.2 to rank entities $x \in X$ based on a query entity q by computing a normalized importance

score. We refer to this scoring function as the baseline ϱ_B. In terms of the dynamic implicit networks, this ranking score can be expressed for a single query entity as

$$\varrho_B(x|q) := \left(\log \frac{|Q_{\eta(q)}|}{|N(x) \cap Q_{\eta(q)}|}\right) \sum_{e=(x,q,\cdot)\in\mathcal{E}} \exp(-\delta(e)) \tag{5.11}$$

where $Q_{\eta(q)}$ denotes the set of all entities of the same type as q (for example, locations or organizations). As discussed in Chapter 5.3.3, this scheme aggregates *all* parallel edges into a single edge, which is weighted by the sum of individual edge importances (that is, the importance weight $\vec{\omega}$ in the static model). Thus, the context of edges is lost in this step in the static model. To generate a ranking based on multiple input entities, we sum over the contributions of individual query entities, and include the coherence as introduced in Chapter 3.2, requiring that each target entity is linked to at least $\min\{coh, |Q|\}$ query entities. As in the original implicit network model, we use a cohesion value of $coh = 2$ in the following evaluation.

Context-sensitive network ranking

In contrast to the static model, two entities may be connected by more than one edge in the dynamic model, which we use to differentiate between target candidates in a context-sensitive ranking. Let $\mathcal{E}_a(x)$ denote this set of aggregated edges between a query entity q and an entity x in the target set. Furthermore, where available, we can include the context of the event description in the query to match the context of candidate entities. Let $\kappa(q)$ denote the context of query entities in the event, which we include in the context-sensitive ranking. We can then define this context-sensitive ranking function ϱ_C as

$$\varrho_C(x|q) := \max_{a\in\mathcal{E}_a(x)} \left[\mathrm{sim}(\kappa(a), \kappa(q)) \left(\log \frac{|Q_{\eta(q)}|}{|N(x) \cap Q_{\eta(q)}|}\right) \omega(a)\right]. \tag{5.12}$$

Intuitively, we normalize the importance of a candidate entity to the query entity with the similarity to the query context (as measured by some vector similarity measure *sim*). Then, we use the best contribution as a ranking score for the candidate entity. While any suitable vector similarity function can be used, we use the cosine similarity in the following since it works well for the comparison of embedding vectors. To obtain a ranking by multiple query entities, we also rely on the notion of improved coherence as introduced in Chapter 4.2, and rank candidates first by the number of neighbours in the query set $|N(x) \cap Q|$, and break ties by the sum of ranking scores ϱ_C.

5.5.2 Evaluation setup

We briefly discuss the setup of the evaluation, the used parameters, and the ground truth, before we present the evaluation results.

Context extraction schemes

To derive contexts for entity cooccurrences, we consider two schemes according to the definition in Chapter 5.3.2: the complete context and the verb context. For the *complete* context as a straightforward approach, we simply use the weighted average embedding of *all* non-stopwords inside the context window. In contrast, based on the importance of verbs for traditional event extraction, we also consider the *verb* context. In this case, we utilize only the embeddings of verbs inside the context window. However, we exclude all forms of the auxiliary verbs *to be* and *to have*. Both schemes are applied separately during network and ground truth construction.

Ground truth data

Since we cannot rely on historic events for the evaluation of contemporary news streams, we need a set of news events. Thus, to obtain ground truth events, we use the Wikipedia Current Events portal [214], which contains summaries of news events with short descriptions that are manually updated by Wikipedia editors (for an example, see Figure 5.3). We crawl the pages to extract each listed item as a news event, and perform named entity recognition and disambiguation by following Wikipedia links in the text, similar to our annotation of Wikipedia in Chapter 4.2. Since the Wikipedia summaries contain references to the original news article sources from which the events were taken, we match the references to articles in our input stream to ensure that the described events are covered in our documents. We exclude all events that consist of less than two entities or have no reference to an article in our network. We obtain 97 individual events that correspond to at least one article in our collection. For each such event, we generate a query from each contained entity by using the remaining entities as query input and the removed entity as ground truth (that is, an event with k entities induces k queries). All words in the event descriptions that are not annotated as entities are extracted as terms for the generation of query contexts. To obtain the verb context, we also manually annotate the verbs in the event summaries. In total, we obtain 293 queries for the evaluation.

As an example, consider the second item in the list of events in Figure 5.3. In addition to the date of July 16, 2016, it contains mentions of the *President of Turkey* as a person,

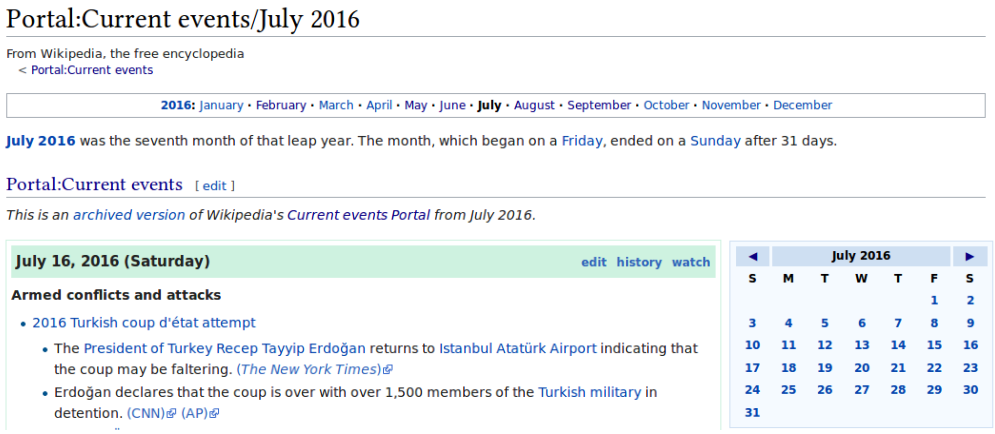

Figure 5.3: Snapshot of an event description for July 16, 2016, from the Wikipedia Current Events portal. Events are sorted by days and annotated with brief event descriptions, as well as links to the source articles that describe the event.

and *Istanbul Airport* as a location. From this set of three entities, we can construct three evaluation queries by removing one hold-out entity from the set and using the remaining two as query input. For example, we try to predict the location *Istanbul Airport*, based on the set of query entities containing July 16, 2016, and *President of Turkey*. We also construct two further queries with the other two entities as targets. In contrast to this example, entities are of course represented by their Wikidata identifiers (or normalized dates) in the evaluation. All remaining terms in the sentence are then be considered for the generation of the context embeddings. However, the verb context is constructed from only the embeddings of *return*, *indicate*, and *falter*.

Clustering aggregation setup

For the clustering approach, we require a clustering algorithm that does not enforce a fixed number of clusters, since it is impossible to divine a reasonable number of clusters that applies equally to all pairs of nodes. Thus, we select DBSCAN [54] with cosine as a distance measure. To determine values for the necessary two parameters ϵ and $minPts$, we conduct a number of preparatory tests. In the results of these tests, we find that the quality of the results suffers for high values of $minPts$, while $minPts = 5$ works well. Since a value of $minPts > |\mathcal{E}_a|$ exceeds the number of edges and would therefore be meaningless for edge aggregation, we use the scheme $minPts = \min\{5, \frac{|\mathcal{E}_a|}{5}\}$, which performs best in our experiments. We then employ the min-points heuristic to obtain a reasonable density value of $\epsilon = 0.3$ as a starting point for the evaluation.

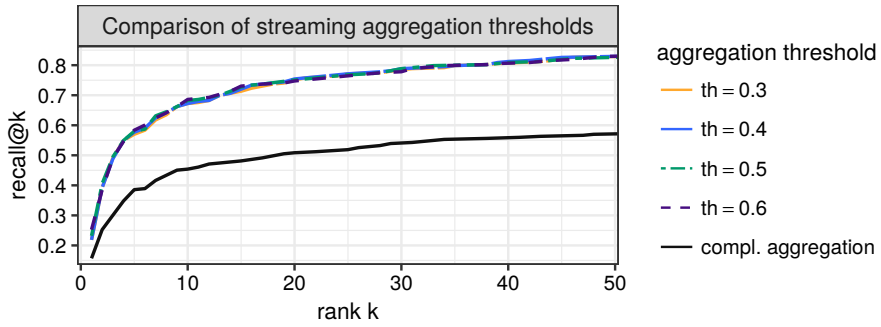

Figure 5.4: Recall comparison for different aggregation threshold values in the streaming aggregation approach, when using the complete context to generate context vectors, and cosine as a similarity measure (the results for the verb context are almost identical and omitted). Shown is the fraction of correctly identified entities in the top k ranks versus the rank k. Complete aggregation corresponds to the static implicit network model and is included as a baseline.

	$th = 0.3$	$th = 0.4$	$th = 0.5$	$th = 0.6$
context (all)	0.218	0.218	0.232	**0.253**
context (verb)	0.225	0.222	0.215	0.208
no context		0.157		

Table 5.2: Evaluation results of the streaming edge aggregation with cosine similarity. Shown is the precision@1 for the complete context embedding derived from all words in the context window, as well as the context that is derived only from verbs. The aggregation without context corresponds to the static implicit network model and is included as a baseline.

5.5.3 Evaluation results

As discussed above, each evaluation query has exactly one correct answer. Therefore, suitable evaluation metrics are the fraction of queries in which the top-ranked prediction is correct (that is, precision@1), and the number of correct predictions among the top k predictions (that is, recall@k). We discuss the results in the following.

Streaming aggregation

We first compare the two approaches for context generation over varying aggregation thresholds, and show the resulting precision scores in Table 5.2. We omit threshold values of $th < 0.3$ since no further changes occur below this point in our data. We find that both context generation methods outperform the static implicit network baseline by a large margin of up to 61% improvement). However, the verb context aggregation shows

	full aggr.	streaming aggr.		clustering aggr.		
		all	verb	$\epsilon = 0.2$	$\epsilon = 0.3$	$\epsilon = 0.4$
cor@1	44	71	61	35	27	25
prc@1	0.165	**0.266**	0.228	0.131	0.101	0.094
recall	0.655	0.955	0.955	0.955	0.955	0.955

Table 5.3: Performance comparison of the clustering and streaming ($th = 0.6$) edge aggregation approaches on a subset of the evaluation data. We show the correct predictions at rank one (cor@1), precision@1, and recall. The full aggregation corresponds to the static implicit network model and is included as a baseline.

a slight decline in performance as the threshold increases. In contrast, the precision of the complete context increases with the threshold value, and it performs better overall. In Figure 5.4, we show the corresponding recall values of the complete context approach, which are almost identical for the verb context, and thus omitted. Varying the threshold values has little influence on the recall, which makes low thresholds attractive in settings where a compact graph representation with as few edges as possible is important and a good recall@5 score is sufficient. Overall, we find that even a heavily aggregated context is still sufficient to increase the performance in comparison to the static implicit network that aggregates all edges regardless of context.

Clustering aggregation

In Table 5.3, we show the performance of the clustering aggregation for a subset of 267 evaluation queries (the remaining 26 clusterings did not finish within 48 hours). We also include the static implicit network as a baseline and the best results of the streaming aggregation approach for comparison. Due to the smaller evaluation set, note that the values vary slightly from the values in Table 5.2. We find that the clustering aggregation performs better than the static implicit network baseline for some of the ϵ settings, but not by a large margin. Interestingly, higher values of ϵ decrease the performance. The recall values shown in Figure 5.5 support the observation of a lower performance of the clustering approach. While some of the clustering aggregation settings eventually outperform the static baseline, clustering does not rival the streaming aggregation.

Overall, we therefore find that streaming edge aggregation is superior to clustering aggregation in this setting. While other clustering algorithms may perform better, our tests were extensive, and the ease of use for the streaming method is much higher. While the optimal parameter settings for clustering approaches are typically difficult to obtain, there

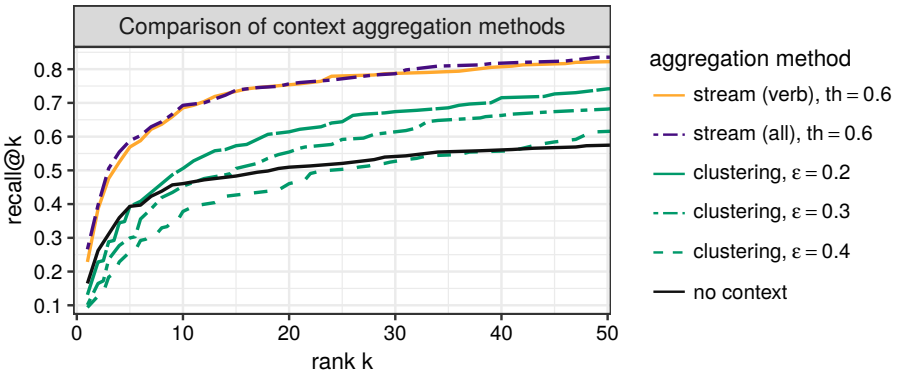

Figure 5.5: Performance and recall comparison for different density values in the clustering aggregation approach using cosine similarity. DBSCAN is used as a clustering algorithm with $minPts = 5$ and varying ϵ values. Shown is the fraction of correctly identified entities in the top k ranks versus the rank k. The aggregation without context corresponds to the static implicit network model and is included as a baseline.

is a direct correlation between the threshold and the prediction quality in the streaming approach. Therefore, the streaming approach is preferable for the analysis of news, not least because it also saves storage space in comparison to the clustering approach, which requires all edges to be collected prior to the aggregation. What remains to be evaluated is the performance of the streaming approach during aggregation, which depends on the threshold selection. We consider this aspect of the model in the following.

5.5.4 Edge deflation for streaming aggregation

The streaming aggregation approach is designed to reduce the number of aggregated edges that have to be stored in the graph representation of the network to a manageable size, and thereby avoid unnecessary redundancy. Especially for entangled news streams, many parallel edges with highly similar context are to be expected due to duplicate articles, similar breaking storylines, or reused content from third party publishers or news agencies. Furthermore, the number of existing aggregated edges negatively influences the runtime of the aggregation of newly generated edges between the same set of entities. Therefore, it makes sense to investigate which degree of compression of the edges can be achieved for a given threshold value.

In Figure 5.6, we show the number of aggregated edges as a function of the number of unaggregated edges for different threshold values applied to the complete context embeddings. We find that aggregation is almost complete and there are never more than three parallel edges for thresholds $th \leq 0.3$. For higher thresholds, this number increases

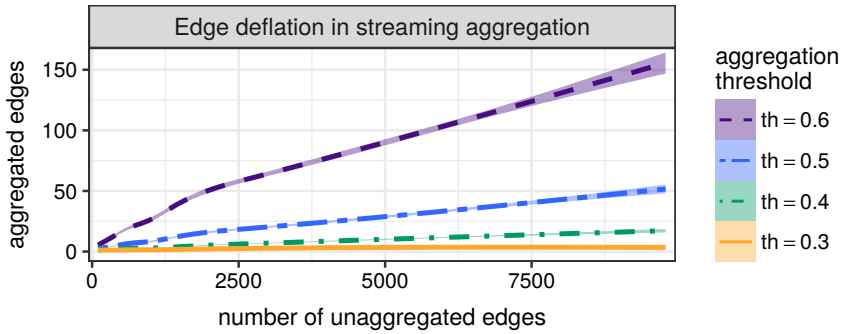

Figure 5.6: Average deflation potential of parallel edges for the streaming aggregation approach with complete context. Shown is a linear fit to the average number of edges after aggregation versus the number of edges prior to aggregation. Shaded areas denote the 0.99 confidence interval.

but is still easily manageable. Since higher thresholds are favourable with regard to the extraction of information from the graph, the threshold thus has to be tuned to the data throughput in an application scenario. For the real-time processing of streams of news articles, this is unproblematic due to the relatively low volume of documents in the news domain. For higher frequency streams, such as the entire blogosphere, however, more sophisticated data structures or similarity approximation heuristics for the computation of edge similarities may be desirable.

5.6 Summary and Discussion

We introduced dynamic implicit networks as a way of extracting and exploring the relations of entities in multiple, potentially entangled streams of documents, and tested the concept on a large stream of news articles. Based on the intuition that entity mentions can serve as stitching points between the news streams and act as focal points for news retrieval tasks, we also investigated the impact that the context of entity mentions has when it is used as an edge attribute. The resulting contextual implicit entity networks can serve as a comprehensive and versatile tool for the representation of such entangled news streams, or other collections of documents with a similar temporal dimension.

On the technical side, we find that the construction speed of the graph representations of documents primarily depends on the speed of the natural language processing and entity recognition tools, while the extraction and aggregation of edges are unproblematic, even in a streaming setting. As a result, the combined representation of entire document

streams can be constructed faster than the publication speed of news articles by several orders of magnitude, and thus efficiently facilitates a multitude of subsequent entity-centric information retrieval tasks from the underlying streams in near real-time.

Practical implications

Consider the case of our journalists and data analysts. For some of their projects, they might be presented with one large stack of documents that is leaked or obtained at the same time, and dumped in their laps for analysis, such as the Panama Papers. Under different circumstances, however, this data set might be expanded at a later time through further inquiries, or there might not even be an initial set of data, but only a slow trickle of documents and reports that are obtained by informants and researchers in the field, such as in the case of the collusion investigation. Clearly, a method that can handle a heterogeneous stream of incoming data is beneficial in this situation. In either case, many documents in the data likely have a temporal component and the investigation of the evolution of cooccurrences over time is of interested. Thus, by using the dynamic implicit network model, it becomes possible for these researchers to investigate the evolution of the relations between interesting entities over time.

By taking the context of edges into account, it is now also possible to better distinguish between multiple facets of entity relationships. For example, to find institutions or organizations in the Panama Papers that are mentioned together on multiple occasions, but with respect to different clients. Alternatively, in the case of the collusion investigation, an analyst can now try to determine potential affiliations and the tendencies of suspects to collaborate with one involved party or the other. In contrast to the static implicit network model, which offered the researchers a novel angle on entity relations, the dynamic model potentially provides multiple angles for a broader picture.

Outlook

Similar to the static implicit networks in Chapter 3, we only explored dynamic entity relations in a local context by ranking edges in the neighbourhood of query entities. Therefore, what remains to be explored is the usefulness of the graph structure for global instead of local rankings of edges. Furthermore, the graph structure itself is a useful exploration and visualization tool, since the graphs enable a visual and intuitive representation of entity and term relations. In Chapter 6, we thus consider the extraction and visualization of descriptive subgraphs from implicit networks as entity-centric topics. Furthermore, the

visual representation then allows us to perform a comparative analysis between individual news streams, for example to assess biases towards certain topics in their coverage.

While we evaluated the model's performance for different parameter settings on a large collection of news streams, what remains to be explored is a more fundamental approach to the representation and querying of such implicit networks. We address this issue in Chapter 7, where we consider a generalized hypergraph representation of term and entity cooccurrences in large document collections, and discuss how they can be mapped to established relational database architectures.

6 Entity-centric Network Topics

In the previous chapters, we have focused on the exploration of relations in the immediate (joint) neighbourhood of entities and terms of interest, and derived local rankings to support the retrieval of entities, sentences, or even documents. However, we have not yet investigated how a global ranking of edges can be used, and we have only extracted and visualized static entity-centric subgraphs. In the following, we focus on global rankings of edges to identify the seeds of interesting topics, around which we can dynamically grow descriptive subgraphs that make the most important relations accessible and enable us to track their evolution over time.

Contributions. In this chapter, we thus make the following three contributions.

I We introduce and formalize entity-centric network topics for the analysis of document streams, and analyze their relation to traditional topic models.

II We use network topics to perform a contrastive comparison of news streams, and investigate the evolution of their contents over time.

III We demonstrate how entity-centric topics can be visualized and explored interactively for large news streams in a Web-based user interface.

References. Parts of this chapter are based on the peer-reviewed publications:

A. Spitz and M. Gertz. "Entity-Centric Topic Extraction and Exploration: A Network-Based Approach". In: *Advances in Information Retrieval - 40th European Conference on IR Research (ECIR)*. 2018, pp. 3–15. DOI: 10.1007/978-3-319-76941-7_1

A. Spitz, S. Almasian and M. Gertz. "TopExNet: Entity-centric Network Topic Exploration in News Streams". In: *Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM)*. 2019. DOI: 10.1145/3289600.3290619

6.1 Motivation

Given a document stream or a collection of time-stamped documents spanning a period of time, a question that is commonly raised in corpus analytics and natural language processing concerns the topics that are covered in the documents. Approaches to answering this question range from the exploration of news media to the analysis of emails or even historic documents, such as correspondence letters [117]. Thus, topic modelling is considered an important tool in the analysis of corpora and in the classification and clustering of documents. For streams of documents or time-stamped document collections in particular, it is often of interest to discern the dynamics of the contents and analyze how identified topics evolve over time. In most cases, the theoretical background for the extraction of topics is provided by probabilistic topic models, which are based on *Latent Dirichlet Allocation*, or LDA [31]. In these models, documents are assumed to be generated by one or more topics, each of which is a distribution of words. Once identified, the topics of a document collection can then also be used, for example, in the classification or clustering of documents. Due to this wide range of applications, a multitude of tools for computing topics and several extensions to LDA have been proposed, such as hierarchical or dynamical topic models (for an overview, see [28]).

Despite the range of applications that use topic models, topics are often simply represented as lists of ranked terms, some of which are even difficult to associate with the topic, especially at lower ranks of the list. More recently, there have thus been approaches that enable the exploration and analysis of the computed topics and ranked terms [77, 99]. To utilize additional annotation data during topic extraction, some approaches also include named entities in the topic model [86, 143], an aspect that is important for the analysis of news articles, since they typically revolve around entities. However, despite their popularity, topic models face problems in the exploration and correlation of the topics that are extracted from a document collection, due to the a priori unknown number of topics in the collection, and the inability to adjust the number of topics without repeating the entire extraction process. The latter is problematic in particular due to the complexity of the underlying graphical models that are computationally expensive. As a result, the extraction of topics can be inflexible in practice, and difficult to adjust to an efficient exploration of topics in streams of documents. This issue is problematic for the automated processing of news in particular, due to the large number of individual topics that are covered by only a handful of documents, thereby drastically scaling up the number of topics.

In contrast, as we have seen in Chapter 5, implicit networks allow us to efficiently and effectively capture not only relations between entities and terms in streams of documents,

but also their relations to sentences or even documents. Since dynamic implicit networks can easily be updated as new documents in a stream arrive, this model is inherently capable of representing the evolution of entity and term relations in the data. Furthermore, subgraphs of the network may offer more intuitive insights into these relations than lists of words. In the following, we therefore investigate an alternative approach to the analysis of topics and their evolution over time that is based on an implicit network representation of the document streams, again with a focus on streams of news articles.

Based on the central role that the relations between entities play in the evolution of news stories, we conjecture that frequently cooccurring pairs of entities are indicative not only of relations, but also of topics. We thus base our approach on the derivation of edge weights that cover globally important relations instead of locally important edges, thereby allowing us to extract *seed edges* from global rankings. Starting from these seed edges between two entities in the network representation, we can then construct the *context* of a potential topic by following other highly weighted edges to adjacent terms and entities, similar to our approach in Chapter 5.4. With this approach, the exploratory character of topic discovery and the overlap between identified topics becomes apparent. Since important seed edges can easily be determined or expanded, the model is not constrained to a fixed number of topics. Furthermore, since the exploration of topics still corresponds to local operations in the network once global seed edges have been identified, it benefits from the implicit network model, making it both efficient and interactive.

For time-stamped documents in streams, publication dates provide an effective means of focussing on entity and term cooccurrences in a given time frame. Thus, as we have seen in Chapter 5.4, the evolution of topics in terms of edge weights and word contexts can be explored in the implicit network representation, which also supports the addition of new documents by adding new nodes, edges, and updating edge weights. Adding new documents to the collection then simply updates the network on a local scope. Most importantly, it is not necessary to recompute topics and word distributions for an evolving corpus when new documents are added. As a result, a network-based framework can support a variety of entity-centric topic analysis and exploration tasks in an on-line setting, as we demonstrate in the following.

Structure. In Chapter 6.2, we discuss related work on topic models. We introduce the network-based approach to the extraction of entity-centric topics in Chapter 6.3, and compare it to traditional LDA topic models in Chapter 6.4. In Chapter 6.5, we present an interactive user interface for the exploration of network topics in news streams, and discuss its implementation details.

6.2 Related Work

The application of implicit networks to the extraction of topics that we propose in this chapter is largely different from the traditional LDA-based graphical topic models. However, we briefly discuss these works to address their shortcomings and provide a background for our comparison in Chapter 6.4.

Since the introduction of topic models based on Latent Dirichlet Allocation [31], numerous frameworks for topic models have been proposed, such as hierarchical [29], temporal [30], or dynamic topic models [98] that are better suited to document streams. For an in-depth overview, we refer to the review by Blei, which summarizes the diverse approaches and directions [28]. However, since topic models primarily extract topics from a document collection in the form of ranked lists of terms, it has been questioned to what extent such representation are semantically meaningful or interpretable, beyond providing an approximate initial parameter of topics to discover. In particular, this issue has been raised by Chang et al. [41], who propose novel quantitative methods for evaluating the semantic meaning of topics. However, while this highlights the issue of coherence in extracted topics, it does not solve the underlying problem of topics as lists of words.

In contrast to the above, some approaches touch on aspects that are similar to the concepts we consider in the following. In particular, some related approaches use entity-centric topic modelling [86, 143], by focussing on (named) entities and their inclusion in the topics. Others use a network structure, but only of the documents themselves, and not their annotated contents [222]. A combination of term collocation patterns and topic models was recently proposed by Zuo et al. [231]. However, compared to the network-based approach that we consider here, their approach is tailored towards short documents and relies on LDA, thus incurring the same problems as the topic models outlined above. Furthermore, all these models share the inherent problems of divining an appropriate number of topics to discover, and the necessity of interpreting lists of ranked terms, which hampers an exploration of the extracted topics.

More recent contributions to topic modelling propose frameworks that enable a more interactive construction and exploration of topics from a collection of documents [77, 99], and provide the user with added flexibility in terms of corpus analysis. Another interesting direction that implicitly addresses the aspect of collocation is the combination of word embeddings and topic models [167]. A common theme of these extensions of probabilistic models is the reliance on a computationally expensive and relatively inflexible construction of lists of ranked terms, which are then explored a posteriori. In contrast to

these approaches, we use the underlying network document model to add versatility to the subsequent steps. Thus, we rely on implicit networks as a document representation that allows the localized extraction, rescaling, and growth of topics during the exploration of the document collection.

6.3 An Entity-centric Topic Model

In the following, we describe how an implicit network representation is constructed from a collection of documents $D = \{d_1, \ldots, d_n\}$ in a document stream to enable the generation of edge weights that support global rankings, and how topics can be extracted from this model. Similar to the case of dynamic networks discussed in Chapter 5, we assume that each document d has an associated timestamp $\tau(d)$. In the case of news articles, for example, these indicate the articles' publication times.

6.3.1 Implicit network construction

We denote the graph representation of the network for the document collection D with $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The set of nodes is constructed similarly to the static implicit network model discussed in Chapter 3, and contains entities and terms. The edges still represent the implicit relations that are extracted from the documents, but we rely on a different weighting scheme in support of the global ranking that we use to identify topic seed edges.

Network nodes

We again utilize entities to construct the network, due to the focus on entity-centric topics. The set of nodes is defined in the same way as for the static network model in Chapter 3.2. However, due to the focus on content, we do not consider sentences or documents as nodes of the network for the extraction of topics in the following. Instead, we only include terms and entities. We assume that words in the sentences have been tagged with respect to a known set of named entities E. After stop words are removed, the remaining untagged words are denoted as the set of terms T, and we let the set of nodes be $\mathcal{V} = E \cup T$. Furthermore, each node can be assigned occurrence statistics of the corresponding entity or term, such as a document or sentence position.

Network edges

To model the entity and term cooccurrences, we again utilize edge attributes in the network model, and first create a multigraph representation with a multiset of parallel edges $\mathcal{E}$, which we then combine to aggregated edges as needed. Due to the entity-centric focus of the topics, we restrict the set of edges that we consider in the following to the subset $\mathcal{E} \subseteq (E \times T) \cup (E \times E)$. That is, we only consider edges $e = (v, w)$ such that at least one of the nodes v or w corresponds to an entity, and we do not model pure term cooccurrences.

Edge attributes

Recall that the cooccurrence of two respective words has an occurrence distance δ that is measured in sentences and can be constrained by the size c of the cooccurrence window. For example, for $c = 0$, only cooccurrences in the same sentence are considered, for $c = 1$, the sentences directly before or after a given sentence, and so on. Edges $e = (v, w)$ are then generated from cooccurrences as described in Chapter 5.3. That is, edges are associated with three (co-)occurrence statistics, namely **(i)** the document $d(e)$ that contains the mention, **(ii)** the publication date $\tau(e)$ of the document that contains the mention, and **(iii)** the textual distances $\delta(e)$ at which the terms or entities v and w cooccur in document $d(e)$. To normalize for the length of documents in the stream, we later only use the minimum distance $\delta(v, w)$ that v and w have in any given document, thereby reducing the number of parallel edges between the two nodes to at most one per document. Thus, in contrast to the implicit networks in Chapters 3 and 5, each cooccurrence of entities and terms induces at most one edge per document, not one edge per cooccurrence instance. This scheme allows us to easily integrate new documents, aggregate parallel edges on the fly, and provides a basis for exploring the evolution of topics. Therefore, each edge is attributed with a tuple $\langle d(e), \tau(e), \delta(e) \rangle$ that we then use to generate edge weights.

Edge attribute extraction

The extraction of edges works along the lines of the network extraction algorithm given in Chapter 3.3. To conceptually simplify the extraction of features and the representation of the graph, we pre-aggregate the observed edges (which correspond to individual cooccurrences) to a set of aggregated edges $\mathcal{A}$ that have set-valued attributes. Formally, consider a single aggregated edge $a = (v, w)$ that is derived from a set of parallel edges $\mathcal{E}_a$ with multiplicity $\lambda = |\mathcal{E}_a|$. Then, each such edge a can be regarded to have an associated list of tuples $\langle d(e), \tau(e), \delta(e) \rangle$ for each $e \in \mathcal{E}_a$, where the length of the list equals

λ. To directly extract the aggregated edges along with the associated lists of attributes, it is then sufficient to iterate once over each document. For the first cooccurrence instance of two words v and w in a document d, we add the corresponding edge to $\mathcal{A}$, as well as the tuple $\langle d, \tau(d), \delta(v, w)\rangle$ to that edge's list. If the same words cooccur again in the same document with a lower value of δ, we simply update the distance if necessary. If a cooccurrence of the two words is found in a new document, a new tuple is added to the list. Thus, we essentially keep track of where, when, and how often two terms or entities cooccur. In a sense, these lists then represent a time series of word cooccurrences that support subsequent explorations, and enable efficient updates of the network representation.

For the nodes of the graph, we store similar lists of tuples $\langle d, \tau\rangle$ as node attributes that we use to normalize the frequency of individual word mentions when generating edge weights. In the following, we discuss how substructures in the network can be associated with topics in the documents and how they can be identified by using edge weights.

6.3.2 Global edge weighting

Given the entity-centric structure of event descriptions as discussed in Chapter 3.4, it is a reasonable conjecture to assume that a high cooccurrence frequency of two entities is indicative of a topic. For example, in the case of news articles as document streams, interactions between politicians, parties, countries, companies, or other actors and locations all involve more than one entity, which supports an entity-centric exploration of topics. For a medical context and the interaction of drugs with diseases, a similar case could be made. This raises the question how such important relations between entities can be identified and filtered from spurious connections in the network on a global scale. Clearly, an edge with a higher frequency is more likely to be at the center of an important topic than an edge with a low frequency. However, frequency statistics are clearly not sufficient if we also want to consider a temporal scope and the evolution of topics, meaning that the model should have a temporal component. Based on these intuitions, we introduce a weight for the edges of the graph $\mathcal{G}$ that supports such a filtering, and includes both the overall and the temporal frequency of joint mentions, as well as the cooccurrence distances.

For an edge $a = (v, w)$, let $L(a) = (\langle d_1, t_1, \delta_1\rangle, \ldots, \langle d_\lambda, t_\lambda, \delta_\lambda\rangle)$ denote the associated list of attribute tuples, where λ denotes the multiplicity of the edge (that is, the number of parallel edges prior to the aggregation). Based on this list, we can reconstruct cooccurrence statistics. Let $D(a) = \{d : \langle d, \cdot, \cdot\rangle \in L(a)\}$ denote the set of documents in which v and w cooccur according to $L(a)$. Similarly, let $\mathcal{T}(a) = \{\tau : \langle \cdot, \tau, \cdot\rangle \in L(a)\}$ denote the set of timestamps at which both mentions that correspond to v and w occur jointly in a document.

Furthermore, let $D(v)$ and $\mathcal{T}(v)$ be defined analogously for a single node v. Finally, let $\Delta(a) = \langle \delta_1, \ldots, \delta_\lambda \rangle$ be the sequence of minimum distances at which the terms or entities that correspond to the two nodes of edge a occur in documents. Based on these statistics, we can define a weight of edges that incorporates the coverage of mentions in documents, as well as a temporal coverage and a similarity that is based on the cooccurrence distances. Due to this three-component scheme, we refer to this weight as $\dddot{\omega}$.

Definition 6.1 (Global edge importance $\dddot{\omega}$). Let $a = (v, w)$ be an aggregated edge with the edge attributes D, $\mathcal{T}$, Δ, and λ. Then the global edge importance is

$$\dddot{\omega}(a) := 3\left[\frac{|D(v) \cup D(w)|}{|D(a)|} + \frac{\max \mathcal{T}(a) - \min \mathcal{T}(a)}{|\mathcal{T}(a)|} + \frac{\lambda}{\sum_{\delta \in \Delta(a)} \exp(-\delta)}\right]^{-1}, \tag{6.1}$$

where $|\mathcal{T}(a)|$ is the number of time intervals on which the nodes v and w cooccur.

Note that the notation $|\mathcal{T}(a)|$ is feasible due to our choice of representing dates as integers. If a truly continuous model of time were to be considered, this discrete count could be replaced by any suitable density measurement.

Intuitively, $\dddot{\omega}$ as defined above represents the harmonic mean of three individual components, namely the number of joint versus individual mentions, the temporal coverage density, and an exponentially decaying weight by mention distance as introduced for the static implicit network. The resulting weight is normalized such that $\dddot{\omega} \in [0, 1]$, and the edge is undirected. Thus, it can serve as a measure of the global importance of an edge. Using these weights enables us to prune low-frequency edges and detect important entity connections, from which we grow and explore topics in the following. Since the components of the edge weights can be computed during network construction (or during network updates with new documents), no additional post-processing costs occur during the exploration of topics.

Most importantly, note that $\dddot{\omega}$ is applicable to any subset of entries in the list $L(a)$, meaning that it is applicable to any set of unaggregated edges between two nodes. Therefore, it is possible to select slices of the data and only compute edge weights for the corresponding subsets. For example, when considering a news stream from multiple news outlets and over multiple months, it becomes possible to consider topics for only one or a few outlets, or only for a selected time span. Given the rapid development of the news cycle, the latter is especially of interest, since it enables us to compare temporal slices that can be selected freely *after* the extraction of the network. Thus, it is possible to interactively focus on different intervals during the extraction of the topics, as we show in Chapter 6.4 and 6.5.

6.3.3 Topic construction and growth

Based on the derived edge weights, we now face the question of extracting topics from the implicit network. To this end, we argue that the core of topics is formed by edges between frequently cooccurring nodes, and that topics can be grown around such edges in a well-defined manner. Thus, we propose two growth approaches that specifically enable an interactive exploration of topics, and discuss the potential evolution of topics over time.

Assuming a non-increasing ordering of edges in $\mathcal{G}$ by weight $\ddot{\omega}$, the top-ranked edges then correlate to topic *seeds* as described above. Thus, we can select the top-ranked k edges for some value of k and treat them as seeds around which the topics are grown. To grow topic substructures around the selected edges, we introduce two types of growth patterns, namely *triangular growth* and *external node growth*. Note that some seed edges may share nodes, an aspect that leads to the *fusion* of two topics.

Triangular growth

Given an edge $a = (v, w)$ between entities v and w along with a network substructure that only contains a, v, and w, this initial substructure can be grown by adding neighbours of both entities, similar to the construction of evolving contextual topics in Chapter 5.4. Formally, recall that $N(v)$ and $N(w)$ denote the neighbours of nodes v and w respectively, then $N(v) \cap N(w)$ is the set of all nodes in $\mathcal{G}$ that share v and w as neighbours. To rank nodes in this potentially very large set, we utilize a scoring function on the edge weights. Specifically, let $\varrho_{vw} : V \to \mathbb{R}$ such that

$$\varrho_{vw}(x) := \min\{\ddot{\omega}(x, v), \ddot{\omega}(x, w)\}. \tag{6.2}$$

Obviously, nodes with a higher score cooccur more often and more consistently with both entities of the seed edge. Ranking nodes in the shared neighbourhood of v and w according to ϱ_{vw} thus allows us to select the most related terms to the topic that is represented by the seed edge (v, w). It is then a simple matter of adding any number of such term nodes to the network substructure (along with the edges connecting them to v and w), in order to incrementally grow the topic. Since all adjacent nodes can be ranked according to ϱ_{vw}, we obtain a relevance score for nodes in relation to the seed edge. In addition to the two seed words v and w, a topic can thus be interpreted as a list of ranked words that are added to the initial two words, based on their cooccurrence patterns, much like a classic topic model. However, this growth strategy also results in a descriptive network substructure as illustrated in Figure 6.1 (top).

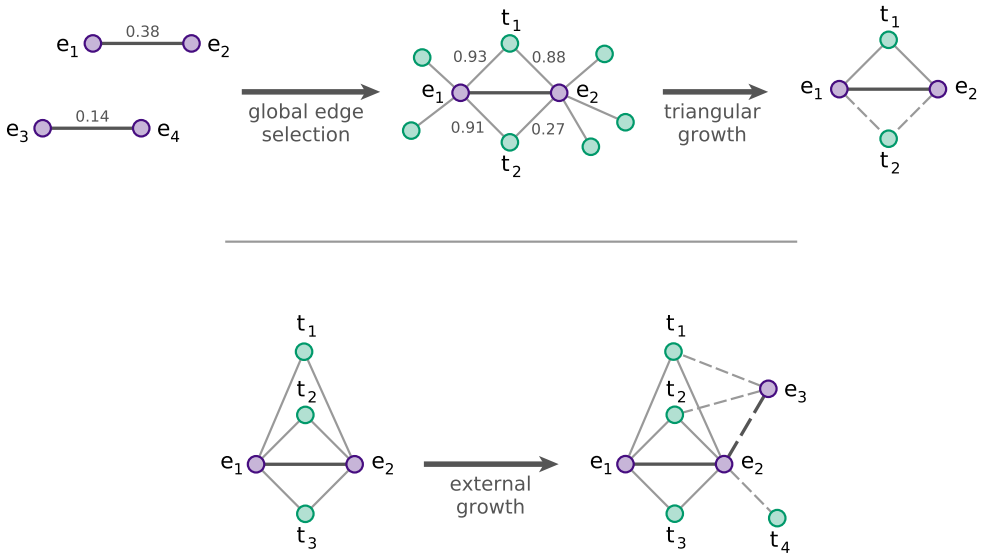

Figure 6.1: Visualization of topic growth with entities e_i and terms t_j. Top: seed edge selection and triangular growth. Seed edges of topics are ranked and selected based on their global edge weight $\ddot{\omega}$. From the set of terms that are connected to either entity of a seed edge, the most closely related are selected by edge weight, and used for triangular growth. Both the number of seed edges and the number of added term triangles can be selected or changed on the fly. Bottom: external growth of a topic. Entities or terms from the (joint) neighbourhood of any node can be added to grow the topic, for example based on edge weights or node type.

Based on the process described above, the incremental addition of words to the seed edge clearly supports different aspects of topic and cooccurrence exploration. First, instead of adding terms as described above, it is equally viable to select entities or even specific types η of entities in the shared neighbourhood. For example, if $\eta(v) = \eta(w) = \text{Loc}$, one could restrict the growth process to add only other locations, or instead only add persons. In a medical setting, this might be useful to restrict relations to subsets of entities that correspond to symptoms of disease and drugs, for example, which could offer insights into their complex relations beyond simply dyadic edges.

Depending on the technical realization and implementation of the storage that is used for the implicit network, it can also be used as an inverted index as we showed for EVELIN in Chapter 4.2. As a result, it becomes feasible to include functions that enable the user to inspect articles and sentences in which two words cooccur during the incremental construction and exploration of the network substructures. Thus, an interactive exploration of the topics in a collection or stream is viable, a possibility that we investigate in more detail for our topic exploration interface in Chapter 6.5.

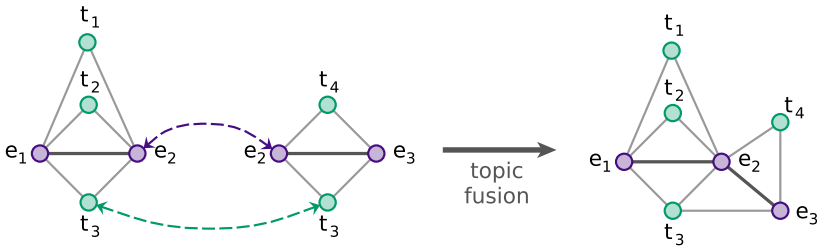

Figure 6.2: Visualization of topic fusion for two topic subgraphs that overlap on two shared nodes. Dotted lines denote the shared entity e_2 and term t_3.

External node growth

While the construction of edge triangles and term rankings is a key component in extracting and exploring a topic in the classic sense, the set of nodes that is determined in this way can also be explored with further expansion techniques. As a topic substructure grows, the seed edge and its incident nodes are not the only available attachment points for further edges and nodes. Instead, we can also add further nodes that are connected to only one of the initial words v or w, but also to some of the other nodes that were added in subsequent triangles. The external node growth process is also illustrated in Figure 6.1 (bottom), where an entity and a term are added to an existing topic substructure by dotted lines. Since these new nodes are not entirely connected to the seed nodes, we refer to this step as external node growth. While this attachment step has no analogy in classic topic extraction, it introduces additional degrees of freedom in an interactive exploration of topics, as we show in Chapter 6.5. Obviously, this addition of external nodes also leads back to the ranking and recommendation of nodes in the immediate neighbourhood of a given query node, which highlights the connection to the approaches that we discussed in the previous chapters.

Topic fusion and overlap

During the network-based extraction of topics using the steps described above, the substructure that is grown around the top-ranked seed edge is self-contained. However, this is not necessarily the case for substructures that are grown from subsequent edges in the list of top-ranked entity edges if $k > 1$ (that is, if more than the top-ranked seed edge is considered). Assume an edge $a' \neq a$ with $\ddot{\omega}(a') < \ddot{\omega}(a)$ from the list of seed edges. While new nodes are added to the substructure around a', it is possible that an edge is added that is incident to a node of the previously extracted substructure around the edge a. In practice,

this overlap between two topics may occur for term or entity nodes. In the most extreme case, even seed edges may overlap in one of their entity nodes, leading to the fusion of two topics. In Figure 6.2, we show an example of the fusion of two topics that overlap on one common entity and one common term. In classic topic models, the same word may belong to different topics with a high probability, which is analogous to partially overlapping topics in our model, where the same node can be part of different topics. In fact, we argue that topics *should* overlap for entities in news articles that participate in multiple topics. Consider, for example, a politician who meets with members of both her own and a foreign government to discuss matters of state with one group, and foreign relations with the other. In Chapter 6.4, we show how a network visualization can be used to highlight such overlapping substructures during the exploration of topics. However, unlike the fixed number of extracted topics in traditional topic models, the number of extracted topics dynamically adjusts itself to fit the data. If we assume that one connected subgraph describes one topic, then the number of topics is not necessarily equivalent to the number of selected seed edges. Instead, it adjusts dynamically with the number of related topics in the data, which is especially beneficial when the topics and their relations are visualized as graph structures.

6.3.4 Evolving topics and network projections

When the analysis of document streams is of interest, an important aspect of topic modelling is the evolution of topics over time [30, 98]. In such a setting, visualizations of the topics turn out to be especially helpful in highlighting changes in a topic's relevance over time, or in determining how it compares to other topics. The network-based approach for constructing topics as discussed above directly supports the exploration of temporal topic characteristics due to the timestamp information contained in the edge labels.

The key to the exploration of evolving topics based on an implicit network representation is the *network projection*. Generally speaking, a network projection filters nodes and edges that do not satisfy certain user-specified conditions. For example, a given network could be projected to only cooccurrences of entities, meaning that all term nodes and incident edges are removed. While such a drastic approach does not seem sensible, the reduction of the set of entities to a specific type, such as organizations, might be beneficial in some applications. However, for studying the evolution of topics, such conditions predominantly concern the timestamps that are associated with the cooccurrence information of edges. In principle, given a collection of documents spanning a time interval from τ_{min} to τ_{max}, the interval can be partitioned to construct corresponding networks. For example,

the evolution of news topics over multiple weeks or months can be considered by focusing on multiple networks that are constructed for each of these intervals. When multiple such snapshots are considered in sequence, the evolution of the topics becomes apparent.

For a seed edge in such a time interval, the user can then even directly compare respective sub-networks from different time intervals side-by-side, thus highlighting what nodes have gained or lost relevance for a given topic in a given interval with respect to neighbouring intervals. Clearly, there are numerous visualization metaphors that can be considered in this setting, all of them relying on the visualization of sub-networks. Central to all these approaches is the representation of topics in the (entity-centric) context of a network structure that explicitly represents the implicit word relations in the documents.

Of course, projections are also possible with regard to any other node attribute. For example, if the sources of documents are stored and classified, the topics that are discussed in different sources may be considered. In the following evaluation, we give examples of such projections for the exploration of the evolution of topics over time and among different outlets in a collection of news streams.

6.4 Comparison to Traditional Topic Models

To analyze the performance of the proposed entity-centric network topics in comparison to established LDA topic models, we again use streams of news articles. In the following, we briefly recap the data and describe the network extraction, before constructing network topics and performing a comparison to LDA topics.

6.4.1 News article data

To demonstrate the advantages of a network-based approach and investigate the evolution of topics, we focus on the extraction of topics from news articles that provide both a temporal component and a large scale. Specifically, the document collection should be sufficiently large, annotated for named entities, and have a temporal dimension that is represented in time stamps. As a rich source of events and numerous topics, news articles are well suited to this task. Therefore, we use the news stream collection that we already relied on for the evaluation of dynamic implicit networks in Chapter 5.5.

Recall that this data is collected from the RSS feeds of 14 English-speaking news outlets located in the U.S. (CNN, Los Angeles Times, New York Times, USA Today, CBS News,

The Washington Post, International Business Times), Great Britain (BBC, The Independent, Reuters, Sky News, The Telegraph, The Guardian), and Australia (Sydney Morning Herald). The data covers political news in the time frame from June 1, 2016, to November 30, 2016, and contains 127,485 articles with 5.4M sentences.

Data preparation

We again use manually created extraction rules for each news outlet to strip the HTML code and cleanly extract the text, before performing sentence splitting, tokenization, entity recognition, entity linking, entity classification, and stemming. For the recognition and classification of named entities, we rely on Ambiverse [93], which disambiguates entity mentions to Wikidata identifiers that we classify into locations, organization, and actors, based on the YAGO taxonomy (for details on the process, see Chapter 5.4). For sentence splitting and part-of-speech tagging, we use the Stanford POS tagger [200]. Since we focus on the evolution of topics in the context of the publication times of news, we do not annotate temporal expressions inside the documents for the exploration of topics. However, including temporal expressions as nodes in the network, is of course feasible and does not impede the extraction of topics.

Network construction

To construct the network, we use a modified version of the implicit network extraction algorithm (see Chapter 3.3) that we adapt to utilize disambiguated entities, and add document timestamps and outlet identifiers to edges. Terms are stemmed with a Porter stemmer [149]. We set the window size for the extraction of entity cooccurrences to $c = 5$. The resulting network has 27.7k locations, 72.0k actors, 19.6k organizations, and 329k terms, which are connected by 10.6 million edges, each of which is assigned a list of tuples as defined above. To generate edge weights for the resulting network, we use the weighting scheme $\ddot{\omega}$ described in Chapter 6.3.2.

6.4.2 Entity-centric extraction of topics

As the first step of our exploration, we consider the emulation of traditional topic models. That is, we focus on the extraction of lists of words with importance weights from the network model. Based on the underlying assumption that topics are focussed on entities, we first obtain a ranking by weight $\ddot{\omega}$ of all edges in the network that connect two

Beirut - Lebanon Q3820 - Q822		Russia - Moscow Q159 - Q649		Russia - Putin Q159 - Q7747		Trump - Obama Q22686 - Q76	
term	score	term	score	term	score	term	score
syrian	0.14	russian	0.28	russian	0.29	presid	0.40
rebel-held	0.12	soviet	0.06	presid	0.18	american	0.21
rebel	0.06	nato	0.06	annex	0.09	republican	0.19
cease-fir	0.05	diplomat	0.06	nato	0.08	democrat	0.19
bombard	0.05	syrian	0.06	hack	0.08	campaign	0.18
bomb	0.04	rebel	0.05	west	0.08	administr	0.17
territori	0.03	west	0.05	relat	0.08	polici	0.16
humanitarian	0.03	annex	0.05	sanction	0.07	state	0.16
citi	0.03	accus	0.05	leader	0.07	year	0.15
civilian	0.03	militari	0.05	admir	0.06	elect	0.15

Table 6.1: Emulated traditional topics as ranked lists of terms, extracted for the four top-ranked edges in an implicit network that is generated from the subset of all New York Times articles published between June and November, 2016. For each edge, the two incident entities and their Wikidata identifiers are given. The scores of terms are derived as the minimum $\ddot{\omega}$ values of the two edges connecting them to the entities of the seed edge. Topics are ranked in descending order from left to right by the weight of the seed edge.

entities. Thus, we utilize a global ranking of edges to identify relevant seeds for topics, which stands in contrast to the local entity-centric approaches that we have used on the implicit networks in previous chapters. The top-ranked edges are then considered to form the seeds of topics. Subsequently, each such seed edge is grown to a topic description by adding neighbouring terms that are connected to both entities as described in Chapter 6.3.3. Specifically, note that seed edges can only occur between entities, while triangular growth only occurs between entities and terms in our following analyses. Ranking scores are then obtained as the minimum edge weight between the term and both entities. Thus, the set of all terms that are associated with a seed edge can be used as a topic in which the weight of terms towards the topic is determined by the term ranking. Of course, the resulting list can be trimmed according to some threshold if it is necessary to bound size of the topic. Since only the local neighbourhood of the two entities is considered after the initial global ranking is obtained, this process is extremely efficient, can be parallelized by edge, and computed at query time to obtain, expand, or reduce an arbitrary number of topics interactively.

As an example, we show the topics that are induced by the four top-ranked entity edges in the subset of articles from the New York Times in Table 6.1. Considering the results, we find that the topics are overall descriptive and can be interpreted within the context of news in 2016. With regard to location mentions, the example shows a rather prevalent bias in news articles, which often include the location of the correspondent or news agency at

the start of the article (that is, articles about the war in Syria are often not reported from Syria itself but from neighbouring Lebanon). Thus, while this edge describes the war in Syria as a topic quite well, the seed edge itself is partially an artefact of the named entity recognition step. Similar artefacts occur for some outlets that reuse content from third parties and begin an article with the name of the news agency that initially broke the news. While such artefacts can never be entirely avoided, they can likely be minimized by adding further rules for the extraction of article contents. Alternatively, if topics that are focused on such synonyms (for example, the mention of a capital in place of a country) are not of interest, filtering edges by entity type is easily possible.

A second interesting aspect is the overlap between the second and third topic on the entity node of Russia. While this is a case of overlap between seed edges, the resulting topics are still descriptive and nuanced. Here, it makes sense to argue that the topics are related, since they overlap in their seed edges as well as some terms, but not identical. The second topic is associated more strongly with Russian military operations, while the third topic covers aspects of Russian politics and the involvement of the Russian president. However, despite these nuances, it is obvious that the two topics are related, but discerning their relation is neither trivial nor obvious. Here, visualizations of the network representations of the topics can be helpful tools as we show in the following.

Overall, we find that the topics and topic qualities vary strongly by news outlet, much like traditional topic models. Since the topics of each edge are independent, it is easier to discard unwanted topics than it would be for traditional topic models with interdependent topics. However, if the extraction of traditional topics is necessary or wanted, we find that we can replicate such list-based topics with this edge-centric network approach.

6.4.3 Network topic extraction and exploration

In contrast to the extraction of lists of terms, we can also fully utilize the network representation to extract complex topic substructures. Instead of lists of terms from nodes that surround seed edges, we extract the nodes themselves to continually grow a descriptive network structure. Conceptually, we proceed in the same way as described for traditional topics above, by extracting a ranked list of entity-centric edges and selecting the top-ranked edges. For each edge, we then include a number of terms that are adjacent to both entities in the network. Instead of retrieving the terms as a list, however, we directly visualize the resulting subgraphs. The number of adjacent terms per seed edge can be selected arbitrarily, but a value between two and five term nodes per edge tends to result

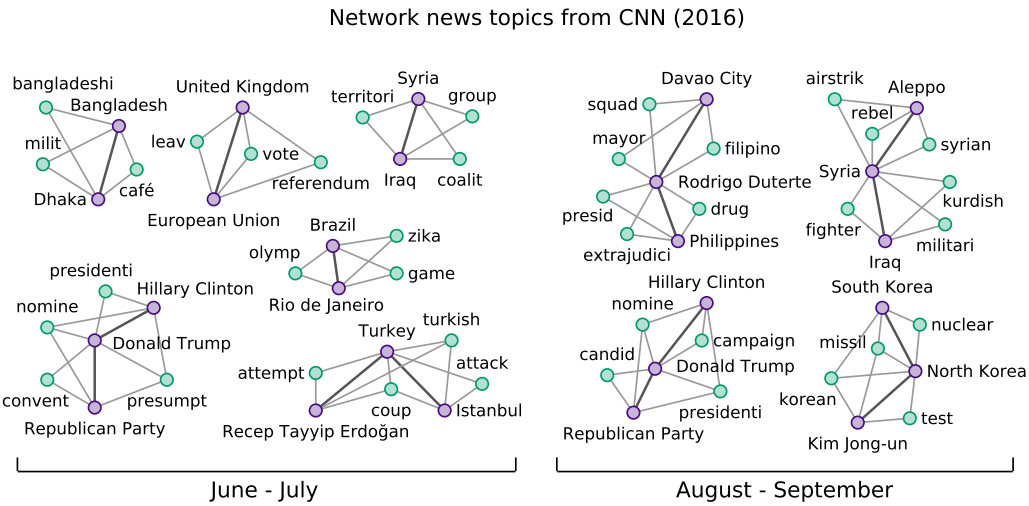

Figure 6.3: Comparison of the topic substructures and topic evolution of entities (purple) and terms (teal) between two time slices for the article subset of CNN. Shown are the eight highest ranked edges and the three most relevant connected terms.

in a visually interpretable network that is not too cluttered, while still being descriptive enough to provide an insight into the topics' contents.

As discussed in Chapter 6.3, by projecting the network according to the publication date of the corresponding article, we can introduce a temporal dimension and investigate the evolution of topic networks over time or even dynamically for selected intervals. Since the temporal information is an integral part of the network, this selection can be changed dynamically by the user. Similarly, if outlet identifiers are stored alongside document identifiers, we can create projections to a part of the network that corresponds to articles published by a selected news outlet or subset of outlets.

Temporal topic comparison

In Figure 6.3, we show two such temporal snapshots for the subnetwork of articles from CNN as a prototypical example. The results for other news outlets are similar, albeit with a regional or political bias that depends on the news outlet. While the network supports the extraction of topics from all articles of a collection simultaneously, such a restriction to particular news outlets can provide facets to the exploration. In the graph visualizations of the network topics, we still find the same descriptive terms as we do in the case of ranked term lists, but we also observe the additional structure of the underlying network. Unlike term lists, which represent isolated topics that are only implicitly linked by

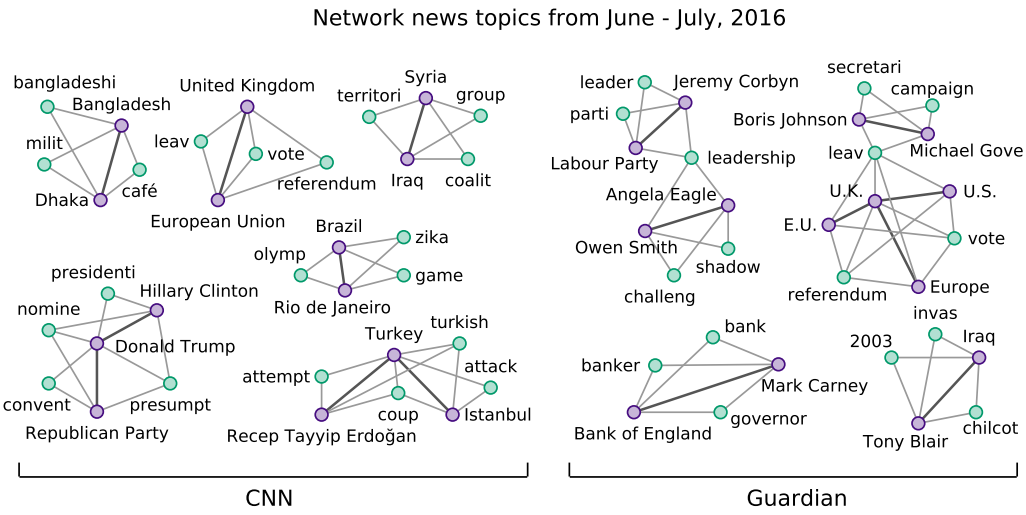

Figure 6.4: Comparison of the topic substructures of entities (purple) and terms (teal) for news articles from the news outlets CNN and The Guardian in Summer 2016. Shown are the eight highest ranked seed edges and the three most relevant connected terms per seed edge. Due to topic fusion, the number of connected components is less than eight, and we observe fusion of topic subgraphs on both terms and entities.

terms that occur in multiple topics, the overlaps of edges show topic relations directly. In fact, we observe fused subgraph structures that emerge from the top-ranked edges and lend further support to their topics. For example, the topics of Trump, Clinton, and the Republican Party are clearly related, as is evident from both the direct connections and the related terms. On the temporal axis, we find that the topics correlate well with political events. For example, the Brexit topic disappears after the date of the referendum in June 2016 (of course, it is more pronounced in British outlets), while several war-related topics shift in focus to follow ongoing campaign locations in the war against the Islamic State. Expectedly, the topic covering the drawn-out process of the U.S. election is stable over time. Overall, we find that the network representation adds a structure to the visualization that is easily recognizable and explorable, thus allowing the user to observe changes in the topics more easily. Obviously, such changes could be further visualized with network animations that highlight the changes over time.

Comparison of topics by news outlet

In Figure 6.4, we show a contrastive comparison for the same time frame, but between the topics that are extracted from two different news outlets. Here, we observe a clear bias in the importance of topics that is caused by the location of the news outlet. CNN,

which is based in the United States but concerned with giving an overview of global news, covers a wider range of topics, including the Brexit discussion. The Guardian, on the other hand, which is a British newspaper, is clearly more focused on this regional topic that is potentially of greater importance to its readers. Upon closer inspection, the snapshot of topics from The Guardian contains two topics that are related to Brexit, which are likely to fuse once more seed edges are considered. As in the case of temporal projects, the focus on news outlets allows us to quickly compare what is being talked about in a given news outlet or region (if multiple outlets are grouped) in comparison to other outlets or regions. While such a comparison is certainly possible even for lists of terms, it would be less intuitive and less visual.

6.4.4 Comparison to LDA topics

As a final step of our investigation into network topic models, we consider a direct comparison to the output of graphical models. However, it is well-known that topics are subjective and a strict evaluation is difficult, especially for an exploratory approach. This issue is further compounded by the fact that there are no suitable entity-annotated data sets with topic ground truth. Therefore, we again use the news article data set and extract topics as ranked lists of words with both network topics and a traditional topic model.

Comparison metric

As a metric for the comparison, we compute the coverage of topics to capture how well each of the topics that are produced by one approach is reflected in the topics that are produced by the other.

Definition 6.2 (Coverage). Let X and Y be two sets of topics of size $k = |X| = |Y|$. Then we define the coverage of X in Y as

$$coverage(X, Y) := \frac{1}{k} \sum_{x \in X} \max_{y \in Y} \frac{|x \cap y|}{|x \cup y|}. \tag{6.3}$$

While the coverage as defined above is based on the Jaccard coefficient of individual topics, note that it is not symmetric. Therefore, we consider it for both directions in the following. Intuitively, for each topic in set X, we find the best matching topic in set Y, so that the resulting score gives us an indication of how consistently topics from one set are covered in the other set.

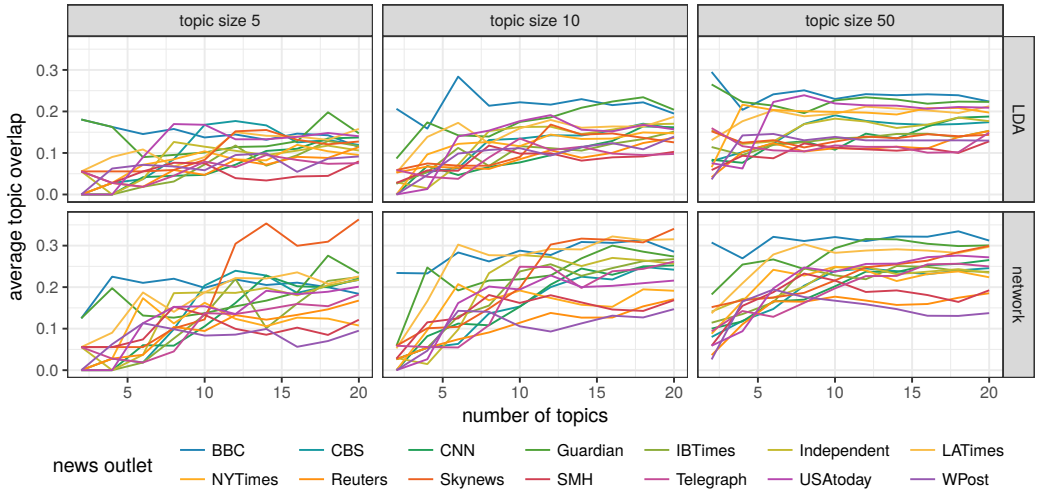

Figure 6.5: Comparison of coverage between network topics and LDA topics. Topic size denotes the number of words per topic. Results are shown for the coverage of LDA topics in network topics (top row) and network topics in LDA topics (bottom row).

Comparison setup

To relate network topics to traditional topic models, we compare their list-of-term representation to LDA topics [31] as a prototypical representative. For network topics, we extract topics for each news outlet from the network as described in Chapter 6.4.2. Since entities are a major component of network topics, we also add the tokenized labels of seed nodes to the term lists before selecting the top-ranked terms of a network topic. Otherwise, since the labels would be missing from the final network topics, while they would still be contained as terms in the LDA topics. For example, if a seed edge contains the node of entity Q67 (*Barack Obama*), we generate the terms *Barack* and *Obama* and add them to the topic. For the LDA topics, we group the news articles by outlet, prepare the plain text with Tidytext [169], and generate topics with the R implementation of LDA [81]. We iterate over the number of topics k for both approaches. However, note that we technically only extract the network topics, once since they can be computed incrementally, and then selected the k highest ranked topics as required.

Comparison results

In Figure 6.5, we show the comparison results for all 14 news outlets in the document collection. Here, the performances for the document collections that correspond to individual news outlets are similar and less interesting than the overall trend. Following this trend,

we find that the coverage tends to increase with the number of topics and with the number of terms per topic. However, the coverage of LDA topics in network topics is much worse than the other way around. Combined with the increasing coverage for larger numbers of topics, this indicates the more narrowly focussed nature of network topics, meaning that they cover more diverse and fine-grained issues, not all of which are reflected in the LDA topics. The coverage of LDA topics in network topics increases with the number of topics, since this increased number narrows the scope of each individual LDA topic. In summary, we find that network topics seem to be well reflected in LDA topics once the number of extracted topics for LDA is large enough to distinguish between the multitude of news topics. However, for a number k of topics beyond 20, the runtime of LDA becomes a serious issue on the larger news outlets. This is due to the fact that the number of topics is a multiplicative factor in the asymptotic runtime complexity of LDA, while it is only an addend in the complexity of network topics. Of course, even the extraction of 50 topics is unlikely to reflect the amount of topics that occur in the news over a span of six months. Thus, using traditional topic models is challenging in such a setting. In contrast, the number of network topics can be dynamically adjusted during the exploration and supports settings where the repeated extraction of traditional topics is too compute intensive.

Based on these observations, it seems reasonable to apply entity-centric network topics in a setting where traditional topic models do not fare as well. In the following, we therefore consider the interactive extraction, visualization, and exploration of topics from news streams as a Web application.

6.5 Interactive Topic Exploration

Based on the advantages that implicit networks offer in a more visual representation of entity-centric topics in unstructured text, we further investigate novel and interactive ways of exploring entity relations in document collections or document streams. Where EVELIN as presented in Chapter 4.2 focussed on individual entity relations, we now focus on small substructures that arise from these relations, and show how they can be manipulated to retrieve information from the documents. In the following, we thus present TopExNet as a tool for Topic Exploration in implicit Networks of news streams, which is available as a Web demonstration[1].

Attribution. This section describes joint work. The Web-frontend of the user interface was implemented by a collaborator.

[1] TopExNet is available online as a Web tool for searching our news stream data set from 2016, along with a user's manual: http://topexnet.ifi.uni-heidelberg.de

6.5.1 Why use entity-centric topic visualization for news?

As we have argued in the motivation for dynamic implicit networks, keeping up with the news is no easy task. Due to the constant deluge of articles that are published non-stop in the global news cycle, finding relevant pieces of information can be such a daunting task that many users resort to reading nothing but headlines. On the other hand, news publishers break down many articles into small bits to advertise for their articles with prominently displayed reading times of as few minutes as possible, but inadvertently increase the number of published articles even further.

As a result, an automated approach to the aggregation of news is clearly beneficial, but no less daunting from a computational perspective. While much research has been devoted to approaches for finding and linking incidents in news [59], the problem is far from trivial and too restrictive in purely exploratory settings. Intuitively, topic models [28] should offer a solid solution to the extraction of relevant topics from collections of documents. However, as we have already seen in the previous section, their performance tends to suffer on large collections of news articles with a multitude of diverse topics, and they are ill-suited to the interactive exploration of documents. In addition to the complexity, the character of topics as lists of terms that are hard to interpret is also ill-suited to a setting in which users want to obtain information at a glance.

In this respect, implicit networks fall into a recent shift in focus towards network-centric representations of documents that stands to provide more intuitive and more *visual* access to the complex relations contained in the texts. Beyond the exploration of implicit networks, examples of this approach also include the proposed use of concept maps as summaries instead of text snippets [57]. In the following, we therefore focus on using network extraction and visualization to identify intuitive topics as network structures of entities and terms. In contrast to traditional topic models, this results in a more dynamic exploration of topics that can be used in place of aggregation and browsing approaches for incidents or articles, such as Google News.

6.5.2 Related work on topic and news exploration tools

Before we describe the tool itself, we briefly discuss related work that covers applications with a similar purpose, and can be partitioned into two areas, namely topic visualization and news exploration tools.

Topic visualization

Due to the difficulties of interpreting topic models, some effort has been put forward to make them more intuitive and visual. Thus, there are some applications that support a visual and interactive analysis of topics. One such example are TopicNets [77], which allow the user to view the contents of document within the larger scope of overarching topics. Similarly, word network topics are designed for the discovery of topic relations in short texts [231]. In contrast, the Topic Browser focuses on topics as lists of words, but enhances the interpretability by word cloud style layouts and the visualization of statistics in one combined package [66].

In contrast to our approach, these applications lack a focus on entities. Since entities serve as the anchors of events in news texts, they should be included as an integral part of any model for the analysis of news articles. However, none of the above tools are designed for an analysis of news streams.

News exploration

On the other hand, there are several tools for the exploration of news streams or collections. For example, STICS is a versatile tool for browsing news with a focus on entities [95]. Similarly, EventRegistry enables the exploration of news events through involved entities [114]. NewsStand takes a different approach by embedding and clustering news articles geographically [199]. Further approaches include the monitoring of multilingual European news [13], and the extraction of semantic word clouds with significance analysis to obtain a quick overview of current news [163]. In contrast, other approaches do not work on news articles, but are designed for raw sources as input and journalists as users [226]. However, none of these news analysis tools include an exploration of topics, which are central to obtaining a glance at what is going on in the news. In the following, we therefore focus on providing a tool that offers this functionality.

6.5.3 Application architecture

We give a brief recap and overview of the used model and data for the construction of a network representation of the news stream that we use in the following. Based on the model, we describe the system architecture that we use for the extraction of network topics, and compare it to the architecture of interactive systems that rely on graphical models.

Implicit network model

As implicit network representation, we utilize the model that we introduced in Chapter 6.3. However, for the ease of representing the data on secondary storage devices, we use a slightly different physical representation when compared to the initial model. In particular, let us assume that we can store the unaggregated edges as tuples $e = \langle v, w, \tau, out, d, \delta \rangle$ for an edge between nodes v and w that occurs in a document d that was published by a news outlet *out* on date τ. This structure then allows us to directly use existing (relational) databases for storing the edges, along with the benefits of being able to use indices on v and w to identify specific edges, as well as τ and *out* for the selection of network projections. To support efficient queries over varying selections of news outlets and date ranges, we furthermore aggregate edges partially to at most one edge per entity pair, publication day, and outlet. Node statistics, such as the occurrences of terms and entities in the documents, are partially aggregated in a similar fashion.

Based on the weighting function $\ddot{\omega}$ introduced in Equation 6.1, it is then a simple matter to obtain a ranking of edges for any selected subset of edges by date or news outlet. Thus, we can construct edge weights for any subset of our data at query time to select seed edges that correspond to topics in the given subset. In the application, entity edges can then be instantiated either by selecting the globally top-ranked edges for a time interval and selection of outlets, or by directly specifying pairs of entities that are of interest to the user.

Data preparation

As data for the application, we again want to focus on the exploration of news streams. Therefore, while any collection of time-stamped documents with annotated entities could be used in principle, we use the news data set from Chapter 5.4 and prepare it according to the description in Chapter 6.4. Similar to the preparation of data for EVELIN in Chapter 4.2, all entities are augmented with short descriptions from Wikidata that can be displayed to the user during query entity selection.

Architectural advantages

In Figure 6.6, we show an overview of the architecture and a comparison to applications that rely on traditional topic models. The entity-centric network approach, as visualized in the top branch, works similar to the entity ranking application we discussed in Chapter 4.2. The input data is passed through a natural language processing pipeline to annotate

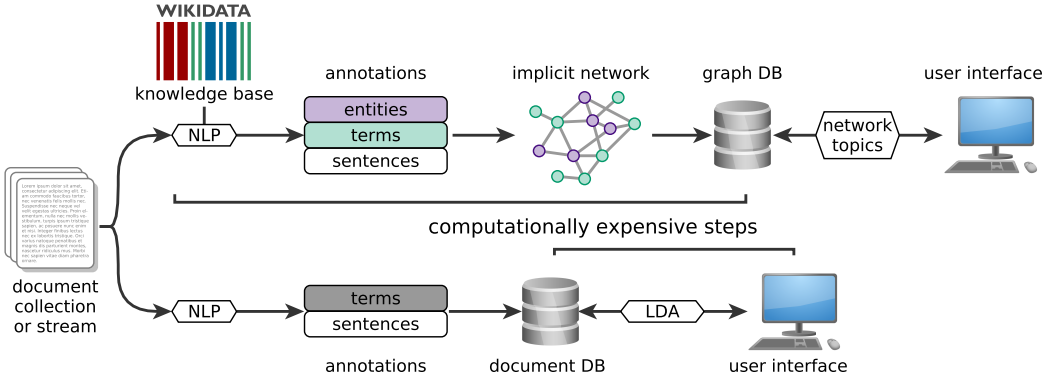

Figure 6.6: Schematic view of the data processing pipeline and system architecture for the network-based topic exploration (top branch) in comparison to an architecture that is based on LDA (bottom branch). Note the different stages of the computationally expensive steps that make an exploration based on LDA difficult. For the implicit networks, we use Wikidata as the knowledge base for entity linking, but the process can be applied with the support of any knowledge base.

and link named entities to a knowledge base. From the annotations, an implicit network representation is constructed and stored in a database. Any query to the network is then run against this database. As discussed in Chapter 6.3, this means that projections, slices of the data, and varying numbers of topics can be selected from the network at runtime. Therefore, all computationally expensive steps are covered in the preprocessing phase, which allows a dynamic adjustment of the topic extraction parameters for an interactive user experience. In contrast, the lower branch shows a typical setup for the extraction of graphical topic models. Here, the computationally most expensive step is the extraction of the topics themselves. Since topics cannot be adjusted dynamically in size, or be limited to a subset of articles on the fly, it is necessary to run the extraction process again whenever the parameters are changed. For document collections in the order of magnitude that we consider here, such an extraction of LDA topics can take several hours on modern hardware, thereby rendering this approach infeasible for an interactive exploration of large collections. As a result, traditional topics would have to be extracted in advance and only visualized afterwards. In this, we see the reason for the limited use of topic models in the interactive and real-time exploration of news, where descriptive topics stand to benefit news consumers the most.

Data processing layer

Once the network representation is extracted, the majority of the work is performed in the data processing layer that links the database to the user interface. Here, we use a

straightforward architecture that utilizes secondary storage in a database. While an in-memory representation of the data is possible and would support faster query processing than secondary storage, it does not scale well and is not feasible for a long-running non-commercial demonstration, as we have argued in Chapter 4.2. Therefore, we use a Core i7 with 32GB main memory and an SSD drive as our demonstration server. The network is stored in a MongoDB, with separate collections for entities, terms, edges between entities, and edges between entities and terms. Entities are enriched with Wikidata information to provide entity descriptions at query time. A text index on the English canonical label is used for compiling a list of entity suggestions to the user from input strings. We rank entity suggestions by the text match score and break ties by the number of occurrences. All edges are partially aggregated at a granularity level of days to speed up subsequent aggregations at query time. Additional node occurrence information is stored in a similar aggregation to retrieve the individual occurrence numbers of nodes in the documents.

The topic extraction methods described in Chapter 6.3 are implemented in Java and enable query processing speeds in the order of a few seconds for all but the most highly connected entities. While topic extraction queries can easily be parallelized by assigning one thread per seed edge, we only allocate one thread per query to serve queries from multiple users simultaneously. To avoid system overload in the case of multiple users, we use an internal anonymous mapping of queries to browser fingerprints, and limit the number of active queries per user.

A potential bottleneck are dense regions of the network. Like most other complex networks, implicit networks have a long-tailed degree distribution, which translates to the presence of a few highly connected hubs in practice. While queries to the network generally benefit from the overall sparseness, hubs may cause longer query response times for incident edges, especially for larger date ranges. However, due to the small number of such hubs, this problem can be addressed by a caching of results in the database, which ameliorates the effect over time. Specifically, to ensure adequate query response times and avoid spikes in the query duration, we use a separate database collection for caching the results of triangular term expansion queries with long response times, which then serve as building blocks for later queries that include the same entities. Based on the intuition that frequently searched entities likely correspond to frequently mentioned entities in the documents (meaning that they represent hubs in the network), it makes sense to implement caching strategies specifically to ameliorate the query response times for these entities. In practice, we use a buffer for individual seed edges. That is, even if multiple seed edges are requested at the same time, they are cached independently. Testing valid cache replacement strategies is difficult since one does not run into realistic scenarios that involve cache

size problems in a scientific demonstration. However, based on our observation about network hubs, we argue that a least-frequently-used replacement strategy is likely to work well. Seed edges that are rarely requested are also less likely to occur between highly connected nodes, thereby reducing the effort needed to recompute their local rankings, meaning that no caching is required.

Presentation layer

To implement the web interface, which serves to accept user input for extracting suitable topics and visualizing the resulting output as graphs, we use a combination of HTML and JavaScript. For entity input and for sending queries to the application layer, we use jQuery. The Bootstrap libraries [33] and Mustache web templates [210] are used for the interactive layout. To recognize and classify input entities, we use the tags-input and typeahead libraries of Bootstrap, which we extend by adding the required functionality for the color-coding of entities. The interactive visualization of the topic networks and the menus is handled by the vis.js JavaScript library [205]. The topic subgraphs are then visualized with a force-directed layout.

The web server itself uses the Java Spark micro framework [213] and is directly integrated with the data processing layer. Communication between the user interface and the server is built on Ajax and uses JSON for transmitting data in both directions (that is, input query entities and output graph data). The user interface thus functions as a website, based on which we discuss the application scenarios in the following.

6.5.4 Application usage scenarios

TopExNet supports three primary modes of exploration, namely global edge ranking, targeted entity explorations, and the exploration of topic networks. We begin by describing the parameters that are available to the user.

Input parameters

The user can input data for queries and limiting constraints based on the five parameters *entities*, *date range*, *news outlets*, the *number of edges* and the *number of terms*. Additionally, a subset of news outlets can be selected. All input choices that we discuss in detail in the following are visible on the initial input screen shown in Figure 6.7.

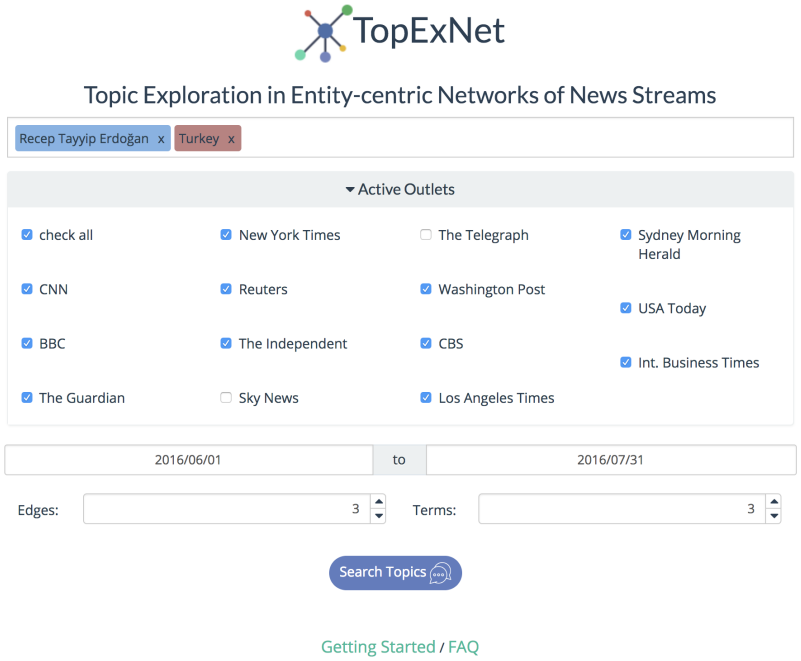

Figure 6.7: Initial view of TopExNet's start page with the input field and all selectable query parameters. Input terms can be types to entities, similar to EVELIN. The user may select a subset of news outlets and a time frame of interest. Furthermore, the number of seed edges for the initial extraction and the number of terms per seed edge during triangular growth are selectable.

The most central components, namely *entities*, are entered as input by selecting them from a list of suggestions that is generated from one or several strings entered by the user. This works identical to the interface for EVELIN and is implemented in a similar manner. Short descriptions and color-coding by entity type in the suggestion menu help to guide the user to a selection. Upon selection, entity suggestions are automatically linked to the corresponding network nodes. As a second parameter, a *date range* allows temporal slicing of the document collection and is selected by using a date range picker with a granularity of days. For practical reasons, we impose a minimum date range of three days to ensure a sufficient number of articles in the interval. The date range is automatically limited to the publication time frame of the article collection. Similar to the selection of a date range, the user may also select a subset of articles by news outlet, which is implemented as checkboxes. By default, all outlets are considered. The *number of edges* can be used to set the number of seed edges when global edge ranking is used as a starting point. Finally,

the *number of terms* can be adjusted, which sets the number of neighbouring terms that are extracted for each seed edge during triangular growth.

Global edge ranking

The first primary use-case is the automatic extraction of seed edges from the network or from a projection of the network to obtain an overview of the most important contained topics. This type of query is activated by specifying a time range, a selection of news outlets, the number of seed edges, and the number of descriptive terms, but no input entities. As output of the query, TopExNet then retrieves a global ranking of all entity edges for the specified date interval and news outlets. The top-ranked seed edges are selected as topic seeds, fused if they share some of their nodes, and expanded by adding descriptive terms in triangular growth steps. An example of the output for four seed edges and three descriptive terms per seed edge is shown in Figure 6.8 (bottom). Due to the background caching, when the number of edges and terms is adjusted on the result screen, the cached results are updated instantaneously.

Targeted entity exploration

In contrast to the global ranking of edges, which focusses on the topics surrounding the entities that are the overall most central during the selected time frame, the user may also focus on selected entities. When two entities are entered as the input of a query, TopExNet focuses on the selected entities and performs the triangular growth step by adding descriptive terms only to the edge between the two provided entities. An example is shown in Figure 6.8 (top). While such individual seed edges naturally generate smaller topic subgraphs, they can serve as the basis of further queries by expanding them in an adjacency exploration step.

Topic network exploration

Independent of how a topic subgraph was generated in either of the above two scenarios, it supports further explorations as a starting point. In addition to the obvious tuning by increasing or decreasing the number of seed edges and descriptive terms per edge, the user may also expand the network by adding further nodes. When selecting any *single* entity in the network, the user is given the choice to include top-ranked adjacent nodes for the selection. To realize this functionality, we rely on the entity ranking method that

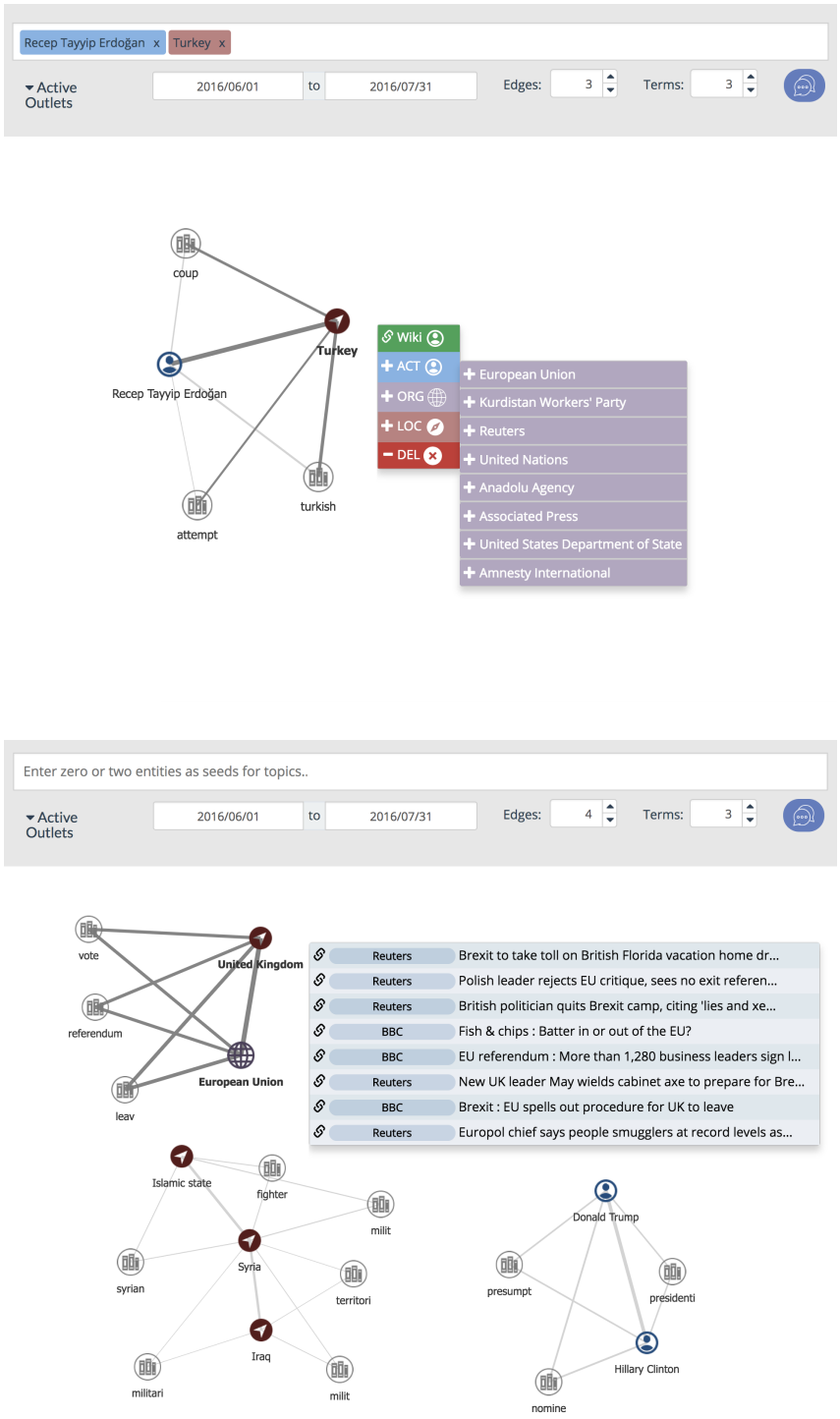

Figure 6.8: Overview of the output of TopExNet, along with further exploration functionality. Top: initial search result for the topic surrounding two explicitly specified input entities. The context menu for external growth by entity addition is opened. Bottom: initial search result for the top four seed edges in a time frame. The context menu for external news article prediction is opened.

we introduced for EVELIN in Chapter 4.2, and adapt it to take the semi-aggregated edges of TopExNet as input, aggregate them, and perform the ranking. The top-ranked neighbouring entities are displayed in a menu, where they can be selected and placed on the canvas. Upon placement, they are automatically connected to the initially selected entity. A visualization of the process is shown in Figure 6.8 (top). Once additional entity edges are introduced, descriptive terms may be included by clicking on seed edges or the canvas and selecting the option to add terms. In a similar fashion, nodes that are not of interest can be deleted. Alternatively, it is possible to obtain information about all displayed entities by following a link to Wikidata, thereby directly integrating the knowledge base.

Once the user has obtained an overview over the existing topics, they might be interested in some of the topics and would like to read original sources. To enable this functionality, we again adapt the neighbourhood search from EVELIN to include document recommendations. If the user selects *multiple* entities on the canvas, right-clicking now opens a recommendation menu of news articles that are relevant to the selected entities. The displayed URLs then link directly to the original source articles. An example of such a recommendation is shown in Figure 6.8 (bottom). As a result of this recommendation feature, the system is well suited to providing an overview of news without having to display proprietary content from the original sources. Instead, the user is presented with an aggregate view until he is redirected to an original source.

Overall, all three approaches enable a contrastive analysis of the results for different date ranges or news outlets. By simply comparing the network topics of different projections in parallel browser windows, it becomes possible to investigate the differences between news outlets, or the evolution of topics over time.

6.6 Summary and Discussion

The detection of topics in a collection of documents is a central task in corpus analysis, in particular for document classification and clustering. While there exist many approaches for topic modelling, they often fall short in terms of intuitive, interactive, and efficient topic exploration methods, especially for dynamic collections and streams of documents. We introduced entity-centric network topics as a solution to this problem, and to provide a novel framework for the exploration in such document streams, which we tested on news articles. We find that this model not only addresses the need for entity-centric topics beyond lists of words, but also provides support for an interactive exploration of topics. Due to the inherently dynamic nature of implicit network representations, it is a

simple task to update the underlying network whenever documents change or are added to the collection or stream. Most prominently, the partial aggregation of entity and term cooccurrence edges allows the efficient retrieval of seed edges and descriptive terms from the collection without the costly requirement of recomputing topics for the entire corpus due to changing parameters. Furthermore, it stands to reason that this representation also supports the sharding of queries and stored data by news outlets or date between multiple compute and storage nodes, which ensures scalability for larger document collections.

While we did not use an actual live stream for our experiments, but instead emulated a stream from a prepared collection of news articles, this is motivated by the limited availability of open source named entity linking tools with stream support. In principle, as long as a natural language pipeline is available that can process documents in a news stream, the output can be fed directly into an implicit network model, thereby allowing us to obtain near-real time updates. As a result, we see applications for this approach in the preprocessing of news or blog-style articles for news aggregation websites. By providing network topics instead of summaries or text snippets, the user stands to obtain an overview of what is currently going on in the news more easily. On the legal side, since we do not use text snippets but generate graph visualizations from the aggregated content, there is no need to publish proprietary content. Thus, the approach can easily be implemented in a Web setting to satisfy current information needs that are not adequately addressed by existing news aggregation techniques.

Practical implications

For the data analysts and researchers in our running example, the extraction of entity-centric topics provides novel visualization capabilities. However, this visualization is not limited to static entity relations, but in fact provides them with a window into the evolution of entity relations. The researchers can now view the network topic representations of different documents side by side, compare the relations of entities in different sources for potential biases, or even discover changing affiliations.

A particularly useful aspect of network topics in such investigations is the scalability of discovering globally important edges. Due to the size of such document collections or streams, it can be hard for the researchers to obtain a quick overview of new documents and determine where to start. With the global importance ranking of relations in network topics, however, they can uncover important relations almost immediately. The inclusion of the surrounding terms in the network topics then provides context, while entities in the neighbourhood provide directions for possible investigations. Especially for continuously

growing document collections, even those in which new documents arrive in batches, this helps the investigators to determine how the contents of new documents can be integrated into the collection, or how the contents differ in comparison to previous documents, in order to quickly spot new angles for the investigation.

For the analysts, the topic-based search interface is an important tool that not only provides near real-time topic retrieval capabilities, but also integrates with the entity-centric search capability of static implicit networks. The bridging of the structured data of knowledge repositories with the unstructured texts makes knowledge base information about selected entities available at a keystroke. Once central topics are identified, the links to the source documents in which the topics are discussed provides investigators with the material to follow up on the subject.

Outlook

A final point in regard to network topics that is worth considering is the generality of the representation. While we used slightly different edge representations when compared to EVELIN in Chapter 4.2, the two representations are compatible. Thus, it is possible to combine both approaches, as we have already seen on the example of the entity ranking queries that we included in TopExNet. However, we have so far used specialized graph representations of implicit networks with distinct applications in mind, and not capitalized on this potential. Thus, a more general modelling framework is conceivable that provides a unified view on implicit networks, which we discuss in the following chapter.

7 A Versatile Hypergraph Document Representation

In the previous chapters, we have introduced implicit networks as a representation of documents that supports a variety of entity-centric retrieval tasks. While all of the applications that we presented thus far are based on the extraction of implicit networks as a common core, we have considered specific aspects of the model separately, and we have focused on pairs of entities. In the following, we go a step further and generalize the underlying notions of implicit networks in a joint framework that captures not only the cooccurrences of pairs of entities, but the cooccurrences of arbitrary sets of terms that can be attributed with any external source of knowledge.

Contributions. In this chapter, we thus make the following four contributions.

I We introduce a hypergraph-based document model for describing higher-order cooccurrences as a generalization of implicit term cooccurrence networks.

II We design the model to jointly represent textual data alongside annotations and links to corresponding entities in knowledge bases, thus bridging the gap between structured and unstructured data.

III We propose fundamental graph operators for filtering and selecting nodes and (subsets of) hyperedges to support a wide range of applications.

IV We demonstrate how the document model can be used in combination with the proposed operators to realize central concepts and tasks from information retrieval in a unified framework.

7.1 Motivation

As we have seen in the previous chapters, the choice of a good document model is bound to have a substantial impact on an application's performance in information retrieval and extraction tasks. But what *makes* a good document model? The answer to this question, of course, heavily depends on the context in which it is posed, and numerous document models have been proposed to address this need, as we have discussed in Chapter 2. First and foremost, different applications come with different information needs that have to be taken into account. When these applications are as diverse as summarization, event detection, search, or document similarity and classification, to name but a few [125, 227], identifying a common denominator is no simple task. However, a closer inspection of these IR tasks reveals that the concept of word dependency frequently plays a central role, which we have also encountered in the context of collocations [56], topic models [28], or word embeddings [22] throughout the previous chapters. As a result, word cooccurrence statistics and word relations tend to be a central component of document models, even if they are modelled as graphs [26, 44, 119]. Of course, the implicit network model that we have introduced here also shares this common root of word cooccurrences.

At a fundamental level, however, whenever these cooccurrences of words, terms, or entities are modelled as entries in a cooccurrence matrix or as the edges of a graph, they essentially constitute *pairwise relations*. From a conceptual point of view, such a limitation can be problematic, since it is well known that dyadic graph models suffer from shortcomings in modelling higher-order relationships with more than two participating terms or entities. In practice, this translates to an inability of the model to encode the cooccurrences of more than two words or entities in the same instance, unless extensive additional information about these instances is also included. As a result, existing document models either tend to fall short in modelling higher-order cooccurrences by breaking them down into pairwise relations, or cover only specialized applications.

To address this shortcoming, we propose cooccurrence *hypergraphs* as a versatile document model that extends beyond the capabilities of dyadic relationship models, and is instead based on the concept of hypergraphs [23, 24]. In contrast to dyadic graphs, in which an edge always connects exactly two nodes, hypergraphs allow the representation of higher-order relationships as hyperedges that may connect an arbitrary number of nodes. It is obvious that the inclusion of cooccurrences between sets of terms or entities is then a simple matter. Of course, this also raises the question of how such a model can be realized efficiently without collapsing under the combinatorial explosion of modelling all cooccurrence possibilities, which we address in the following. Furthermore, similar to the

implicit networks we discussed so far, we include not just terms as nodes, but also structural components of the document such as sentences and the document itself. Together with the possibility of linking nodes to entities of different classes in external knowledge bases, this leads to the notion of a *heterogeneous hypergraph* model that represents both the data and external information about the contained annotations.

While such a model for higher-order term cooccurrences is valuable in itself, our primary contribution and key to the practical application of the model is a core set of operators that support a variety of IR tasks. In the spirit of the base operators of relational algebra [47], we introduce a set of operators on hypergraphs that allow the selection and transformation of hyperedges to satisfy information needs and retrieval operations from the documents, even beyond the tasks that we have so far approached with the help of implicit networks. The operators then enable a seamless integration of additional data from external structured knowledge bases into the unstructured text as node attributes, and thus bridge the gap between structured and unstructured data.

Structure. The remainder of the chapter is structured as follows. In Chapter 7.2, we review related work for hypergraphs and their applications. We make the main contribution in Chapter 7.3, where we introduce our hypergraph-based document model and describe fundamental operations on such graphs. In Chapter 7.4, we discuss the implications for an applications of the model in practice, and show how central IR metrics and methods can be realized on top of the hypergraph representation in Chapter 7.5.

7.2 Related Work on Hypergraphs

Given the basic concept of our framework to model and explore term (co-)occurrences in document collections by using hypergraph structures, the related work can be split into three broad broad categories. The first two are network-based modelling of term cooccurrences, and knowledge extraction for knowledge bases, which we cover in Chapter 2.2 and 2.3. These networks, however, are focused either on term cooccurrences, or on knowledge-base like relationships. Thus, a general model that fully merges structured knowledge and unstructured text data independently of the application is still missing. The third category covers related work on hypergraphs, as we discuss in the following.

Hypergraphs

A major shortcoming of most existing models for the analysis and exploration of term cooccurrences is the restriction to dyadic term relationships by employing a dyadic graph

model that is limited to relations between exactly two nodes. Such models suffer from several deficiencies when the joint cooccurrences of multiple terms are modelled and analyzed, which can only partially be compensated by supplementing additional metadata as node attributes. To address these shortcomings, recent approaches increasingly utilize hypergraphs. In graph theory, hypergraphs have a long tradition and have been studied extensively [23, 24]. Due to their computational complexity and difficult realization on top of existing data management infrastructures, hypergraphs have not seen frequent use in practice. However, as is evident from recent publications, hypergraph implementations and computations scale well on novel computing infrastructures. For example, Heintz et al. discuss several challenges and opportunities when realizing hypergraph management systems [88] and present a flexible, distributed, and scalable processing system for hypergraphs called MESH [89]. Similarly, Huang et al. introduce the HyperX system, which supports efficient learning on and processing of distributed hypergraphs in Spark [100]. Unfortunately, these existing systems are generic and not well suited to the representation of documents or implicit network queries without major adaptation.

Applications of hypergraphs

Nowadays, despite the apparent shortage in wide-spread infrastructures, one can find emerging approaches that successfully and efficiently employ hypergraph models for diverse graph management and analysis tasks. On the one hand, some approaches extend traditional network analysis tasks to hypergraphs, for example to compute clustering coefficients [55], centrality measures [107], or spectral clustering [221]. On the other hand, there are also a number of hypergraph-based approaches that are more situated in traditional document-based information retrieval settings. Bellaachia and Al-Dhelaan model sentences as hyperedges and words as nodes in a hypergraph [20]. Based on this document model, they present an approach to the ranking of sentences by using random walks. In [19], they adapt this approach to short documents from microblogs. Similarly, Wang et al. model sentences as vertices and their multi-relationships by using hyperedges [208]. Based on this text hypergraph, they present a semi-supervised sentence ranking algorithm for query-based extractive summarization. Bendersky and Croft utilize hypergraphs to model queries (but not documents) to include term and phrase dependencies and improve retrieval operations [21]. Hypergraphs have also been proposed as basis for recommendation systems, for example in tagging data [230] and music recommendation [193].

Almost all of the above approaches come with their own specialized document model that is based on hypergraphs, ranging from a pure document-oriented view [20], to ap-

proaches that model complex relationships among select (typed) entities or features that are extracted from text documents. Furthermore, none of these approaches utilize hypergraph representations of generalized term cooccurrences to support other application frameworks. Given the plethora of information retrieval, analysis, and exploration approaches that are based on text documents and term cooccurrences in particular, we argue that it is pertinent to have a single yet flexible document model that can support the majority of the above applications. In the following, we discuss the formal basis of such a document model that uses hypergraphs, and include the necessary graph operators to support a wide range of applications on top of this model.

7.3 The Hypergraph Document Model

Before we introduce our hypergraph document model, we start by defining the underlying concepts and discussing the necessary document segmentation strategies. Afterwards, we formally describe the construction of the model, followed by the operators for the support of retrieval tasks on the hypergraph representation.

7.3.1 Preliminaries

Recall that a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_\mathcal{G})$ is a tuple consisting of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}_\mathcal{G}$ that connect the nodes. In most commonly used graphs, edges connect exactly two nodes and each edge therefore represents a pair of nodes. Thus, we find that $\mathcal{E}_\mathcal{G} \subseteq \mathcal{V} \times \mathcal{V}$. In the following, we refer to such graphs as *dyadic graphs*. Dyadic graphs may be weighted, meaning that a weight is associated with each edge, or directed, meaning that the order of nodes in an edge is of relevance.

In contrast to dyadic graphs, edges in a hypergraph consist of an arbitrary subset of nodes. Thus, for a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, the set of edges is a subset of the power set of nodes $\mathcal{E} \subseteq 2^\mathcal{V}$. We refer to such edges as *hyperedges*. Similar to edges in dyadic graphs, hyperedges may be weighted or directed (by partitioning the set of nodes that constitute an edge into a head set and a tail set). In the following, we focus on undirected hyperedges and simply refer to them as edges when the meaning is clear from the context. Recall that a graph or hypergraph is called *heterogeneous* if the set of nodes consists of nodes with distinct properties that can be used to partition them into subsets. Consider, for example, the entities in a knowledge graph that may represent persons, places, or organizations.

In the following, we discuss how documents can be segmented in such a way that we may represent them as a type of heterogeneous hypergraph. External knowledge such as

node type hierarchies are then easily included as dyadic graphs over the same set of nodes, or as leaves of a tree that encodes some term hierarchy.

7.3.2 Document segmentation

As unstructured text input, we consider a document collection given by a set of documents D. Each individual document $d \in D$ may have associated metadata, such as an author, a publisher, or a publication date. Each such document can then be segmented into smaller units. Specifically, we use terms as the smallest atomic unit and sentences as groups of terms. That is, for a given set of sentences S we consider a document $d \in D$ to be a subset of sentences $d \subseteq S$. In turn, sentences consist of terms $t \in T$ from the set of all terms in the collection. For the purpose of our model, we consider a term to be a word, compound word, or a multi-word expression with a specific meaning in the given sentence. For example, both *apple* and *apple tree* constitute terms, even though the latter consists of two words. A sentence $s \in S$ is then a subset of terms $s \subseteq T$.

Note that additional granularity levels in this segmentation hierarchy are possible. For example, *phrases* can serve as groups of terms that are parts of sentences, *paragraphs* allow the modelling of groups of sentences, whereas *volumes* could represent sets of documents within a collection. We do not include these additional segmentation hierarchies here, but it should be obvious how they can be formalized as a possible hierarchy of sentences, in analogy to the hierarchies for terms that we discuss in the following.

7.3.3 External term augmentations

The segmentation of documents into terms instead of mere tokens or words is both aided and required by the utilization of external knowledge sources such as knowledge bases. Thus, it is sensible to utilize a model that transcends mere document knowledge and that integrates structured and unstructured information instead. To this end, we design our model to support the augmentation of identified terms with additional external knowledge, including (named) entity information and links to knowledge bases that contain such data. Since terms constitute the majority of nodes of the hypergraph, knowledge about terms can be modelled as node attributes, while ontological or hierarchical information can be modelled as dyadic graphs or trees over the set of nodes. For example, the term *apple* can be tagged as part-of-speech *noun* or, depending on the context and position in the sentence, could be linked to an entity in a knowledge base representing the fruit or the technology company. Based on these links to external knowledge bases, terms can then be classified

into entity categories and hierarchies. For example, the company *Apple* should be classified as an organization if the mention can be linked to the corresponding knowledge base entity. Thus, we consider additional term information as attributes that are associated with the corresponding nodes. Naturally, the output of named entity recognition and named entity linking is a rich source of augmenting information.

Targets for entity linking could be gazetteers or knowledge bases such as YAGO [124], DBpedia [14], or Wikidata [206] as discussed in Chapter 2.3. On a linguistic level, it is also possible to link terms based on lexical background networks like WordNet [132] or one of the many analogous resources for different languages. Ideally, it should be possible to link any term to an underlying knowledge base or lexical resource. In reality, since information is often missing or incomplete, the resulting set of nodes is heterogeneous with regard to the available information (that is, attributes) for individual terms. For terms that cannot be linked, it is reasonable to assume some form of lemmatization or stemming to ensure that distinct terms and lexemes with identical meaning are also mapped to the same node in the graph, similar to the linking of entities. Alternatively, terms can be clustered and linked by more recent embedding approaches such as word2vec [131] or GloVe [146] to represent dyadic semantic relations that are present within the document collection or a reference corpus. While the possibilities of linking terms to external information are numerous, the approach can always be modeled as (heterogeneous) nodes that are linked to external dyadic graph structures for which numerous querying and reasoning approaches exist. In the following, we thus focus on a representation of the information that is contained in the document collection itself. To this end, we put a focus on term cooccurrences, which cannot be handled adequately by dyadic approaches.

7.3.4 Hyperedge composition

To introduce the construction principle of hyperedges in the following, we briefly define the set of nodes and a system for describing relative term positions within the sentence structure of the documents. Ultimately, we obtain a hypergraph representation $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ of the documents. External knowledge base structures can then be considered as dyadic graphs $\mathcal{G}_{KB} = (\mathcal{V}, \mathcal{E}_{KB})$ on the same set of nodes or a subset thereof.

Relative term and sentence position

To represent the occurrence of terms in sentences as well as the cooccurrences of terms, we introduce the concept of relative term and sentence position as a function that maps sentences and terms to integer values.

Definition 7.1 (Relative position *pos*). Let ς denote a monotonic, consecutive numbering of sentences such that each sentence s is mapped to its index $\varsigma(s)$ in the document. For two sentences s_1 and s_2, we then define the relative position $pos : (T \cup S)^2 \to \mathbb{Z}$ as

$$pos(s_1, s_2) := \varsigma(s_2) - \varsigma(s_1). \tag{7.1}$$

Analogously, we define positions for terms. For a given term t, let $s(t) \in S$ denote the sentence that contains the term. For two terms t_1 and t_2, we then define their position as

$$pos(t_1, t_2) := \varsigma(s(t_1)) - \varsigma(s(t_2)). \tag{7.2}$$

Note that position values may be negative, and that they are symmetric for inverse arguments. The absolute value $|pos|$ of a position score then is a proper distance in the number of sentences between the input terms or sentences. For ease of notation, we also include a relative position between documents and sentences. To this end, we simply let the position of a document d with regard to a sentence s be $pos(d, s) := 0$ if and only if $s \in d$, and $pos(d, s) := \infty$ otherwise.

A position system on the sentence level as defined above is the approach that we consider to be the most useful, since sentences represent coherent units of linguistic structure. However, alternative position functions are equally viable and could, for example, be based on a paragraph-distance instead of sentences, or even on a term-distance to support the traditional occurrence windows that are often used in other document models.

Graph nodes $\mathcal{V}$

To construct the set of graph nodes for the hypergraph, we first require a set that contains the structural components, which we call the core set.

Definition 7.2 (Core set Core). For a set of documents D, sentences S, and terms T, we define the core set as $\textsc{Core} := T \cup S \cup D$.

Based on the notion of unique identifiers for terms, sentences, and documents, this allows the identification of individual nodes. To satisfy the requirement of modelling cooccurrences across sentence boundaries, we also include the position of the node core in the final node representation.

Definition 7.3 (Hypergraph nodes $\mathcal{V}$). For a core set Core, let the set of nodes for a hypergraph be $\mathcal{V} \subseteq \textsc{Core} \times \mathbb{Z}$.

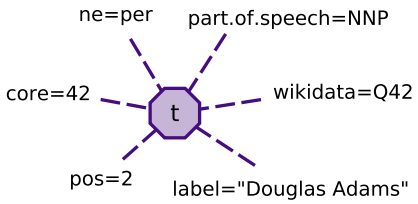

Figure 7.1: Schematic example of a term t with attributes core $t.core$ and position $t.pos$, as well as added part-of-speech and named entity annotations, and external Wikidata knowledge base attributes.

Thus, each node of the hypergraph consists of two components, where the first component denotes the identifier that represents the content, while the second component denotes the relative position that is used to determine cooccurrences and cooccurrence distances. Note that, based on this definition, the set of nodes in potentially infinite even for a finite number of documents. However, since the size of documents is finite in practice and the maximum position values are therefore bounded by the number of sentences in a document, the set of nodes is also finite in practice.

Node attributes

Each node as defined above is a tuple with the two primary components *core* (the term identifier) and *position* (the relative position of the term in the document). In the following, we refer to the core of a node $v \in \mathcal{V}$ as $v.core$, and use $v.pos$ to denote the position. Note that the content of each node is uniquely identified by the core component (in practice, this can be any unique identifier). For term- and entity-centric analyses, terms can be associated with additional attributes by introducing a function that maps nodes to appropriate attribute spaces (for example, named entity types, alternative labels, knowledge base identifiers, or parts-of-speech). We represent these optional attributes with the same component notation. An important attribute in this context is the type of a node, denoted as $v.type$, which classifies it into a document, sentence, or term. Other useful attributes include the named entity type $v.ne$ of a term, or the publication date $v.date$ of the document in which this term occurs. For an example of a node with attributes, see Figure 7.1.

Node equivalences

To compare graph nodes, we introduce the notion of node equivalences. Naturally, we base these on the two primary attributes, namely the core *core* and the position *pos*.

Definition 7.4 (Node equivalence). We say that two nodes $v, w \in \mathcal{V}$ are equal and write $v = w$ if and only if their two primary components are identical. That is,

$$v = w :\Leftrightarrow v.core = w.core \wedge v.pos = w.pos \tag{7.3}$$

However, since terms, sentences, and documents are uniquely identified by the core, we also use the notion of approximate equivalence, for which the positions do not match.

Definition 7.5 (Approximate node equivalence). We say that two nodes $v, w \in \mathcal{V}$ are approximately equal and write $v \approx w$ if and only if their two primary components are identical. That is,

$$v \approx w :\Leftrightarrow v.core = w.core \tag{7.4}$$

Further relations are of course viable, such as the less-than and more-than relations $\leq_n$ and $\geq_n$, in which the core is identical and nodes are ordered by their position component. These can be used to induce a partial order on the set of hyperedges, but are not required for the applications that we present in the context of this chapter.

Hyperedges $\mathcal{E}$

Following the segmentation of documents and the definition of term and sentence positions, we construct hyperedges to represent the entire document collection around the cooccurrences of terms. Based on the set of nodes $\mathcal{V}$ of the graph, we obtain the set of all *possible* hyperedges over these nodes, which we require for the proofs of correctness of the hyperedge operators.

Definition 7.6 (Set of possible hyperedges Σ). For a given set of core nodes Core, we define the set of possible hyperedges as $\Sigma := 2^{\text{Core}\times\mathbb{Z}}$.

Thus, Σ describes all possible sets that can be constructed from all possible nodes. Obviously, Σ is too large to be useful in practice, and only serves to formalize the approach. However, from these possible edges, we can identify a subset $\mathcal{E} \subseteq \Sigma$ that represents the input document collection and allows us to define a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, in which each edge $e \in \mathcal{E}$ is constructed around a sentence s_e in the document collection. We call s_e the *primary sentence* of e and set the relative position of the sentence to itself to zero, that is, $p(s_e, s_e) := 0$. To model the content and context of sentences, each edge is then composed of nodes as defined above.

Definition 7.7 (Hyperedge e). Let s_e denote a primary sentence around which an edge is to be constructed. Let $c \in \mathbb{N}$ denote the size of a suitable context window, counted in the

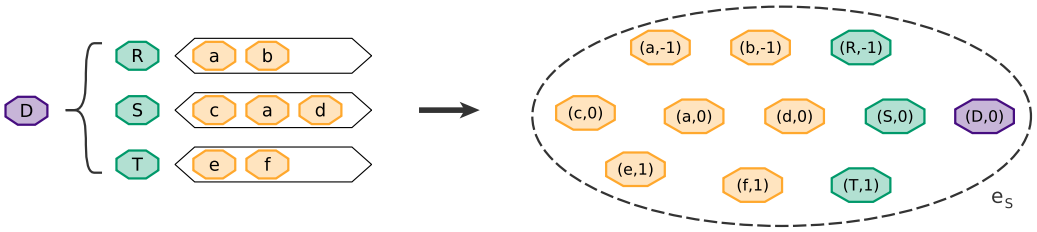

Figure 7.2: Schematic visualization of the hyperedge creation process. Shown is a document D with sentences R, S, T and terms $a - f$, along with the resulting hyperedge e_s that represents the primary sentence S at a context window size of $c = 1$.

number of sentences to each side of a primary sentence. Then $e := \mathcal{V}_T^e \cup \mathcal{V}_S^e \cup \{(d_e, 0)\}$ is a hyperedge that contains as nodes the set of terms in s_e and in nearby sentences, along with their relative positions in document d_e. The node contributions from surrounding sentences within the window c and the respective term positions are defined as

$$\mathcal{V}_T^e := \{(t, pos(t, s_e)) : t \in T \wedge |pos(t, s_e)| \leq c\} \tag{7.5}$$

$$\mathcal{V}_S^e := \{(s, pos(s, s_e)) : s \in S \wedge |pos(s, s_e)| \leq c\} \tag{7.6}$$

Formally, each edge constitutes a set $e \subset \mathcal{V}$ that contains the terms in and around the primary sentence s_e up to a distance of c.

The parameter c is thus directly related to the size of the context window around a primary sentence that induces the term cooccurrences, and corresponds to the context window that we use for the implicit network models in previous chapters. In Chapter 7.4, we discuss the practical implications of this parameter for the storage size of the data.

Definition 7.8 (Set of hyperedges $\mathcal{E}$). The set of hyperedges of a hypergraph representation of a document collection D is then the set $\mathcal{E} \subseteq \Sigma$, such that each $e \in \mathcal{E}$ is a valid hyperedge for some sentence in a document in D.

In Figure 7.2, we show an overview of the edge generation process. In the following, since hyperedges essentially constitute sets of nodes, we write $v \in e$ to denote that edge e is incident on node v.

Edge relations

To construct operators that work on the hypergraph model, we use a number of relations between edges that are based on their contained nodes. Most fundamentally, we define the notion of edge equality based on set semantics.

Definition 7.9 (Edge equivalence). We say that two edges are equal if and only if node equality as defined in Equation 7.3 is a bijection between the two edges. Similarly, we say that edges e and f are approximately equivalent and write $e \approx f$ if and only if a bijection between the two sets can be defined on the basis of approximate node equivalence.

Finally, we consider edge containment, which we require to represent overlaps between nodes to formally define join operators for hyperedges.

Definition 7.10 (Edge containment). We say that edge e is contained in edge f and write $e \sqsubseteq f$, if all nodes in e have an approximately equivalent node in f whose absolute position is at most as large as in e. Formally, we have

$$e \sqsubseteq f :\Leftrightarrow \forall v \in e : (v \in f) \wedge (|pos(v, f)| \leq |pos(v, e)|). \tag{7.7}$$

In the following, we also refer to e as a *subedge* of f if $e \sqsubseteq f$. Intuitively, this notion of edge containment requires that all terms are at least as closely positioned in the containing edge as they are in the subedge.

7.3.5 Propositional expressions

Based on the hypergraph document model, we are almost ready to consider the base operators that can be used to select and transform the hyperedge representations of document collections. However, to formalize these operators, we first introduce the notation for propositional expressions.

In analogy to relational algebra, we rely on propositional expressions for the selection of nodes from an edge or of edges from a set of edges. Formally, a propositional expression (referred to as *expression* in the following) can be any syntactically adequate unary formula that takes a node or edge as argument and maps it to a truth value. With node attributes being the most discriminative feature of hyperedges, most relevant expressions rely directly on attribute values and are of the form $v.att\, \phi\, x$, where att is some node attribute, x is a value from the domain of this attribute (or a subset thereof), and $\phi \in \{=, \neq, \leq, \geq, <, >, \in\}$. In the following, we consider expressions θ that contain node attributes to be true for an edge if the edge contains at least one node for which the expression is true. That is, for an edge $e \in \mathcal{E}$, we let

$$\theta(e) = true :\Leftrightarrow \exists v \in e : \theta(v) = true \tag{7.8}$$

For the sake of maintaining a readable notation, we use an abbreviated notation and instead of $\exists v \in e : \theta(v)$, we simply write $\theta(e)$ when using the expression.

Distance in expressions

As a special shorthand, we use the concept of distance δ instead of position *pos* where the sign of the position value does not matter. Specifically, we define the distance as

$$\delta := |v.p| \tag{7.9}$$

and use it in expressions of the form $\delta\, \phi\, k$, where $k \in \mathbb{Z}$ and ϕ is some valid relation over the real numbers. The expression is true if ϕ holds for δ and k. Note that δ may not always be a proper distance metric, since the identity of indiscernibles is violated when sentence or paragraph distances are used. For term nodes, δ corresponds to the sentence-based distances that we use for generating the edge weights of implicit entity networks in the previous chapters.

Existence of attributes

Since nodes may or may not possess a certain attribute due to the heterogeneity of the graph, an important distinction criterion is the existence of an attribute, regardless of the value. For example, it may be important to distinguish between named entities and all other terms. Here, we simply denote with $\exists v.att$ an expression that is true if node v has attribute att, and false if it does not.

7.3.6 Closed operators on hyperedges

To support the information retrieval tasks that we consider in Chapter 7.5, we require two types of operators. We first introduce a number of operators that are closed on sets of hyperedges, meaning that both the input and the output are sets of hyperedges. Since isolated nodes can be treated as edges with one element, all sets of hyperedges can without loss of generality also be considered to represent hypergraphs. Afterwards, we introduce operators that are not closed, and instead map sets of hyperedges to the set of real numbers $\mathbb{R}$, or to integers $\mathbb{Z}$, which are useful for generating aggregation statistics.

Set operators

Three basic binary operators, which are trivially usable, are the the asymmetric set minus $-$, the union $\cup$, and the intersection $\cap$. Naturally, they conform to their usual set semantics. However, note the difference between set operations on two hyperedges that merge sets of nodes, and on sets of hyperedges that essentially merge entire hypergraphs.

Selection operator σ

The selection is defined in analogy to the definition of a subhypergraph, and equates to the selection of all hyperedges from an input set that satisfy some selection expression. For example, a subset of hyperedges could be selected based on certain nodes that these edges contain, or on attributes of those contained nodes.

Definition 7.11 (Selection σ). Let θ be an expression, and let $\hat{\mathcal{E}} \subseteq \mathcal{E}$ be a set of input edges. Then $\sigma_\theta : 2^\Sigma \to 2^\Sigma$ is the selection function that selects a subset of these edges according to θ as output edges. Thus,

$$\sigma_\theta(\hat{\mathcal{E}}) := \{e \in \hat{\mathcal{E}} : \theta(e)\}. \tag{7.10}$$

If we were to consider this operator in analogy to relational algebra, we could relate hyperedges to tuples. The selection of edges from a set of edges is then semantically similar to the selection of tuples from a table based on the values of some combination of attributes of the tuples.

Projection operator π

The projection of hyperedges can be defined in analogy to partial hypergraphs, which is to say that it handles the removal of nodes from hyperedges based on the provided expression. That is, all nodes that do not satisfy a given condition are removed from the input hyperedges. For example, all nodes of the sentence type or all nodes with a given attribute in an external knowledge base could be removed from the input edges. For the sake of notation, we first define the projection for a single hyperedge and then generalize.

Definition 7.12 (Projection π). For some expression θ, let the projection $\pi_\theta : \Sigma \to \Sigma$ be defined for an input edge $e \in \mathcal{E}$ as

$$\pi_\theta(e) := \{v \in e : \theta(v)\}. \tag{7.11}$$

On this basis, for a set of edges $\hat{\mathcal{E}} \subseteq \mathcal{E}$, we can then define the more general projection function $\pi_\theta : 2^\Sigma \to 2^\Sigma$ as

$$\pi_\theta(\hat{\mathcal{E}}) := \{\pi_\theta(e) : e \in \hat{\mathcal{E}}\}. \tag{7.12}$$

Similar to the selection, we can consider the projection operator in analogy to relational algebra. If we were to identify the nodes of hyperedges with the attributes of a tuple, then the projection works in the same way as it does in relational algebra by reducing the tuples

to the given attributes. However, note that edges do not necessarily need to contain nodes with every attribute that occurs in θ, due to the heterogeneity of the set of nodes.

For simplicity, we use three shorthand notations for common projection operations that relate to the structure of the documents in the modelled collection and are are needed frequently. Specifically, we use π_{term}, π_{sen}, and π_{doc} to project hyperedges to the contained terms, sentences, or documents. That is, we remove all nodes from all input edges that are not of the specified type.

Reduction operator r

The likely most important aspect of a hypergraph document model is its capability to represent higher-order cooccurrences. However, many existing models use dyadic graph representations, so the inclusion of an operator that transforms hypergraphs into dyadic graphs by creating dyadic edges between all nodes in a hyperedge is required. Essentially, such an operator replaces hyperedges with cliques of dyadic edges that fully connect the nodes of the hyperedge. However, note that the resulting list of edges can still be represented as a 2-uniform hypergraph, meaning that all edges have a cardinality of two, and the set of hyperedges is therefore closed under this operator. We refer to this operation as a *reduction* since it decreases the size of hyperedges.

Definition 7.13 (Reduction r). The reduction of a single hyperedge e is defined as the function $r : \Sigma \to 2^{\Sigma}$, such that

$$r(e) := \{\{v, w\} : v \neq w \wedge v, w \in e\}. \tag{7.13}$$

Based on this single-edge reduction, let the general reduction function $r : 2^{\Sigma} \to 2^{\Sigma}$ for sets of hyperedges $\hat{\mathcal{E}}$ be defined as

$$r(\hat{\mathcal{E}}) := \bigcup_{e \in \hat{\mathcal{E}}} r(e). \tag{7.14}$$

For simplicity, we use $r_{\neq}$ as shorthand notation for a reduction in which edges in the resulting dyadic graph are discarded if they connect nodes of the same type (term, sentence, or document). Similarly, we use $r_{=}$ to denote a resulting dyadic graph in which only edges between nodes of the same type are retained. Both can be formally defined based on the operators σ, π, $-$, and $\cup$. Of course, similar operators can be conceived for other term attributes, such as the type of entity for nodes that have linked knowledge base attributes.

More generally, we may also consider an operator that does not extract dyadic edges but k-uniform hyperedges as subsets of fixed size k. If we denote with $[A]^k$ the set of all subsets of a set A of size k, then r^k is defined as

$$r^k(e) := [e]^k \qquad \text{and} \qquad r^k(\hat{\mathcal{E}}) := \bigcup_{e \in \hat{\mathcal{E}}} r^k(e). \tag{7.15}$$

Therefore, the dyadic reduction is then a special case of r^k for $k = 2$, and we write r instead of r^2 where it is clear from context. Similar to the case above, we use $r^k_{\neq}$ and $r^k_{=}$ as shorthand for resulting graphs in which all nodes of an edge have a different or identical type, respectively. In Chapter 7.5, we provide examples of applications in which these hypergraph to hypergraph operators are of practical use.

As a final observation, we note that the created lower-order edges in the reduced graphs are not necessarily distinct, especially for larger sets of hyperedges. Consider, for example, the reduction of two hyperedges that contain identical term nodes but differing sentence nodes. In a resulting dyadic reduction, all edges between terms would then be duplicated. Due to set semantics, those duplicate edges are lost after the reduction unless multi-sets are used, which would result in a much more complex model that does not appear useful from a practical perspective. As an alternative, an aggregation weighting function ω can be used to assign a weight to the resulting edges (which is, to the best of our knowledge, how this is predominantly handled in practical applications). As a simple example, the total number of all such edges could be assigned as a weight, which surmounts to counting the duplicates. For the dyadic reduction r^2, this then results in a graph in which edges between terms are assigned their cooccurrence count. More complex functions that also include the relative term position are of course possible. To formalize this notion, assume a function r_m that behaves equivalently to r, but uses multiset semantics. Furthermore, for a multiset $\mathcal{S}$ of edges, let $\{\!\{\mathcal{S}\}\!\}_e := \{e' \in \mathcal{S} : e \approx e'\}$ denote the subset of edges that are approximately equivalent to a given edge e. Formally, we can then regard the reduction with an aggregation weight function as a family of functions $r_\omega : \Sigma \to \mathbb{R}$ such that

$$r_\omega(\hat{\mathcal{E}}) := \{(e, \omega(\{\!\{r_m(\hat{\mathcal{E}})\}\!\}_e)) : e \in r_m(\hat{\mathcal{E}})\}. \tag{7.16}$$

Thus, any conceivable function ω that takes a set of edges between a fixed set of nodes and computes a weight for the aggregated edge can be used in this context. Obviously, such functions are especially useful in retrieving implicit networks from the hypergraph representation. We use this reduction operation in Chapter 7.5 to demonstrate how localized queries to an implicit network representation can be processed by the hypergraph model.

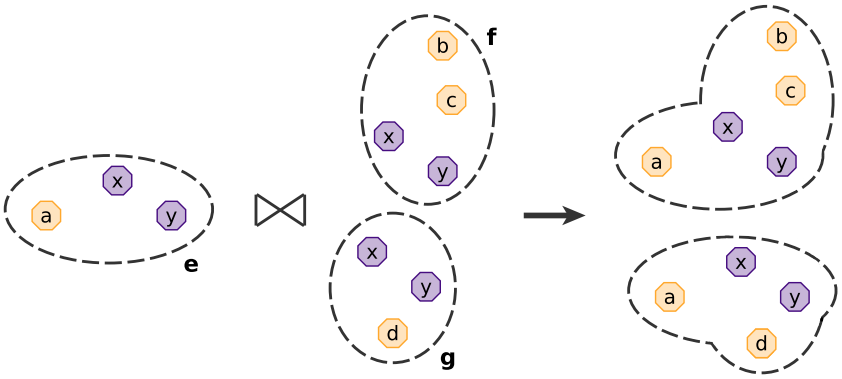

Figure 7.3: Schematic example of a join between two sets of edges. For the join $\{e\} \bowtie \{f, g\}$ on the common subedge $\{x, y\}$, two edges are created in the output set, one by extending f with e, and one by extending g with e. The common nodes x and y of the edges are highlighted in purple.

Join operators $\bowtie$

As a final operator, we introduce the join of hyperedges, which is inspired by the concept of joins in relational algebra, but is semantically distinct. Unlike edges in a dyadic graph, hyperedges can be extended to include additional nodes. Thus, we consider the extension of edges with nodes from other edges that overlap on some subset of nodes. In terms of a dyadic graph, this translates to the construction of growing paths from a starting node or set of starting nodes, or to growing clusters. From a retrieval perspective, this enables the (context-sensitive) expansion of relevant cooccurrences around some set of terms. Conceptually, we distinguish between two types of join operations: joins around a common core of nodes, and joins around common subedges. The first case we consider is the join on a common subedge ε that is shared by all edges, which we call the ε-join.

Definition 7.14 (ε-join $\bowtie_\varepsilon$). For two sets of hyperedges $\hat{\mathcal{E}}, \hat{\mathcal{F}} \subseteq \mathcal{E}$ and a hyperedge ε, let the ε-join be the operation that extends each edge in $\hat{\mathcal{E}}$ that contains the given subedge, with each edge in $\hat{\mathcal{F}}$ that contains the given subedge. Thus,

$$\hat{\mathcal{E}} \bowtie_\varepsilon \hat{\mathcal{F}} := \{e \cup f : e \in \hat{\mathcal{E}} \wedge f \in \hat{\mathcal{F}} \wedge \varepsilon \sqsubseteq e \wedge \varepsilon \sqsubseteq f\}. \tag{7.17}$$

In an application scenario, this join allows, for example, the identification and exploration of common or distinct cooccurrences of terms in the graph with a set of query terms that are provided as input.

To expand on this specific join on fixed subedges, we can also define the more general j-join, which joins all edges that contain any common subedge of a given size j or larger.

Definition 7.15 (j-join $\bowtie_j$). Let $\hat{\mathcal{E}}$ and $\hat{\mathcal{F}}$ be two sets of hyperedges $\hat{\mathcal{E}}, \hat{\mathcal{F}} \subseteq \mathcal{E}$. Furthermore, let $j \geq 1$. Then the j-join is defined as

$$\hat{\mathcal{E}} \bowtie_j \hat{\mathcal{F}} := \{e \cup f : e \in \hat{\mathcal{E}} \wedge f \in \hat{\mathcal{F}} \wedge (\exists \varepsilon : |\varepsilon| \geq j \wedge \varepsilon \sqsubseteq e \wedge \varepsilon \sqsubseteq f)\}. \tag{7.18}$$

In contrast to the ε-join, the j-join does not expand all edges in a common direction, but rather expands each edge in a (possibly distinct) suitable direction.

Both joins rely on the relative position values of nodes and thus allow a restriction of the cooccurrences in the underlying documents to a desired proximity level. For a schematic view of the join operations, see Figure 7.3. Note that for the hyperedges that result from a join operation, position values are not necessarily well defined. The edge join operation is thus powerful, but not every possible join result is semantically meaningful, similar to a join in relational algebra. In Chapter 7.5, we discuss how edge joins can be used for path and context expansion operations in an interactive investigation of a document collection that relies on querying the hypergraph model.

In addition to the two join operators defined above, it is of course possible to conceive further join criteria, such as the $\approx$ edge relation for subedges, or a more general θ-join on arbitrary expressions. However, since these types of joins are not required for the applications that we consider in the following, and they are straightforward to define based on the above intuitions, we omit them here.

7.3.7 Closure under operators

Most retrieval operations on the hypergraph structure require the combination (or chaining) of multiple operators on the set of input edges to obtain the desired result. Therefore, it is important that the output that is produced by the operators is guaranteed to be a set of hyperedges that can be used as input for the next operation. Formally, this requires us to show that the set of all possible sets of hyperedges 2^{Σ} is closed under the operators that we introduced above. In the following, we briefly discuss and prove this property for sets of hyperedges and all operators.

Lemma 7.1 (Closure under set operations). The set of possible sets of hyperedges 2^{Σ} is closed under the set operators set minus $-$, union $\cup$, and intersection $\cap$.

Proof. Since power sets are closed under the basic set operators and 2^{Σ} is a power set, it follows that 2^{Σ} must be closed under the minus, union, and intersection operators. □

Lemma 7.2 (Closure under selection). The set of possible sets of hyperedges 2^{Σ} is closed under the selection operator σ.

Proof. Given a subset of edges $\hat{\mathcal{E}} \subseteq \Sigma$ and some expression θ, it must hold that $\sigma_\theta(\hat{\mathcal{E}}) \subseteq \hat{\mathcal{E}}$ and $\sigma_\theta(\hat{\mathcal{E}}) \subseteq \Sigma$ by transitivity of the set containment. Therefore, the set of possible sets of hyperedges is closed under selection. □

Lemma 7.3 (Closure under projection). The set of possible sets of hyperedges 2^{Σ} is closed under the projection operator π.

Proof. Given a subset of edges $\hat{\mathcal{E}} \subseteq \Sigma$ and some expression θ, let $\pi_\theta(\hat{\mathcal{E}})$ denote the set of edges that is obtained by projecting $\hat{\mathcal{E}}$ according to expression θ. Then for all edges $e \in \pi_\theta(\hat{\mathcal{E}})$, there exists an edge $e' \in \hat{\mathcal{E}}$ such that $e \subseteq e'$. Since $e' \in \Sigma$ and Σ is also a power set that is closed under the set minus operation, it must hold that $e \in \Sigma$. Thus, $\pi_\theta(\hat{\mathcal{E}}) \subseteq \Sigma$, meaning that 2^{Σ} is closed under the projection operator. □

Lemma 7.4 (Closure under reduction). The set of possible sets of hyperedges 2^{Σ} is closed under the reduction operator r.

We observe that the reduction operator effectively creates the set of all k-uniform subedges for each hyperedge in the input set (that is, it creates all possible subedges of size exactly k). Therefore, we omit the proof of closure since it follows directly from the proof provided above for the projection operator.

Lemma 7.5 (Closure under join). The set of possible sets of hyperedges 2^{Σ} is closed under the join operators $\bowtie_\varepsilon$ and $\bowtie_j$.

Proof. To show the closure of the join operation, we can consider both the ε-join and the j-join simultaneously. Without loss of generality, let $f = \{e_1\} \bowtie \{e_2\}$ denote the single hyperedge that results from the join of edges $e_1 \in \Sigma$ and $e_2 \in \Sigma$ on some common subedge. Then, $f = e_1 \cup e_2$, from which it directly follows that $f \in \Sigma$ since Σ is closed under union. Given that the above observation holds for any two edges, it must also hold for the join $\hat{\mathcal{F}} = \hat{\mathcal{E}}_1 \bowtie \hat{\mathcal{E}}_2$ of any two sets of edges $\hat{\mathcal{E}}_1 \subseteq \Sigma$ and $\hat{\mathcal{E}}_2 \subseteq \Sigma$ that $\hat{\mathcal{E}} \subseteq \Sigma$. Thus, 2^{Σ} is closed under the join operator. □

Based on these arguments, we find that it is possible to combine and chain hypergraph operators as needed. This is then the basis for the emulation of implicit networks and further information retrieval tasks on top of the hypergraph model in Chapter 7.5.

7.3.8 Non-closed operators

In addition to the five types of basic operators defined in the previous section, there are applications that require additional operators that map sets of hyperedges to scalar values instead of other sets of hyperedges. A prominent example that we use in the following is the counting operator that returns the number of edges in a graph. Formally, we define it as a function $count : 2^{\Sigma} \rightarrow \mathbb{N}$, such that for an edge set $\hat{\mathcal{E}} \in 2^{\Sigma}$, we obtain $count(\hat{\mathcal{E}}) := |\hat{\mathcal{E}}|$. While the operator is trivial, it also forms the basis for the large collection of existing statistical methods that rely on counts of term cooccurrences.

Further similar functions are of course viable, but since they are not required for the example applications that we consider in the following, we do not discuss them here.

7.4 Practical Implications

In our description of the model, we have so far focussed on the conceptual aspect of representing multiple documents as a hypergraph, and omitted the details of a physical representation of the data. In the following, we thus briefly highlight and discuss key implications for the application of the proposed model in practice. Among these potential practical implications, we consider three to be of primary concern, namely the question to which degree hyperedges need to be materialized, the space requirements of the model, and the potential for using distributed storage solutions.

Physical storage and materialization

The efficient storage of dyadic graphs is by no means a simple task, and arbitrary hypergraphs are justifiably known for inducing an even more daunting complexity than dyadic graphs. This raises the question of how a hypergraph representation of large document collections can be realized efficiently. In this particular instance, however, hypergraphs only serve as a formalization of the structure that is inherent to natural language constructs, which limits the actually observed complexity on real data. In practice, we can rely on three key observations about the structure of documents. First, we find that documents are either immutable in many applications (for example, consider streams of news articles in which documents are only added), or edited in a local and limited scope (for example, consider gradually changing Wikipedia articles). Second, while natural language is recursive and sentences or paragraphs could theoretically grow to arbitrary length, they

tend to have limited extent in practice, thereby putting an upper bound on the size of hyperedges. Third, we can assume a roughly Zipfian distribution of words [150], with many of the interesting query terms being located in the long tail of the distribution. This observation holds even more true for applications that are focussed on entities, since entity mentions are rare in comparison to the remaining terms in most contexts. As a result, we can store the data of documents and sentences sequentially and only materialize the hyperedges around selected terms at query time. Due to the sequential storage of sentences that are contained in documents and the terms that are contained in sentences, the generation of hyperedges does not suffer from overheads that have the potential to grow arbitrarily large, but is instead limited to a localized scope.

Storage space requirements

For storing large document collections, the storage size of the textual data is an important consideration. At first glance, the replication of the data that is stored in the edges of the hypergraph model appears to be excessive, due to cooccurrence window sizes that exceed the sentence boundary. However, this observation is purely limited to the conceptual view. In practice, if the sentences of the documents are stored sequentially as discussed above, they can be read efficiently to materialize hyperedges at query time with minimal access to secondary or tertiary storage. As a result, the physical replication of the data is not necessary, meaning the actual storage requirements are in $O(|S|)$ and thus linear in the number of stored documents. Potentially stored additional attributes of terms do not increase this beyond a constant factor. Furthermore, we find that the representation of terms as unique nodes is bound to save space in comparison to raw text, since an integer identifier is generally much more compact than a character string. Depending on the utilized position scheme, it may not even be necessary to explicitly store the position values of terms, and instead realize sentences and hyperedges as ordered tuples of terms. Sentence- or term-based positions are than easily obtained at query time. Overall, the storage requirements are therefore in the same order of magnitude as they would be for the raw text, since term labels and attributes can be stored in a central dictionary or lookup table. Based on this structure, the efficiency of selection operations can be ensured by using a variety of indices for hyperedges that are based on relevant node attributes. Thus, the model may be implemented on top of existing storage systems for hypergraphs or even in relational database management systems, where it fully profits from existing indexing techniques for efficient retrieval. For the retrieval of term occurrences in the data during as selection operations,

such index structures then require an amount of space that is loglinear in the number of terms, and therefore loglinear in the size of the entire document collection.

Data distribution

Given the amount of text data that is available on a Web scale, distributed storage solutions are a necessity in many applications. For the proposed hypergraph model, we find that a partition of the underlying document collection can be implemented at the document level. As we have seen during the exploration of topics in Chapter 6.5, document clusters (such as those formed by news outlets) can be used to partition the data in meaningful ways. Especially for implicit network applications, the results of individual distributed queries are then easy to merge. Since hyperedges are well-defined within documents, and the merging of hypergraphs from multiple documents is both commutative and associative, the order in which they are merged does not matter. Therefore, selection and projection operations can be distributed to shards that store different partitions of the document collection, with their results being combined afterwards. While there is certainly potential for optimized partitioning strategies, and any such approach would require additional research and experimentation, the underlying concept of the model is inherently compatible with a distribution of the data.

7.5 Hypergraph Model Support for IR Applications

Based on the hypergraph model and the operators defined in Chapter 7.3, we now investigate what applications can be supported by such a representation. The literature has a multitude of methods that rely on the extraction of the cooccurrences of terms, words, or entities for tasks so diverse as exploratory search, event detection, or summarization. To highlight the versatility of the proposed hypergraph model for document collections, we show on a number of example applications, how the model can be used to reproduce and support existing information extraction and retrieval techniques. An exhaustive coverage of techniques is beyond the scope of this thesis, so we focus on a representative selection that employs diverse operators or reproduces well-known baseline methods. We show how the model can emulate or support a wide range of existing approaches, solely on the basis of the hypergraph operators. Since these operators are inspired by those of relations algebra, they tend to resemble queries in relational algebra.

7.5.1 Exploratory search

We first consider a number of elementary cooccurrence query operations that represent collocation analyses, and serve as examples of an initial investigations into the content of a document collection. Typically, such an investigation or manual inspection starts with a query term t, whose context is explored. This basic search operation is realized by the selection of terms that cooccur with the given term in a context window of size at most c, which we can represent as

$$\mathcal{E}_{cooc}(t, c) = \pi_{v.core \neq t}(\pi_{\delta \leq c}(\sigma_{v.core=t}(\pi_{term}(\mathcal{E})))) \tag{7.19}$$

Thus, such a query can be answered by projecting the content of hyperedges to terms, selecting edges that contain the term in a given window size, and dropping the search term from these edges. Of course, we can also retrieve sentences or documents as source information of the cooccurrence instances by adjusting the projections accordingly. Going beyond single-term queries, if we are interested in a sets of terms T_q as query input, we can combine the above operations for individual terms by intersection to obtain

$$\mathcal{E}_{cooc}(T_q, c) = \bigcap_{t \in T_q} \mathcal{E}_{cooc}(t, c). \tag{7.20}$$

Similar to the case above, retrieving source information may be used to address additional document or sentence retrieval tasks. For further explorations, or to obtain count statistics (for example, to create rankings of adjacent term pairs), both of the above results can also be reduced to a dyadic graph. As we can see from these simple applications, it becomes fairly straightforward to think of cooccurrence-related retrieval tasks in terms of selection and projection operations that are based on desired types or node attributes.

To explore more complex cooccurrence patterns, joins are helpful tools. Consider, for example, a query in which we want to extract location mentions (so called toponyms) at which a specified person is mentioned together with groups of other persons (that is, this person's meeting places). For a given term t_p of named entity type person, and a minimum group size j, we can formulate the operation as

$$\begin{aligned} \mathcal{E}_1 &= \sigma_{v.core=t_p}(\pi_{v.ne=per}(\mathcal{E})) \\ \mathcal{E}_2 &= \pi_{v.ne \in \{per, loc\}}(\mathcal{E}) \\ \mathcal{E}_{places}(t_p, j) &= \pi_{v.ne=loc}(\mathcal{E}_1 \bowtie_j \mathcal{E}_2) \end{aligned} \tag{7.21}$$

that returns all such place mentions. If we are instead interested in places where a given person was mentioned with a specific group of other persons, an ε-join could be used instead. While these are only some examples of potential queries, they highlight the possible applications of the operators in an exploration scenario.

7.5.2 Vector space model

The vector space model (see Chapter 2.1) is a classic representative of document models, and is based on the bag-of-words representation for sentences, which is easily emulated through hyperedges. Established retrieval and ranking methods for the vector space model include numerous variations of *tf-idf* or the BM25 metric [156] that are based on the term count statistics term frequency *tf* (the frequency of a term in a document) and document frequency *df* (the number of documents in which the term occurs) of documents or sections of documents. Using the hypergraph representation and operators, the frequency of a term t in a document d can be obtained as

$$tf(t,d) = count(\sigma_{v.core=t}(\pi_{\delta=0}(\sigma_{v.core=d}(\mathcal{E})))) \tag{7.22}$$

by counting the sentences in the document that contain the term. Similarly, the document frequency is given by

$$tf(t) = count(\pi_{doc}(\sigma_{v.core=t}(\mathcal{E}))). \tag{7.23}$$

Thus, obtaining the essential components for these commonly used scoring functions is a simple task. Other variations of these measures can be formulated analogously since they depend on the same or similar count statistics, which shows that the proposed model includes these measures as baselines. Obviously, the computation of such scores can be combined with subsequent explorations as described above within the same framework. Of course, depending on the underlying architecture for storing the hypergraph, it is likely beneficial to include aggregation operators to count occurrences more efficiently in batches if the values are required for all terms.

7.5.3 Graph-based summarization

Automatic text summarization is a large area of research in which a multitude of methods employ dyadic graph representations of the documents. Here, we consider LexRank [52] as a well known example for such an approach. Common to the majority of summarization methods is the representation of sentences as nodes of a graph. Nodes are then connected

by edges that encode some form of sentence similarity. Subsequent steps extract representative sentences from this graph, for example through centrality computations or random walks. Based on a hypergraph representation of the input documents, a sentence graph $\mathcal{G}_{sen}$ can be generated as

$$\mathcal{G}_{sen} = r_{=}\Big(\bigcup \pi_{\delta=0}(\pi_{sen}(\mathcal{E}))\Big), \tag{7.24}$$

such that a sentence similarity function $sim : S \times S \to \mathbb{R}$ allows the derivation of sentence similarity from the context of a sentence s, that can in turn be extracted by the operation $\pi_{term}(\pi_{\delta=0}(\sigma_{v.core=s}(\mathcal{E})))$. Then, subsequent graph centrality computations are performed directly on the reduced graph. In the example of LexRank, these are based on the statistics of the vector space model discussed above.

Some more recent summarization approaches rely directly on a hypergraph representation of sentences [20]. As a result, these approaches can be replicated even more easily. In these cases, only a projection $\pi_{\delta=0}$ to the primary sentence level along with a set union are required to represent the underlying data in the model of the summarization approach.

7.5.4 Event extraction and detection

Based on the definition of an event as *something that happens at a specific date and location and involves an actor* [46] that we have used as motivation of our event-centric analyses in Chapters 3 and 5, event extraction is ultimately aimed at the efficient detection of person-location-date triples (or their subsets in the case of partial mentions), along with a suitable context. Similarly, much of event detection is based on tracking evolving statistics of entity or term cooccurrences, which are easily extractable from the hypergraph model. For example, extracting actor-location-date triples is equivalent to the reduction of edges to a 3-uniform hypergraph in which edges constitute triples of entity nodes with distinct type. For an occurrence context window of size c, this extraction can be formulated as

$$\mathcal{E}_{triples}(c) = r^{3}_{\neq}(\sigma_{term}(\pi_{\delta \leq c}(\pi_{v.ne \in \{loc,per,dat\}}(\mathcal{E})))). \tag{7.25}$$

A weighting function can be used to extract counts or relevance scores for the occurrences, or for further refinement of the events [179]. For more involved approaches that subsequently aim to perform linguistic analyses on the sentence level, retrieving provenance information for the mentions equates to a simple inclusion of sentence nodes with distance $\delta = 0$ in the expression. Thus, entity triples and occurrence statistics are easily extracted to serve as seeds for any more specialized approach.

7.5.5 Query hypergraph support

As one of the few applications in information retrieval that utilize hypergraphs, query hypergraphs were introduced to model higher-order term dependencies in queries [21]. In principle, they are based on the same intuition as our hypergraph document representation, but are specifically limited to modelling the dependency of terms in the query (or more complex query concepts) as hypergraphs, while the document model is not considered. Naturally, a hypergraph formalization of the documents directly enables the efficient processing of similarly structured hypergraph queries and constitutes the logical next step. Specifically, query hypergraphs as introduced by Bendersky and Croft model hyperedges between terms and documents to facilitate document ranking and retrieval. Thus, retrieving relevant edges from the document collection is enabled implicitly by use of the edge containment relation $\sqsubseteq$ in our model (see Section 7.3.4).

By modelling queries as a set of so called query concepts $\kappa \in \mathcal{K}^Q$, query hypergraphs are constructed from *local edges* and *global edges*. Local edges have the formal structure $e_l = \{\kappa, d\}$ and simply link each concept to the document d. As the authors observe themselves, these edges are not hyperedges but dyadic edges and thus structurally similar to traditional bag-of-word representations. As a result, relevance computations can be processed on the hypergraph document representation according to Equations 7.22 and 7.23, albeit by using count statistics that are different from *count* where necessary.

In contrast, global edges link the entire set of query concepts to the document and are formalized as $e_g = \mathcal{K}^Q \cup \{d\}$. To retrieve documents that are perfect matches, we can thus construct a global query edge e_q in our notation as $e_q := \{(v, 0) \mid v \in e_g\}$ and retrieve the set of matching hyperedges $\mathcal{E}_{global}(e_q) = \{e \in \mathcal{E} \mid e \sqsubseteq e_q\}$. Obviously, ranking documents also requires the retrieval of partial matches, a process which can be formalized based on the hypergraph operators as

$$\mathcal{E}_{global}(\mathcal{K}^Q) = \pi_{v.core \in \mathcal{K}^Q \vee v.type=doc}(\sigma_{v.core \in \mathcal{K}^Q}(\mathcal{E})). \tag{7.26}$$

Note that this sequence of operations does not restrict the cooccurrences of terms to the sentence level, since cross-sentence adjacencies are required for proximity-structure query hyperedges [21]. Intuitively, we are retrieving hyperedges from the hypergraph representation that (partially) match the query hyperedges and then use them for scoring. Adding further query restrictions such as maximum cooccurrence windows through chained projection or selection operations is then a trivial matter. Thus, query hypergraphs can be used directly on top of the hypergraph representation of documents.

7.5.6 Implicit entity networks

Of course, we can also extract implicit networks as introduced in the previous chapters from the hypergraph representation. In contrast to the information retrieval approaches that we considered above, implicit networks benefit directly from window sizes beyond sentence boundaries. Obviously, networks that are based on term distances are equally supported as discussed in Section 7.3.4. Since the implicit network model is based on a dyadic entity graph for a context window size of c sentences, it can be represented in the hypergraph model by using the reduction operation. Thus, we obtain the original dyadic network representation used in Chapter 3 from the hypergraph as

$$\mathcal{G}_{imp}(c) = r_{\omega,\neq}(\pi_{\delta=0}(\mathcal{E}) \cup \pi_{0\leq pos\leq k}(\pi_{\exists v.ne}(\mathcal{E}))) \tag{7.27}$$

where the union combines the set of edges between terms and entities that are only extracted from cooccurrences inside sentences, and the set of entity edges that exceed beyond sentence boundaries. Note the use of $r_{\neq}$ to ensure that edges do not occur between entities of the same type, since we used this approach for the implicit network of Wikipedia. Clearly, this is not a requirement, and any other dyadic reduction function is viable. Also note that nodes with negative position values are ignored to prevent the duplicate counting of long-range cooccurrences across sentence boundaries. The edge weights ω used in the implicit network are then recreated by using the aggregation function

$$\omega(\mathcal{E}_{imp}) := \sum_{(v,w)\in\mathcal{E}_{imp}} \exp(-|\max\{v.p - w.p\} - \min\{v.p - w.p\}|). \tag{7.28}$$

on the set of edges $\mathcal{E}_{imp}$. The resulting dyadic graph is equivalent to the original implicit network and supports all extraction and ranking methods proposed for such a network. Thus, by using the hypergraph formalization, we can reduce the extraction of implicit networks to the operations in the above two equations. Similar approaches that do not use edge weights but rely on discrete edge attributes for the extraction of information networks are of course equally viable, but may be longer in a formal representation.

7.6 Summary and Discussion

In this chapter, we have abstracted and generalized the intuitions from the application of implicit networks in the previous chapters to develop a hypergraph model for representing and querying term cooccurrences in large document collections. In particular, we

have based the model on the premise that hyperedges can encode the implicit relations that are contained in the context of sentences and their contents, not just for pairwise cooccurrences, but especially for higher-order cooccurrences of multiple terms or entities. As a result, we have provided the first such model that represents arbitrary term cooccurrences beyond sentence boundaries. Furthermore, external structured data sources, such as knowledge bases, are now directly included in the representation of unstructured text and are thus treated as first-class citizens of operations on the data.

By bridging this gap between unstructured and structured data, we have come full circle to the reason why it made sense to include entity annotations and entity mentions in the model in the first place: They induce structure in otherwise unstructured textual data. By introducing a set of hyperedge operators that enable the representation of numerous fundamental information retrieval methods in a unified model, we have taken the first step towards making this semi-structured representation available to further methods in a universal notation and framework. Most importantly, we find that the model supports the extraction of true graph representations of the documents for a variety of methods, unlike vector-based representations of terms that only enable the approximate and lossy reconstruction of graph structures after the fact through vector similarity.

Practical implications

For our journalists, data analysts, and FBI investigators, the concepts that we have introduced in this chapter have no immediate implication. They do, however, directly affect the approach that a developer or systems administrator might take, who is tasked with developing or maintaining a system that provides the capabilities that we have introduced in the previous chapters. Beyond the problem of integrating data from heterogeneous sources, a common challenge is the amount of data in such applications, which has a terabyte-scale in the case of the Panama Papers [102], and is unlikely to be much smaller in the FBI's collusion investigation. As a result, the efficient representation of the documents is key to enabling the subsequent analyses. Here, we have provided the first step towards a generalized model that can unify structured and unstructured data sources in support of not only network- and entity-centric analyses, but also numerous traditional retrieval tasks.

Outlook

Naturally, the road beyond this first step is long, and subsequent steps for this hypergraph model have to include the implementation that we only considered in theory. While an

implementation on top of existing data storage architectures is viable, the implementation details are less clear and depend on the chosen architecture. Specifically, the choice of architecture is bound to have a considerable impact on the design, since hypergraph storage databases, relations database, graph databases, or other NoSQL databases are viable options with substantially different benefits and drawbacks. Furthermore, the degree to which edges can or should be materialized ahead of time or during queries may depend on the context of the documents, the amount of annotations and external knowledge sources, and even the language of the documents due to varying sentence lengths. Therefore, an evaluation of a variety of implementation architectures is essential to the creation of a framework that can integrate the represented document collections with the predominant knowledge bases, and to find the optimal framework that provides a seamless augmentation of unstructured text with structured knowledge for IR applications.

8 Summary and Outlook

Entity-centric information retrieval tasks are a key component to the understanding and exploration of documents in many different settings. The variety of entities in these tasks ranges from the traditional named entities, such as persons or locations in news analysis and event detection, over abstract concepts in scientific or scholarly settings, to biomedical entities, such as drugs and diseases in healthcare. A versatile document model that can support such tasks independent of the application domain (or even across application domains) is a necessary step towards generalized entity-centric retrieval, and towards enabling the exchange and application of methods between disciplines. In this final chapter, we summarize our contributions towards this goal, and discuss the advancements that the implicit network model and its extensions provide.

8.1 Summary and Contributions

In Chapter 1, we motivated the necessity for representing a collection of documents, the relation between annotated entities in their context within these documents, and knowledge from external sources jointly in a versatile document model to bridge the gap between structured and unstructured data in entity-centric retrieval tasks. In the following Chapter 2, we gave an overview of established document models and highlighted, for each of these models, why they are inadequate for modelling entity relations, or why their drawbacks prevent them from being usable as a generalized model. Based on these considerations, we then introduced the implicit network model for entity-centric information retrieval and document exploration in the following chapters.

Retrieving and understanding implicit entity relations

As our first contribution, we approached the retrieval, exploration, and understanding of relations between entities that are implicitly contained in a document collection. We began in Chapter 3 by introducing **implicit networks** as a **representation of documents** that

capture not only the content of the documents and individual words or annotated entities, but also the relations between them. Based on the novel concept of cooccurrence distances that we pushed beyond the commonly used context windows and beyond sentence boundaries, we devised a weighting scheme for entity and term relations that supports the **identification of related entities**. By representing these relations as a network and by using a localized ranking of entities, we showed that an **efficient retrieval of entity relations** is possible even for large document collections. For the event participation tasks as prototypical examples of entity relationship extraction in our evaluation, we found the implicit network model to perform better than the baseline of word embeddings.

In the following Chapter 4, we investigated the practical application of implicit networks to the tasks of entity-centric search and exploration, as well as **summarization**. On the example of the fully annotated English Wikipedia, we demonstrated not only that implicit networks support **entity-centric search** operations on millions of documents, but also that they enable such interactions in near real-time. By including structural elements such as sentences and documents as central components of the network in addition to the contained entities and terms, we showed how entities can be described and linked to source articles. Focusing on sentences in the second part of Chapter 4, we tackled the task of **extracting descriptive sentences** for pairs or even sets of entities. Based on these descriptive sentences, we obtained descriptions of **complex entity relations** whose detection is beyond the capabilities of traditional knowledge extraction approaches.

Modeling dynamic entity relations in context

Building on the construction of implicit networks from static document collections, we considered document streams and the entailing necessity for a **dynamic implicit network model** in Chapter 5. Since documents in many applications have publication or relevance dates as metadata, and since sets of such documents should therefore be considered streams rather than collections, we investigated the **streaming aggregation** of entity relations by taking advantage of the inherent stream compatibility of the underlying graph representation of implicit networks. In doing so, we were able to improve upon the implicit network model for static document collections by adding the publication time of documents as a new dimension to the analyses. Differentiating between **entity relations in different contexts** then allowed us to consider the evolution of entity relations over time. Here, we found that word embeddings, while not helpful as an alternative method for discovering relations, are useful as additional attributes of entity relations to distinguish between the diverse contexts of relation mentions.

In Chapter 6 we expanded on the use of temporal metadata that is associated with the documents by including it not only as an attribute for browsing or aggregating the data on a timeline, but as an integral component of the weighting of entity relations. The resulting globally weighted edges in the network then allowed us to identify **globally interesting relations** between entities, and extract **descriptive subgraph structures** around them. In doing so, we capitalized on the descriptive efficacy of graph visualizations for convening concepts and relations quickly. We showed how the extracted subgraphs can serve as intuitive and dynamic replacements of topics in the exploration of documents streams, and discussed their relation to traditional topic models. By extending our entity-centric search interface to support the search and visualization of topics, we demonstrated that this enables the comparison and **exploration of network topics in document streams** at a previously impossible level of interactivity.

Generalizing cooccurrence analysis and storage

For our final contribution in Chapter 7, we took a step back and considered the generalization of implicit networks to a document model that supports not only implicit entity and term relations in particular, but (co-)occurrence-based term and entity relations in general. By formally **modelling cooccurrences of terms as hyperedges**, we showed that even higher-order cooccurrence patterns can be extracted. Based on a set of fundamental **hypergraph operators** that serve as a common basis for retrieval queries, we showed how this hypergraph model can be used to support traditional document models as well as implicit networks. We discussed the implications of such a model and how it could be implemented on top of established storage architectures. With this generalized model, we have therefore provided a **unified representation** not only for several document models and retrieval approaches, but also for entity annotations of any kind in unstructured texts, alongside the corresponding knowledge from structured knowledge bases.

8.2 Practical Implications

In bringing it back to our running example of journalists or data analysts tasked with identifying and following up on relevant or interesting relations hidden within large collections of documents, we now have a better and more encompassing understanding of what implicit network models can bring to the table. While the underlying concept is easy to grasp, the potential is significant. By framing the cooccurrences of entities across multiple mentions in separate documents and across different contexts as one unified weighted

network, the search for information immediately becomes more visual and more relatable to the user. Instead of thinking in terms of lists and retrieval scores, it is now possible to think directly in terms of relations. Of course, as we have shown throughout the previous chapters, the retrieval of such ranked lists is still viable, even in the implicit network model, but they can also be integrated into the graph structure.

For our investigators as end users of this network representation, this entails the potential for constructing a holistic system in which one common graph representation is used to support various retrieval and exploration tasks. In particular, as we have seen on the example of our search interfaces in Chapter 4.3 and 6.5, such an exploration can happen interactively and in near-real time, even as new documents are added. By implementing gazetteer-based entity annotation as an interactive component of such a system, it even becomes possible to grow the annotations (and thus the network) during the exploration by annotating new entities and pushing these annotations as templates to the collection. As a result, even if an investigator starts with an entirely unannotated document collection (or an empty stream) in which everything is initially treated as a term, they can incrementally grow and explore an implicit network representation simultaneously.

Since the implicit network model effectively merges unstructured and structured data, it makes sense to discuss the relation between these data types. On the one hand, it is clear that the semantics of weighted relations in an implicit network cannot be directly compared to those of the discrete relations in knowledge graphs. Thus, not all tasks in which knowledge graphs or relation extraction are useful can also rely on implicit networks as a substitute. On the other hand, it is precisely these differences that make the implicit semantics valuable. While knowledge graphs can (and are) used for tasks in which the weighting and ranking of relations is important, an inherently weighted representation is favourable in these situations. Therefore, while we see some areas (such as disambiguation) where currently used methods stand to benefit from relying on implicit networks instead of knowledge bases for the purpose of determining the context of entities, we see the main benefit in the combination of implicit networks and knowledge graphs.

From the perspective of the investigators in an applied field, this means that it is unlikely for them to start entirely from scratch in practice (even though they could). There is always some available prior knowledge, even if it is just a general knowledge graph with very broad but sparse coverage that is not tailored to the application domain of the investigation, like Wikidata. Based on this initial seed of structure that can be injected into the unstructured documents, it becomes possible to grow the identified bits of knowledge in local repositories that are directly tied into the collection. Since the hypergraph model from

Chapter 7 can be used to model the structured and unstructured data jointly (and likely even on top of the same underlying database architecture), analyses are easy to perform and integrate. As a result, the investigators have structured and unstructured information available at their fingertips in an interconnected web of knowledge and information.

The relation between implicit networks and word embeddings is similar to the relation between implicit networks and knowledge graphs, meaning that they cover different aspects of modelling cooccurrences and work best in conjunction. Word embeddings model the relation between words that *occur in similar contexts* in the data, while implicit networks model relations of words that *cooccur in a joint context*. While the difference may sound subtle, the consequences are not. Where embeddings excel at modelling similarity, implicit networks model relatedness. In particular for entities, this difference directly translates to an increased performance in detecting entity relations, as we have demonstrated in Chapter 3.5. On the other hand, embeddings are necessary and beneficial for modelling the context of joint entity mentions, and are an integral component of our dynamic network model as discussed in Chapter 5.

In the application of implicit networks, this opens up the possibility of benefiting from the best of two worlds. On the one hand, embeddings that are used in conjunction with implicit networks enable a fine-grained distinction between the contexts of relation mentions. On the other hand, the generalized implicit network representation can directly be used to extract relevant cooccurrences from arbitrary window sizes for the training of entity embeddings, which is beneficial to the training process [8]. Thus, while our model contains uncompressed relations between words and entities, and is therefore an orthogonal approach to the modelling of words as an abstract and vectorized representation, it is also closely intertwined with the concept of continuous vector embeddings.

Finally, we note that the investigators in our examples are likely to work predominantly on English texts, due to the international nature of the Panama Papers, or the official language of the U.S. for the investigation into collusion. However, these examples are chosen for their relatability and do not reflect limitations of the model. In fact, especially with Wikidata as a knowledge base, the implicit models that we have described in the previous chapters are entirely language agnostic. While they do depend on language specific resources and NLP tools, there are no inherent limitations beyond the requirement of using texts as input data, which is contained in some logical structure, such as documents, paragraphs, or sentences. Therefore, the implicit network model is designed to support investigators in any one language setting, or even multiple languages in joint collections, due to the role of entities as stitching points between common contexts.

8.3 Outlook

Given the diversity in potential applications of the implicit network model, there are, of course, a number of open questions that warrant further investigation. In particular, we identify three general directions on which to concentrate, namely technical aspects, semantic representations, and an extended application of the model.

Technical aspects

In our view, the next step with a focus on the technical aspects of the model is necessarily a realization of the hypergraph model on top of a database architecture. Due to the design of the hypergraph operators, an implementation that utilizes a relational database architecture seems most feasible, but other options are certainly possible. Both NoSQL databases like MongoDB, and specialized hypergraph storage systems could be viable options. In contrast, specialized graph databases are likely less viable, not least due to the high density of implicit networks, especially if the graph databases are focused on representing paths or patterns, which are not essential components in our experiments with implicit networks. However, while this is an informed conjecture based on our experiments with the data and on our implementations of the search interfaces, it is conjecture nonetheless. Ultimately, realizations of the model will have to be optimized and tested to determine the most efficient solution.

Semantic representation of entity relations

On the theoretical side, the relation to embeddings is of primary interest. While learned representations are typically considered superior to engineered solutions, we found the opposite to be the case for entity relations. This is, not least, owed to the fact that in contrast to vector embeddings, methods for training network embeddings (not to be confused with embeddings of networks) have never been proposed. Thus, there is no known way of training full-scale network representations of entity relations. Of course, the development of such methods for learning network structures would likely stand to improve the performance of a model in which relationship weights are learned directly. In comparison to vector embeddings of words, it would be interesting to see experiments in further application domains to determine in which cases similarity-based embeddings are preferable over relatedness-based networks, or vice versa. The insights from such an evaluation would then help in developing joint approaches that can benefit from both models.

Application and extension

Our final observation concerns the number of potential applications. Since we designed the model to be as versatile as possible without losing the benefits of an entity-centric representation, the applications of this document model are numerous. As a result, the application domains that we have covered in examples and application scenarios in this thesis cover only a fraction of the potential applications and available methods in information retrieval. By proposing a network view on entity-centric retrieval and exploration tasks, we have therefore planted a flag in the tip of an iceberg to give it visibility and provide a vantage point, well aware of the fact that the center of mass lies beneath the surface, waiting to be uncovered.

Glossary

$\mathcal{A}$	A set of aggregated edges
$\mathcal{E}$	A set of edges or hyperedges
$\mathcal{G}$	A graph or network
$\mathcal{H}$	A hypergraph
$\mathcal{M}$	An adjacency matrix of a graph
$\mathcal{V}$	A set of nodes
Σ	The set of possible hyperedges for a set of nodes
D	A set of documents
E	A set of entities
I	A set of (co-)occurrence instances
S	A set of sentences
T	A set of terms
Act	A set of actors/persons or nodes with type actor/person
Dat	A set of dates or nodes with type date
Loc	A set of locations or nodes with type location
Org	A set of organizations or nodes with type organization
δ	Edge attribute denoting a cooccurrence distance
ε	Node attribute denoting a term embedding
η	Node attribute denoting the node type
ι	Node or edge attribute denoting a (co-)occurrence instance
κ	Edge attribute denoting a context embedding
λ	Edge attribute denoting the multiplicity of the edge
ω	Edge attribute denoting an importance weight (undirected)
$\vec{\omega}$	Edge attribute denoting an importance weight (directed)
$\ddot{\omega}$	Edge attribute denoting a global importance weight
π	Projection operator for hyperedges
ϱ	A ranking score function
σ	Selection operator for hyperedges
ς	Sentence index of a (co-)occurrence instance

τ	Node or edge attribute denoting the publication time
θ	A propositional expression
c	Parameter denoting the extraction window size
coh	Cohesion score used in edge ranking
$core$	Node attribute denoting the core of a node in a hypergraph
len	Node attribute denoting the length of a sentence
n	Parameter denoting the number of relevant terms
pos	Node attribute denoting the sentence position
r	Reduction operator for hyperedges
th	Parameter denoting an edge aggregation threshold value
win	Context window around two nodes

References

1. A. Abujabal and K. Berberich. "Important Events in the Past, Present, and Future". In: *Proceedings of the 24th International Conference on World Wide Web (WWW), Companion Volume*. 2015, pp. 1315–1320. DOI: 10.1145/2740908.2741692.
2. B. Adams, G. McKenzie, and M. Gahegan. "Frankenplace: Interactive Thematic Mapping for Ad Hoc Exploratory Search". In: *Proceedings of the 24th International Conference on World Wide Web (WWW)*. 2015, pp. 12–22. DOI: 10.1145/2736277.2741137.
3. N. Aggarwal, K. Asooja, H. Ziad, and P. Buitelaar. "Who are the American Vegans related to Brad Pitt?: Exploring Related Entities". In: *Proceedings of the 24th International Conference on World Wide Web (WWW), Companion Volume*. 2015, pp. 151–154. DOI: 10.1145/2740908.2742851.
4. A. Ahmed, Q. Ho, J. Eisenstein, E. P. Xing, A. J. Smola, and C. H. Teo. "Unified Analysis of Streaming News". In: *Proceedings of the 20th International Conference on World Wide Web (WWW)*. 2011, pp. 267–276. DOI: 10.1145/1963405.1963445.
5. M. Al-Dhelaan. "StarSum: A Simple Star Graph for Multi-document Summarization". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2015, pp. 715–718. DOI: 10.1145/2766462.2767790.
6. J. Allan. *Topic Detection and Tracking: Event-based Information Organization*. Vol. 12. Springer Publishing Company, 2012. DOI: 10.1007/978-1-4615-0933-2.
7. J. Allan, R. Papka, and V. Lavrenko. "On-Line New Event Detection and Tracking". In: *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1998, pp. 37–45. DOI: 10.1145/290941.290954.
8. S. Almasian, A. Spitz, and M. Gertz. "Word Embeddings for Entity-annotated Texts". In: *Advances in Information Retrieval - 41st European Conference on IR Research (ECIR)*. 2019.

9. O. Alonso and K. Shiells. “Timelines as Summaries of Popular Scheduled Events”. In: *Proceedings of the 22nd International Conference on World Wide Web (WWW), Companion Volume*. 2013, pp. 1037–1044. DOI: 10.1145/2487788.2488114.

10. F. Alvanaki, S. Michel, K. Ramamritham, and G. Weikum. “See What’s enBlogue: Real-time Emergent Topic Identification in Social Media”. In: *Proceedings of the 15th International Conference on Extending Database Technology (EDBT)*. 2012, pp. 336–347. DOI: 10.1145/2247596.2247636.

11. E. Amitay and C. Paris. “Automatically Summarising Web Sites - Is There A Way Around It?” In: *Proceedings of the 9th ACM CIKM International Conference on Information and Knowledge Management (CIKM)*. 2000, pp. 173–179. DOI: 10.1145/354756.354816.

12. M. Arenas, B. C. Grau, E. Kharlamov, S. Marciuska, D. Zheleznyakov, and E. Jiménez-Ruiz. “SemFacet: Semantic Faceted Search over Yago”. In: *Proceedings of the 23rd International Conference on World Wide Web (WWW), Companion Volume*. 2014, pp. 123–126. DOI: 10.1145/2567948.2577011.

13. M. Atkinson and E. van der Goot. “Near Real Time Information Mining in Multilingual News”. In: *Proceedings of the 18th International Conference on World Wide Web (WWW)*. 2009, pp. 1153–1154. DOI: 10.1145/1526709.1526903.

14. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. “DBpedia: A Nucleus for a Web of Open Data”. In: *6th International Semantic Web Conference (ISWC) and 2nd Asian Semantic Web Conference (ASWC)*. 2007, pp. 722–735. DOI: 10.1007/978-3-540-76298-0_52.

15. P. Baker. “The Shapes of Collocation”. *International Journal of Corpus Linguistics* 21:2, 2016, pp. 139–164. DOI: https://doi.org/10.1075/ijcl.21.2.01bak.

16. M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. “Open Information Extraction from the Web”. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*. 2007, pp. 2670–2676. URL: http://ijcai.org/Proceedings/07/Papers/429.pdf.

17. A.-L. Barabási. *Linked: The New Science of Networks*. Perseus Publishing, 2002.

18. H. Bast, F. Bäurle, B. Buchhold, and E. Haußmann. “Semantic Full-text Search with Broccoli”. In: *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2014, pp. 1265–1266. DOI: 10.1145/2600428.2611186.

19. A. Bellaachia and M. Al-Dhelaan. "HG-Rank: A Hypergraph-based Keyphrase Extraction for Short Documents in Dynamic Genre". In: *Proceedings of the 4th Workshop on Making Sense of Microposts co-located with the 23rd International World Wide Web Conference (WWW)*. 2014, pp. 42–49. URL: http://ceur-ws.org/Vol-1141/paper_06.pdf.

20. A. Bellaachia and M. Al-Dhelaan. "Multi-document Hyperedge-based Ranking for Text Summarization". In: *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM)*. 2014, pp. 1919–1922. DOI: 10.1145/2661829.2662036.

21. M. Bendersky and W. B. Croft. "Modeling Higher-order Term Dependencies in Information Retrieval Using Query Hypergraphs". In: *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2012, pp. 941–950. DOI: 10.1145/2348283.2348408.

22. Y. Bengio, R. Ducharme, and P. Vincent. "A Neural Probabilistic Language Model". In: *Advances in Neural Information Processing Systems 13 (NIPS)*. 2000, pp. 932–938. URL: http://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.

23. C. Berge. *Graphs and Hypergraphs*. North-Holland Publishing, 1973.

24. C. Berge. *Hypergraphs: Combinatorics of Finite Sets*. Vol. 45. Elsevier, 1984.

25. F. Biadsy, J. Hirschberg, and E. Filatova. "An Unsupervised Approach to Biography Production Using Wikipedia". In: *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2008, pp. 807–815. URL: http://www.aclweb.org/anthology/P08-1092.

26. R. Blanco and C. Lioma. "Graph-based Term Weighting for Information Retrieval". *Inf. Retr.* 15:1, 2012, pp. 54–92. DOI: 10.1007/s10791-011-9172-x.

27. R. Blanco and H. Zaragoza. "Finding Support Sentences for Entities". In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2010, pp. 339–346. DOI: 10.1145/1835449.1835507.

28. D. M. Blei. "Probabilistic Topic Models". *Commun. ACM* 55:4, 2012, pp. 77–84. DOI: 10.1145/2133806.2133826.

29. D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. "Hierarchical Topic Models and the Nested Chinese Restaurant Process". In: *Advances in Neural Information Processing Systems 16 (NIPS)*. 2003, pp. 17–24. URL: http://papers.nips.cc/paper/2466-hierarchical-topic-models-and-the-nested-chinese-restaurant-process.

30. D. M. Blei and J. D. Lafferty. "Dynamic Topic Models". In: *Proceedings of the 23rd International Conference on Machine Learning (ICML)*. 2006, pp. 113–120. DOI: 10.1145/1143844.1143859.

31. D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation". *Journal of Machine Learning Research* 3, 2003, pp. 993–1022. URL: http://www.jmlr.org/papers/v3/blei03a.html.

32. K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. "Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 2008, pp. 1247–1250. DOI: 10.1145/1376616.1376746.

33. *Bootstrap: A HTML, CSS, and JavaScript Mobile Web Development Framework*. 2011. URL: http://getbootstrap.com.

34. M. Bostock, V. Ogievetsky, and J. Heer. "D^3 Data-Driven Documents". *IEEE Trans. Vis. Comput. Graph.* 17:12, 2011, pp. 2301–2309. DOI: 10.1109/TVCG.2011.185.

35. V. Brezina, T. McEnery, and S. Wattam. "Collocations in Context: A New Perspective on Collocation Networks". *International Journal of Corpus Linguistics* 20:2, 2015, pp. 139–173. DOI: https://doi.org/10.1075/ijcl.20.2.01bre.

36. J. P. Callan. "Document Filtering With Inference Networks". In: *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1996, pp. 262–269. DOI: 10.1145/243199.243273.

37. R. Campos, G. Dias, A. M. Jorge, and A. Jatowt. "Survey of Temporal Information Retrieval and Related Applications". *ACM Comput. Surv.* 47:2, 2014, 15:1–15:41. DOI: 10.1145/2619088.

38. R. F. i. Cancho and R. V. Solé. "The Small World of Human Language". *Proceedings of the Royal Society of London B: Biological Sciences* 268:1482, 2001, pp. 2261–2265. DOI: 10.1098/rspb.2001.1800.

39. A. Ceroni, U. Gadiraju, J. Matschke, S. Wingert, and M. Fisichella. "Where the Event Lies: Predicting Event Occurrence in Textual Documents". In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2016, pp. 1157–1160. DOI: 10.1145/2911451.2911452.

40. A. X. Chang, V. I. Spitkovsky, C. D. Manning, and E. Agirre. "A Comparison of Named-Entity Disambiguation and Word Sense Disambiguation". In: *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*. 2016. URL: http://www.lrec-conf.org/proceedings/lrec2016/summaries/685.html.

41. J. Chang, J. L. Boyd-Graber, S. Gerrish, C. Wang, and D. M. Blei. "Reading Tea Leaves: How Humans Interpret Topic Models". In: *Advances in Neural Information Processing Systems 22 (NIPS)*. 2009, pp. 288–296. URL: http://papers.nips.cc/paper/3700-reading-tea-leaves-how-humans-interpret-topic-models.

42. W. Chen, E. Fosler-Lussier, N. Xiao, S. Raje, R. Ramnath, and D. Sui. "A Synergistic Framework for Geographic Question Answering". In: *Seventh International Conference on Semantic Computing (ICSC)*. 2013, pp. 94–99. DOI: 10.1109/ICSC.2013.25.

43. S. Chhabra and S. Bedathur. "Towards Generating Text Summaries for Entity Chains". In: *Advances in Information Retrieval - 36th European Conference on IR Research (ECIR)*. 2014, pp. 136–147. DOI: 10.1007/978-3-319-06028-6_12.

44. M. Choudhury, D. Chatterjee, and A. Mukherjee. "Global Topology of Word Co-occurrence Networks: Beyond the Two-regime Power-law". In: *23rd International Conference on Computational Linguistics (COLING), Posters Volume*. 2010, pp. 162–170. URL: http://aclweb.org/anthology/C/C10/C10-2019.pdf.

45. C. Christodoulopoulos, S. Goldwater, and M. Steedman. "Two Decades of Unsupervised POS Induction: How Far Have We Come?" In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2010, pp. 575–584. URL: http://www.aclweb.org/anthology/D10-1056.

46. C. Cieri, S. Strassel, D. Graff, N. Martey, K. Rennert, and M. Liberman. "Corpora for Topic Detection and Tracking". In: *Topic Detection and Tracking: Event-based Information Organization*. Ed. by J. Allan. Springer US, 2002, pp. 33–66. DOI: 10.1007/978-1-4615-0933-2_3.

47. E. F. Codd. "A Relational Model of Data for Large Shared Data Banks". *Commun. ACM* 13:6, 1970, pp. 377–387. DOI: 10.1145/362384.362685.

48. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. "Indexing by Latent Semantic Analysis". *Journal Of The American Society For Information Science* 41:6, 1990, pp. 391–407.

49. F. Diaz, B. Mitra, and N. Craswell. "Query Expansion with Locally-Trained Word Embeddings". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2016. URL: http://aclweb.org/anthology/P/P16/P16-1035.pdf.

50. G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. M. Strassel, and R. M. Weischedel. "The Automatic Content Extraction (ACE) Program - Tasks, Data, and Evaluation". In: *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*. 2004. URL: http://www.lrec-conf.org/proceedings/lrec2004/pdf/5.pdf.

51. S. Dutta and G. Weikum. "Cross-Document Co-Reference Resolution Using Sample-Based Clustering with Knowledge Enrichment". *TACL* 3, 2015, pp. 15–28. URL: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/522.

52. G. Erkan and D. R. Radev. "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization". *J. Artif. Intell. Res.* 22, 2004, pp. 457–479. DOI: 10.1613/jair.1523.

53. J. Esquivel, D. Albakour, M. Martinez-Alvarez, D. Corney, and S. Moussa. "On the Long-Tail Entities in News". In: *Advances in Information Retrieval - 39th European Conference on IR Research (ECIR)*. 2017, pp. 691–697. DOI: 10.1007/978-3-319-56608-5_67.

54. M. Ester, H. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*. 1996, pp. 226–231. URL: http://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.

55. E. Estrada and J. A. Rodríguez-Velázquez. "Subgraph Centrality and Clustering in Complex Hyper-networks". *Physica A* 364, 2006, pp. 581 –594. DOI: 10.1016/j.physa.2005.12.002.

56. S. Evert. "The Statistics of Word Cooccurrences: Word Pairs and Collocations". PhD thesis. University of Stuttgart, Germany, 2005.

57. T. Falke and I. Gurevych. "Bringing Structure into Summaries: Crowdsourcing a Benchmark Corpus of Concept Maps". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017, pp. 2951–2961. URL: https://aclanthology.info/papers/D17-1320/d17-1320.

58. G. Feher, A. Spitz, and M. Gertz. "Evaluating Word Embeddings for the Prediction of Entity Participation in News Events". In preparation. 2019.

59. A. Feng and J. Allan. "Finding and Linking Incidents in News". In: *Proceedings of the 16th ACM International Conference on Information and Knowledge Management (CIKM)*. 2007, pp. 821–830. DOI: 10.1145/1321440.1321554.

60. M. Filannino and G. Nenadic. "Mining Temporal Footprints from Wikipedia". In: *Proceedings of the 1st AHA!-Workshop on Information Discovery in Text*. 2014, pp. 7–13. URL: http://aclweb.org/anthology/W14-4502.pdf.

61. J. R. Finkel, T. Grenager, and C. D. Manning. "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling". In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*. 2005, pp. 363–370. URL: http://aclweb.org/anthology/P/P05/P05-1045.pdf.

62. J. R. Firth. *Papers in Linguistics, 1934-1951*. Oxford University Press, London, 1957.

63. W. A. Gale, K. W. Church, and D. Yarowsky. "One Sense Per Discourse". In: *Proceedings of the Speech and Natural Language Workshop (HLT)*. 1992. URL: https://aclanthology.info/papers/H92-1045/h92-1045.

64. K. Ganesan. "ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks". *CoRR* abs/1803.01937, 2018. URL: http://arxiv.org/abs/1803.01937.

65. D. Ganguly, D. Roy, M. Mitra, and G. J. F. Jones. "Word Embedding based Generalized Language Model for Information Retrieval". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2015, pp. 795–798. DOI: 10.1145/2766462.2767780.

66. M. J. Gardner, J. Lutes, J. Lund, J. Hansen, D. Walker, E. Ringger, and K. Seppi. "The Topic Browser: An Interactive Tool for Browsing Topic Models". In: *Workshop on Challenges of Data Visualization at the Conference for Advances in Neural Information Processing Systems (NIPS)*. 2010. URL: https://cseweb.ucsd.edu/~lvdmaaten/workshops/nips2010/papers/gardner.pdf.

67. A. Gattani, D. S. Lamba, N. Garera, M. Tiwari, X. Chai, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. "Entity Extraction, Linking, Classification, and Tagging for Social Media: A Wikipedia-Based Approach". *PVLDB* 6:11, 2013, pp. 1126–1137. URL: http://www.vldb.org/pvldb/vol6/p1126-gattani.pdf.

68. J. Geiß and M. Gertz. "With a Little Help from my Neighbors: Person Name Linking Using the Wikipedia Social Network". In: *Proceedings of the 25th International Conference on World Wide Web (WWW), Companion Volume*. 2016, pp. 985–990. DOI: 10.1145/2872518.2891109.

69. J. Geiß, A. Spitz, and M. Gertz. "Beyond Friendships and Followers: The Wikipedia Social Network". In: *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2015, pp. 472–479. DOI: 10.1145/2808797.2808840.

70. J. Geiß, A. Spitz, J. Strötgen, and M. Gertz. "The Wikipedia Location Network: Overcoming Borders and Oceans". In: *Proceedings of the 9th Workshop on Geographic Information Retrieval (GIR)*. 2015, 2:1–2:3. DOI: 10.1145/2837689.2837694.

71. M. Georgescu, N. Kanhabua, D. Krause, W. Nejdl, and S. Siersdorfer. "Extracting Event-Related Information from Article Updates in Wikipedia". In: *Advances in Information Retrieval - 35th European Conference on IR Research (ECIR)*. 2013, pp. 254–266. DOI: 10.1007/978-3-642-36973-5_22.

72. F. C. Gey, R. R. Larson, N. Kando, J. Machado, and T. Sakai. "NTCIR-GeoTime Overview: Evaluating Geographic and Temporal Search". In: *Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access (NTCIR)*. 2010, pp. 147–153. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings8/NTCIR/01-NTCIR8-OV-GeoTime-GeyF.pdf.

73. F. C. Gey, R. R. Larson, J. Machado, and M. Yoshioka. "NTCIR9-GeoTime Overview - Evaluating Geographic and Temporal Search: Round 2". In: *Proceedings of the 9th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access (NTCIR)*. 2011. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings9/NTCIR/01-NTCIR9-OV-GEOTIME-GeyF.pdf.

74. F. C. Gey, R. R. Larson, M. Sanderson, H. Joho, P. D. Clough, and V. Petras. "GeoCLEF: The CLEF 2005 Cross-Language Geographic Information Retrieval Track Overview". In: *Revised Selected Papers from the 6th Workshop of the Cross-Language Evalution Forum (CLEF)*. 2005, pp. 908–919. DOI: 10.1007/11878773_101.

75. Y. Goldberg. "A Primer on Neural Network Models for Natural Language Processing". *J. Artif. Intell. Res.* 57, 2016, pp. 345–420. DOI: 10.1613/jair.4992.

76. S. Gottschalk and E. Demidova. "EventKG: A Multilingual Event-Centric Temporal Knowledge Graph". In: *15th Extended Semantic Web Conference (ESWC)*. 2018, pp. 272–287. DOI: 10.1007/978-3-319-93417-4_18.

77. B. Gretarsson, J. O'Donovan, S. Bostandjiev, T. Höllerer, A. U. Asuncion, D. Newman, and P. Smyth. "TopicNets: Visual Analysis of Large Text Corpora with Topic Modeling". *ACM TIST* 3:2, 2012, 23:1–23:26. DOI: 10.1145/2089094.2089099.

78. S. T. Gries. "50-something Years of Work on Collocations: What Is or Should Be Next..." *International Journal of Corpus Linguistics* 18:1, 2013, pp. 137–166. DOI: 10.1075/ijcl.18.1.09gri.

79. R. Grishman and B. Sundheim. "Message Understanding Conference-6: A Brief History". In: *16th International Conference on Computational Linguistics (COLING)*. 1996, pp. 466–471. URL: http://aclweb.org/anthology/C96-1079.

80. O. Gross, A. Doucet, and H. Toivonen. "Document Summarization Based on Word Associations". In: *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2014, pp. 1023–1026. DOI: 10.1145/2600428.2609500.

81. B. Grün and K. Hornik. "topicmodels: An R Package for Fitting Topic Models". *Journal of Statistical Software* 40:13, 2011, pp. 1–30. DOI: 10.18637/jss.v040.i13.

82. A. Guisepi. *History World: On This Day in History*. [accessed October 2, 2015]. URL: http://history-world.org/ontd.htm.

83. D. Gupta and K. Berberich. "Identifying Time Intervals of Interest to Queries". In: *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM)*. 2014, pp. 1835–1838. DOI: 10.1145/2661829.2661927.

84. D. Gupta, J. Strötgen, and K. Berberich. "DIGITALHISTORIAN: Search & Analytics Using Annotations". In: *Proceedings of the 3rd HistoInformatics Workshop on Computational History (HistoInformatics 2016) co-located with the Digital Humanities conference (DH)*. 2016, pp. 5–10. URL: http://ceur-ws.org/Vol-1632/paper_1.pdf.

85. D. Gupta, J. Strötgen, and K. Berberich. "EventMiner: Mining Events from Annotated Documents". In: *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval (ICTIR)*. 2016, pp. 261–270. DOI: 10.1145/2970398.2970411.

86. X. Han and L. Sun. "An Entity-Topic Model for Entity Linking". In: *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 2012, pp. 105–115. URL: http://www.aclweb.org/anthology/D12-1010.

87. S. Heindorf, M. Potthast, B. Stein, and G. Engels. "Towards Vandalism Detection in Knowledge Bases: Corpus Construction and Analysis". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2015, pp. 831–834. DOI: 10.1145/2766462.2767804.

88. B. Heintz and A. Chandra. "Beyond Graphs: Toward Scalable Hypergraph Analysis Systems". *SIGMETRICS Performance Evaluation Review* 41:4, 2014, pp. 94–97. DOI: 10.1145/2627534.2627563.

89. B. Heintz, S. Singh, C. Tesdahl, and A. Chandra. *MESH: A Flexible Distributed Hypergraph Processing System*. Technical report. Department of Computer Science and Engineering, University of Minnesota, 2016. URL: https://www.cs.umn.edu/sites/cs.umn.edu/files/tech_reports/16-038_0.pdf.

90. J. Hirschberg and C. D. Manning. "Advances in Natural Language Processing". *Science* 349:6245, 2015, pp. 261–266. DOI: 10.1126/science.aaa8685.

91. G. Hirst. *Anaphora in Natural Language Understanding: A Survey*. Vol. 119. Lecture Notes in Computer Science. Springer, 1981. DOI: 10.1007/3-540-10858-0.

92. P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC Press, 2010. URL: http://www.semantic-web-book.org/.

93. J. Hoffart, L. Del Corro, and G. Weikum. *The Ambiverse Natural Language Understanding Suite*. 2016. URL: https://github.com/ambiverse-nlu.

94. J. Hoffart, D. Milchevski, and G. Weikum. "AESTHETICS: Analytics with Strings, Things, and Cats". In: *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM)*. 2014, pp. 2018–2020. DOI: 10.1145/2661829.2661835.

95. J. Hoffart, D. Milchevski, and G. Weikum. "STICS: Searching with Strings, Things, and Cats". In: *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2014, pp. 1247–1248. DOI: 10.1145/2600428.2611177.

96. J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, and G. Weikum. "YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages". In: *Proceedings of the 20th International Conference on World Wide Web (WWW), Companion Volume*. 2011, pp. 229–232. DOI: 10.1145/1963192.1963296.

97. J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. "Robust Disambiguation of Named Entities in Text". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2011, pp. 782–792. URL: http://www.aclweb.org/anthology/D11-1072.

98. L. Hong, D. Yin, J. Guo, and B. D. Davison. "Tracking Trends: Incorporating Term Volume into Temporal Topic Models". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2011, pp. 484–492. DOI: 10.1145/2020408.2020485.

99. Y. Hu, J. L. Boyd-Graber, B. Satinoff, and A. Smith. "Interactive Topic Modeling". *Machine Learning* 95:3, 2014, pp. 423–469. DOI: 10.1007/s10994-013-5413-0.

100. J. Huang, R. Zhang, and J. X. Yu. "Scalable Hypergraph Learning and Processing". In: *IEEE International Conference on Data Mining (ICDM)*. 2015, pp. 775–780. DOI: 10.1109/ICDM.2015.33.

101. T. Huet, J. Biega, and F. M. Suchanek. "Mining History with Le Monde". In: *Proceedings of the Workshop on Automated Knowledge Base Construction (AKBC) co-located with the International Conference on Information and Knowledge Management (CIKM)*. 2013, pp. 49–54. DOI: 10.1145/2509558.2509567.

102. International Consortium of Investigative Journalists. *Panama Papers*. [accessed January 21, 2019]. URL: https://panamapapers.sueddeutsche.de/en/.

103. A. Jatowt, C. A. Yeung, and K. Tanaka. "Estimating Document Focus Time". In: *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM)*. 2013, pp. 2273–2278. DOI: 10.1145/2505515.2505655.

104. S. Julinda, C. Boden, and A. Akbik. "Extracting a Repository of Events and Event References from News Clusters". In: *Proceedings of the 1st AHA!-Workshop on Information Discovery in Text*. 2014, pp. 14–18. URL: http://aclweb.org/anthology/W14-4503.pdf.

105. N. Kanhabua and W. Nejdl. "On the Value of Temporal Anchor Texts in Wikipedia". In: *Workshop on Temporal, Social and Spatially Aware Information Access (TAIA) co-located with the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2014.

106. N. Kanhabua, S. Romano, and A. Stewart. "Identifying Relevant Temporal Expressions for Real-world Events". In: *Workshop on Temporal, Social and Spatially Aware Information Access (TAIA) co-located with the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2012.

107. K. Kapoor, D. Sharma, and J. Srivastava. "Weighted Node Degree Centrality for Hypergraphs". In: *Proceedings of the 2nd IEEE Network Science Workshop (NSW)*. 2013, pp. 152–155. DOI: 10.1109/NSW.2013.6609212.

108. H. D. Kim, M. Castellanos, M. Hsu, C. Zhai, U. Dayal, and R. Ghosh. "Ranking Explanatory Sentences for Opinion Summarization". In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2013, pp. 1069–1072. DOI: 10.1145/2484028.2484172.

109. O. Kolomiyets and M. Moens. "A Survey on Question Answering Technology from an Information Retrieval Perspective". *Inf. Sci.* 181:24, 2011, pp. 5412–5434. DOI: 10.1016/j.ins.2011.07.047.

110. S. A. Kripke. *Naming and Necessity*. Harvard University Press, 1980.

111. A. Kutuzov and E. Kuzmenko. "Cross-Lingual Trends Detection for Named Entities in News Texts with Dynamic Neural Embedding Models". In: *Proceedings of the 1st International Workshop on Recent Trends in News Information Retrieval (NewsIR) co-located with the 38th European Conference on Information Retrieval (ECIR)*. 2016, pp. 27–32. URL: http://ceur-ws.org/Vol-1568/paper5.pdf.

112. E. Kuzey, J. Vreeken, and G. Weikum. "A Fresh Look on Knowledge Bases: Distilling Named Events from News". In: *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM)*. 2014, pp. 1689–1698. DOI: 10.1145/2661829.2661984.

113. E. Kuzey and G. Weikum. "Extraction of Temporal Facts and Events from Wikipedia". In: *Proceedings of the 21st International Conference on World Wide Web (WWW), Companion Volume*. 2012, pp. 25–32. DOI: 10.1145/2169095.2169101.

114. G. Leban, B. Fortuna, J. Brank, and M. Grobelnik. "Event Registry: Learning About World Events from News". In: *Proceedings of the 23rd International Conference on World Wide Web (WWW), Companion Volume*. 2014, pp. 107–110. DOI: 10.1145/2567948.2577024.

115. J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. "DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia". *Semantic Web* 6:2, 2015, pp. 167–195. DOI: 10.3233/SW-140134.

116. O. Levy and Y. Goldberg. "Dependency-Based Word Embeddings". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL), Volume 2: Short Papers*. 2014, pp. 302–308. URL: http://aclweb.org/anthology/P/P14/P14-2050.pdf.

117. H. Li. "Social Network Extraction and Exploration of Historic Correspondences". PhD thesis. University of Heidelberg, Germany, 2018.

118. W. Liang. "Spectra of English Evolving Word Co-occurrence Networks". *Physica A: Statistical Mechanics and its Applications* 468, 2017, pp. 802–808. DOI: 10.1016/j.physa.2016.11.096.

119. W. Liang, Y. Shi, C. K. Tse, J. Liu, Y. Wang, and X. Cui. "Comparison of Co-occurrence Networks of the Chinese and English Languages". *Physica A: Statistical Mechanics and its Applications* 388:23, 2009, pp. 4901–4909. DOI: 10.1016/j.physa.2009.07.047.

120. C.-Y. Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Proceedings of the Text Summarization Branches Out Workshop co-located with the Annual Meeting of the Association for Computational Linguistics (ACL)*. 2004. URL: http://www.aclweb.org/anthology/W04-1013.

121. E. Lloret and M. Palomar. "Text Summarisation in Progress: A Literature Review". *Artif. Intell. Rev.* 37:1, 2012, pp. 1–41. DOI: 10.1007/s10462-011-9216-z.

122. L. Lloyd, D. Kechagias, and S. Skiena. "Lydia: A System for Large-Scale News Analysis". In: *Proceedings of the 12th International Conference on String Processing and Information Retrieval (SPIRE)*. 2005, pp. 161–166. DOI: 10.1007/11575832_18.

123. J. B. Lovins. "Development of a Stemming Algorithm". *Mech. Translat. & Comp. Linguistics* 11:1-2, 1968, pp. 22–31. URL: http://www.mt-archive.info/MT-1968-Lovins.pdf.

124. F. Mahdisoltani, J. Biega, and F. M. Suchanek. "YAGO3: A Knowledge Base from Multilingual Wikipedias". In: *Proceedings of the 7th Biennial Conference on Innovative Data Systems Research (CIDR)*. 2015. URL: http://cidrdb.org/cidr2015/Papers/CIDR15_Paper1.pdf.

125. C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

126. C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL), System Demonstrations*. 2014, pp. 55–60. URL: http://aclweb.org/anthology/P/P14/P14-5010.pdf.

127. M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. "Building a Large Annotated Corpus of English: The Penn Treebank". *Computational Linguistics* 19:2, 1993, pp. 313–330. URL: http://aclweb.org/anthology/J93-2004.pdf.

128. S. McKeown, M. Buivys, and L. Azzopardi. "InfoScout: An Interactive, Entity Centric, Person Search Tool". In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2016, pp. 1113–1116. DOI: 10.1145/2911451.2911468.

129. M. Mesgar and M. Strube. "Lexical Coherence Graph Modeling Using Word Embeddings". In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*. 2016, pp. 1414–1423. URL: http://aclweb.org/anthology/N/N16/N16-1167.pdf.

130. R. Mihalcea and P. Tarau. "TextRank: Bringing Order into Text". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2004, pp. 404–411. URL: http://www.aclweb.org/anthology/W04-3252.

131. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26 (NIPS)*. 2013, pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.

132. G. A. Miller. "WordNet: A Lexical Database for English". *Commun. ACM* 38:11, 1995, pp. 39–41. DOI: 10.1145/219717.219748.

133. G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. "Introduction to WordNet: An On-line Lexical Database". *International Journal of Lexicography* 3:4, 1990, pp. 235–244. DOI: 10.1093/ijl/3.4.235.

134. A. Minard, M. Speranza, E. Agirre, I. Aldabe, M. van Erp, B. Magnini, G. Rigau, and R. Urizar. "SemEval-2015 Task 4: TimeLine: Cross-Document Event Ordering". In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval) co-located with the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*. 2015, pp. 778–786. URL: http://aclweb.org/anthology/S/S15/S15-2132.pdf.

135. A. Mishra and K. Berberich. "Event Digest: A Holistic View on Past Events". In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2016, pp. 493–502. DOI: 10.1145/2911451.2911526.

136. A. Mishra and K. Berberich. "EXPOSÉ: EXploring Past news for Seminal Events". In: *Proceedings of the 24th International Conference on World Wide Web (WWW), Companion Volume*. 2015, pp. 223–226. DOI: 10.1145/2740908.2742844.

137. A. Mishra, D. Milchevski, and K. Berberich. "Linking Wikipedia Events to Past News". In: *Workshop on Temporal, Social and Spatially Aware Information Access (TAIA) co-located with the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2014.

138. S. Moran, R. McCreadie, C. Macdonald, and I. Ounis. "Enhancing First Story Detection using Word Embeddings". In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2016, pp. 821–824. DOI: 10.1145/2911451.2914719.

139. D. Nadeau and S. Sekine. "A Survey of Named Entity Recognition and Classification". *Lingvisticæ Investigationes* 30:1, 2007, pp. 3–26. DOI: 10.1075/li.30.1.03nad.

140. R. Nallapati, A. Feng, F. Peng, and J. Allan. "Event Threading Within News Topics". In: *Proceedings of the 13th ACM International Conference on Information and Knowledge Management (CIKM)*. 2004, pp. 446–453. DOI: 10.1145/1031171.1031258.

141. R. Navigli. "Word Sense Disambiguation: A Survey". *ACM Comput. Surv.* 41:2, 2009, 10:1–10:69. DOI: 10.1145/1459352.1459355.

142. A. Nenkova and K. R. McKeown. "A Survey of Text Summarization Techniques". In: *Mining Text Data*. Ed. by C. C. Aggarwal and C. Zhai. Springer US, 2012, pp. 43–76. DOI: 10.1007/978-1-4614-3223-4_3.

143. D. Newman, C. Chemudugunta, and P. Smyth. "Statistical Entity-topic Models". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2006, pp. 680–686. DOI: 10.1145/1150402.1150487.

144. M. Newman. *Networks: An Introduction*. Oxford University Press, 2010.

145. Y. Ni, Q. K. Xu, F. Cao, Y. Mass, D. Sheinwald, H. Zhu, and S. S. Cao. "Semantic Documents Relatedness using Concept Graph Representation". In: *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*. 2016, pp. 635–644. DOI: 10.1145/2835776.2835801.

146. J. Pennington, R. Socher, and C. D. Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://aclweb.org/anthology/D/D14/D14-1162.pdf.

147. J. M. Perea-Ortega, E. Lloret, L. A. U. López, and M. Palomar. "Application of Text Summarization Techniques to the Geographical Information Retrieval Task". *Expert Syst. Appl.* 40:8, 2013, pp. 2966–2974. DOI: 10.1016/j.eswa.2012.12.012.

148. S. Petrovic, M. Osborne, and V. Lavrenko. "Streaming First Story Detection with Application to Twitter". In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*. 2010, pp. 181–189. URL: http://www.aclweb.org/anthology/N10-1021.

149. M. F. Porter. "An Algorithm for Suffix Stripping". *Program* 14:3, 1980, pp. 130–137. DOI: 10.1108/eb046814.

150. D. M. W. Powers. "Applications and Explanations of Zipf's Law". In: *Proceedings of the Joint Conference on New Methods in Language Processing (NeMLaP) and Computational Natural Language Learning (CoNLL)*. 1998, pp. 151–160. URL: http://aclweb.org/anthology/W/W98/W98-1218.pdf.

151. J. Pustejovsky, J. M. Castaño, R. Ingria, R. Saurí, R. J. Gaizauskas, A. Setzer, G. Katz, and D. R. Radev. "TimeML: Robust Specification of Event and Temporal Expressions in Text". In: *Papers from the AAAI Spring Symposium on New Directions in Question Answering*. 2003, pp. 28–34. URL: http://www.timeml.org/publications/timeMLpubs/IWCS-v4.pdf.

152. T. Rattenbury, N. Good, and M. Naaman. "Towards Automatic Extraction of Event and Place Semantics From Flickr Tags". In: *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2007, pp. 103–110. DOI: 10.1145/1277741.1277762.

153. J. Read, R. Dridan, S. Oepen, and L. J. Solberg. "Sentence Boundary Detection: A Long Solved Problem?" In: *24th International Conference on Computational Linguistics (COLING)*. 2012, pp. 985–994. URL: http://aclweb.org/anthology/C/C12/C12-2096.pdf.

154. X. Ren, Z. Wu, W. He, M. Qu, C. R. Voss, H. Ji, T. F. Abdelzaher, and J. Han. "CoType: Joint Extraction of Typed Entities and Relations with Knowledge Bases". In: *Proceedings of the 26th International Conference on World Wide Web (WWW)*. 2017, pp. 1015–1024. DOI: 10.1145/3038912.3052708.

155. A. Ritter, Mausam, O. Etzioni, and S. Clark. "Open Domain Event Extraction from Twitter". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2012, pp. 1104–1112. DOI: 10.1145/2339530.2339704.

156. S. E. Robertson and H. Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". *Foundations and Trends in Information Retrieval* 3:4, 2009, pp. 333–389. DOI: 10.1561/1500000019.

157. F. Rousseau and M. Vazirgiannis. "Graph-of-word and TW-IDF: New Approach to Ad Hoc IR". In: *Proceedings of the 22rd ACM International Conference on Information and Knowledge Management (CIKM)*. 2013, pp. 59–68. DOI: 10.1145/2505515.2505671.

158. R. S. Roy, R. Katare, N. Ganguly, S. Laxman, and M. Choudhury. "Discovering and Understanding Word Level User Intent in Web Search Queries". *J. Web Sem.* 30, 2015, pp. 22–38. DOI: 10.1016/j.websem.2014.07.010.

159. E. Sandhaus. “The New York Times Annotated Corpus”. *Linguistic Data Consortium, Philadelphia* 6:12, 2008, e26752.

160. A. D. Sarma, A. Jain, and C. Yu. “Dynamic Relationship and Event Discovery”. In: *Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM)*. 2011, pp. 207–216. DOI: 10.1145/1935826.1935867.

161. A. Schmidt, J. Hoffart, D. Milchevski, and G. Weikum. “Context-Sensitive Auto-Completion for Searching with Entities and Categories”. In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2016, pp. 1097–1100. DOI: 10.1145/2911451.2911461.

162. T. Schnabel, I. Labutov, D. M. Mimno, and T. Joachims. “Evaluation Methods for Unsupervised Word Embeddings”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2015, pp. 298–307. URL: http://aclweb.org/anthology/D/D15/D15-1036.pdf.

163. E. Schubert, A. Spitz, and M. Gertz. “Exploring Significant Interactions in Live News”. In: *Proceedings of the 2nd International Workshop on Recent Trends in News Information Retrieval (NewsIR) co-located with the 40th European Conference on Information Retrieval (ECIR)*. 2018, pp. 39–44. URL: http://ceur-ws.org/Vol-2079/paper9.pdf.

164. E. Schubert, M. Weiler, and H. Kriegel. “SigniTrend: Scalable Detection of Emerging Topics in Textual Streams by Hashed Significance Thresholds”. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2014, pp. 871–880. DOI: 10.1145/2623330.2623740.

165. V. Setty, S. J. Bedathur, K. Berberich, and G. Weikum. “InZeit: Efficiently Identifying Insightful Time Points”. *PVLDB* 3:2, 2010, pp. 1605–1608. URL: http://www.comp.nus.edu.sg/~vldb2010/proceedings/files/papers/D23.pdf.

166. D. Seyler, T. Dembelova, L. D. Corro, J. Hoffart, and G. Weikum. “A Study of the Importance of External Knowledge in the Named Entity Recognition Task”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 2: Short Papers*. 2018, pp. 241–246. URL: https://aclanthology.info/papers/P18-2039/p18-2039.

167. B. Shi, W. Lam, S. Jameel, S. Schockaert, and K. P. Lai. “Jointly Learning Word Embeddings and Latent Topics”. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2017, pp. 375–384. DOI: 10.1145/3077136.3080806.

168. C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu. "A Survey of Heterogeneous Information Network Analysis". *IEEE Trans. Knowl. Data Eng.* 29:1, 2017, pp. 17–37. DOI: 10.1109/TKDE.2016.2598561.

169. J. Silge and D. Robinson. "tidytext: Text Mining and Analysis Using Tidy Data Principles in R". *Journal of Statistical Software* 1:3, 2016. DOI: 10.21105/joss.00037.

170. J. Sinclair. *Corpus, Concordance, Collocation*. Oxford University Press, 1991.

171. J. Singh, W. Nejdl, and A. Anand. "Expedition: A Time-Aware Exploratory Search System Designed for Scholars". In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2016, pp. 1105–1108. DOI: 10.1145/2911451.2911465.

172. R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. "Reasoning With Neural Tensor Networks for Knowledge Base Completion". In: *Advances in Neural Information Processing Systems 26 (NIPS)*. 2013, pp. 926–934. URL: http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion.

173. K. Spärck Jones. "A Statistical Interpretation of Term Specificity and its Application in Retrieval". *Journal of Documentation* 60:5, 2004, pp. 493–502. DOI: 10.1108/00220410410560573.

174. A. Spitz, S. Almasian, and M. Gertz. "EVELIN: Exploration of Event and Entity Links in Implicit Networks". In: *Proceedings of the 26th International Conference on World Wide Web (WWW), Companion Volume*. 2017, pp. 273–277. DOI: 10.1145/3041021.3054721.

175. A. Spitz, S. Almasian, and M. Gertz. "TopExNet: Entity-centric Network Topic Exploration in News Streams". In: *Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM)*. 2019. DOI: 10.1145/3289600.3290619.

176. A. Spitz, V. Dixit, L. Richter, M. Gertz, and J. Geiß. "State of the Union: A Data Consumer's Perspective on Wikidata and Its Properties for the Classification and Resolution of Entities". In: *Papers from the Wikipedia Workshop (Wiki) co-located with the International Conference on Web and Social Media (ICWSM)*. 2016. URL: http://aaai.org/ocs/index.php/ICWSM/ICWSM16/paper/view/13200.

177. A. Spitz, G. Feher, and M. Gertz. "Extracting Descriptions of Location Relations from Implicit Textual Networks". In: *Proceedings of the 11th Workshop on Geographic Information Retrieval (GIR)*. 2017, 1:1–1:9. DOI: 10.1145/3155902.3155909.

178. A. Spitz, J. Geiß, and M. Gertz. "So Far Away and Yet so Close: Augmenting Toponym Disambiguation and Similarity with Text-based Networks". In: *Proceedings of the 3rd International Workshop on Managing and Mining Enriched Geo-Spatial Data (GeoRich) co-located with the ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 2016, 2:1–2:6. DOI: 10.1145/2948649.2948651.

179. A. Spitz, J. Geiß, M. Gertz, S. Hagedorn, and K. Sattler. "Refining Imprecise Spatio-temporal Events: A Network-based Approach". In: *Proceedings of the 10th Workshop on Geographic Information Retrieval (GIR)*. 2016, 5:1–5:10. DOI: 10.1145/3003464.3003469.

180. A. Spitz and M. Gertz. "Breaking the News: Extracting the Sparse Citation Network Backbone of Online News Articles". In: *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2015, pp. 274–279. DOI: 10.1145/2808797.2809380.

181. A. Spitz and M. Gertz. "Entity-Centric Topic Extraction and Exploration: A Network-Based Approach". In: *Advances in Information Retrieval - 40th European Conference on IR Research (ECIR)*. 2018, pp. 3–15. DOI: 10.1007/978-3-319-76941-7_1.

182. A. Spitz and M. Gertz. "Exploring Entity-centric Networks in Entangled News Streams". In: *Proceedings of the 27th International Conference on World Wide Web (WWW), Companion Volume*. 2018, pp. 555–563. DOI: 10.1145/3184558.3188726.

183. A. Spitz and M. 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 (SIGIR)*. 2016, pp. 503–512. DOI: 10.1145/2911451.2911529.

184. A. Spitz, A. Gimmler, T. Stoeck, K. A. Zweig, and E.-Á. Horvát. "Assessing Low-Intensity Relationships in Complex Networks". *PloS one* 11:4, 2016, e0152536. DOI: 10.1371/journal.pone.0152536.

185. A. Spitz, J. Strötgen, T. Bögel, and M. 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 (WWW), Companion Volume*. 2015, pp. 1375–1380. DOI: 10.1145/2740908.2741693.

186. R. Sproat, C. Shih, W. Gale, and N. Chang. "A Stochastic Finite-State Word-Segmentation Algorithm for Chinese". *Computational Linguistics* 22:3, 1996, pp. 377–404. URL: http://www.aclweb.org/anthology/J96-3004.

187. T. Steiner. "Bots vs. Wikipedians, Anons vs. Logged-Ins (Redux): A Global Study of Edit Activity on Wikipedia and Wikidata". In: *Proceedings of The International Symposium on Open Collaboration (OpenSym)*. 2014, 25:1–25:7. DOI: 10.1145/2641580.2641613.

188. J. Strötgen and M. Gertz. "Multilingual and Cross-domain Temporal Tagging". *Language Resources and Evaluation* 47:2, 2013, pp. 269–298. DOI: 10.1007/s10579-012-9179-y.

189. J. Strötgen and M. Gertz. "Temporal Tagging on Different Domains: Challenges, Strategies, and Gold Standards". In: *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*. 2012, pp. 3746–3753. URL: http://www.lrec-conf.org/proceedings/lrec2012/summaries/425.html.

190. F. M. Suchanek and G. Weikum. "Knowledge Harvesting in the Big-data Era". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 2013, pp. 933–938. DOI: 10.1145/2463676.2463724.

191. Y. Sun and J. Han. "Mining Heterogeneous Information Networks: A Structural Analysis Approach". *SIGKDD Explorations* 14:2, 2012, pp. 20–28. DOI: 10.1145/2481244.2481248.

192. C. Tan, A. Friggeri, and L. A. Adamic. "Lost in Propagation? Unfolding News Cycles from the Source". In: *Proceedings of the 10th International Conference on Web and Social Media (ICWSM)*. 2016, pp. 378–387. URL: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM16/paper/view/13011.

193. S. Tan, J. Bu, C. Chen, B. Xu, C. Wang, and X. He. "Using Rich Social Media Information for Music Recommendation via Hypergraph Model". *TOMCCAP* 7:Supplement, 2011, p. 22. DOI: 10.1145/2037676.2037679.

194. B. Taneva, M. Kacimi, and G. Weikum. "Finding Images of Difficult Entities in the Long Tail". In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM)*. 2011, pp. 189–194. DOI: 10.1145/2063576.2063608.

195. J. Tang, M. Qu, and Q. Mei. "PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks". In: *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2015, pp. 1165–1174. DOI: 10.1145/2783258.2783307.

196. T. P. Tanon, D. Vrandecic, S. Schaffert, T. Steiner, and L. Pintscher. "From Freebase to Wikidata: The Great Migration". In: *Proceedings of the 25th International Conference on World Wide Web (WWW)*. 2016, pp. 1419–1428. DOI: 10.1145/2872427.2874809.

197. T. Tao and C. Zhai. "An Exploration of Proximity Measures in Information Retrieval". In: *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2007, pp. 295–302. DOI: 10.1145/1277741.1277794.

198. C. Tardy, G. Falquet, and L. Moccozet. "Semantic Enrichment of Places with VGI Sources: A Knowledge Based Approach". In: *Proceedings of the 10th Workshop on Geographic Information Retrieval (GIR)*. 2016, 6:1–6:2. DOI: 10.1145/3003464.3003470.

199. B. E. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. "NewsStand: A New View on News". In: *Proceedings of the 16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (ACM-GIS)*. 2008, 18:1–18:10. DOI: 10.1145/1463434.1463458.

200. K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. "Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network". In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*. 2003. URL: http://aclweb.org/anthology/N/N03/N03-1033.pdf.

201. N. K. Tran, A. Ceroni, N. Kanhabua, and C. Niederée. "Time-travel Translator: Automatically Contextualizing News Articles". In: *Proceedings of the 24th International Conference on World Wide Web (WWW), Companion Volume*. 2015, pp. 247–250. DOI: 10.1145/2740908.2742841.

202. T. A. Tuan, S. Elbassuoni, N. Preda, and G. Weikum. "CATE: Context-aware Timeline for Entity Illustration". In: *Proceedings of the 20th International Conference on World Wide Web (WWW), Companion Volume*. 2011, pp. 269–272. DOI: 10.1145/1963192.1963306.

203. J. P. Turian, L. Ratinov, and Y. Bengio. "Word Representations: A Simple and General Method for Semi-Supervised Learning". In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2010, pp. 384–394. URL: http://www.aclweb.org/anthology/P10-1040.

204. C. J. Van Rijsbergen. "A Theoretical Basis for the Use of Co-occurrence Data in Information Retrieval". *Journal of Documentation* 33:2, 1977, pp. 106–119. DOI: 10.1108/eb026637.

205. *vis.js: A Dynamic Browser based Visualization Library*. 2010. URL: http://visjs.org/.

206. D. Vrandecic and M. Krötzsch. "Wikidata: a Free Collaborative Knowledgebase". *Commun. ACM* 57:10, 2014, pp. 78–85. DOI: 10.1145/2629489.

207. J. B. P. Vuurens, A. P. de Vries, R. Blanco, and P. Mika. "Online News Tracking for Ad-Hoc Queries". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2015, pp. 1047–1048. DOI: 10.1145/2766462.2767872.

208. W. Wang, F. Wei, W. Li, and S. Li. "HyperSum: Hypergraph Based Semi-supervised Sentence Ranking for Query-oriented Summarization". In: *Proceedings of the 18th ACM International Conference on Information and Knowledge Management (CIKM)*. 2009, pp. 1855–1858. DOI: 10.1145/1645953.1646248.

209. Y. Wang, M. Zhu, L. Qu, M. Spaniol, and G. Weikum. "Timely YAGO: Harvesting, Querying, and Visualizing Temporal Knowledge from Wikipedia". In: *Proceedings of the 13th International Conference on Extending Database Technology (EDBT)*. 2010, pp. 697–700. DOI: 10.1145/1739041.1739130.

210. C. Wanstrath. *Mustache: Logic-less Templates*. 2009. URL: http://mustache.github.io.

211. G. Weikum and M. Theobald. "From Information to Knowledge: Harvesting Entities and Relationships from Web Sources". In: *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. 2010, pp. 65–76. DOI: 10.1145/1807085.1807097.

212. S. M. Weiss, N. Indurkhya, and T. Zhang. *Fundamentals of Predictive Text Mining, Second Edition*. Texts in Computer Science. Springer, 2015. DOI: 10.1007/978-1-4471-6750-1.

213. P. Wendel. *Spark - A Micro Framework for Creating Web Applications*. 2011. URL: http://sparkjava.com/.

214. Wikipedia contributors. *Wikipedia Current Events Portal*. [accessed December 28, 2016]. URL: https://en.wikipedia.org/wiki/Portal:Current_events.

215. Wikipedia contributors. *Wikipedia Glossary of Astronomy*. [accessed February 26, 2017]. URL: https://en.wikipedia.org/wiki/Glossary_of_astronomy.

216. Wikipedia contributors. *Wikipedia Glossary of Biology*. [accessed February 26, 2017]. URL: https://en.wikipedia.org/wiki/Glossary_of_biology.

217. Wikipedia contributors. *Wikipedia Glossary of Chemistry*. [accessed February 26, 2017]. URL: https://en.wikipedia.org/wiki/Glossary_of_chemistry_terms.

218. Wikipedia contributors. *Wikipedia Glossary of Geology*. [accessed February 26, 2017]. URL: https://en.wikipedia.org/wiki/Glossary_of_geology.

219. Wikipedia contributors. *Wikipedia List of Cities in Germany*. [accessed July 6, 2017]. URL: https://en.wikipedia.org/wiki/List_of_cities_in_Germany_by_population.

220. Wikipedia contributors. *Wikipedia List of National Capitals*. [accessed July 6, 2017]. URL: https://en.wikipedia.org/wiki/List_of_national_capitals_in_alphabetical_order.

221. M. M. Wolf, A. M. Klinvex, and D. M. Dunlavy. "Advantages to Modeling Relational Data Using Hypergraphs Versus Graphs". In: *IEEE High Performance Extreme Computing Conference (HPEC)*. 2016, pp. 1–7. DOI: 10.1109/HPEC.2016.7761624.

222. W. Yang, J. L. Boyd-Graber, and P. Resnik. "A Discriminative Topic Model using Document Network Structure". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2016. URL: http://aclweb.org/anthology/P/P16/P16-1065.pdf.

223. Y. Yang, T. Pierce, and J. G. Carbonell. "A Study of Retrospective and On-Line Event Detection". In: *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1998, pp. 28–36. DOI: 10.1145/290941.290953.

224. Y. Yang, J. Zhang, J. G. Carbonell, and C. Jin. "Topic-conditioned Novelty Detection". In: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2002, pp. 688–693. DOI: 10.1145/775047.775150.

225. D. Yarowsky. "One Sense per Collocation". In: *Proceedings of the Human Language Technology Workshop (HLT)*. 1993. URL: https://aclanthology.info/papers/H93-1052/h93-1052.

226. S. M. Yimam, H. Ulrich, T. von Landesberger, M. Rosenbach, M. Regneri, A. Panchenko, F. Lehmann, U. Fahrer, C. Biemann, and K. Ballweg. "new/s/leak - Information Extraction and Visualization for Investigative Data Journalists". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), System Demonstrations*. 2016, pp. 163–168. DOI: 10.18653/v1/P16-4028.

227. C. Zhai and S. Massung. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Association for Computing Machinery and Morgan & Claypool, New York, 2016.

228. L. Zhang, M. Färber, and A. Rettinger. "XKnowSearch!: Exploiting Knowledge Bases for Entity-based Cross-lingual Information Retrieval". In: *Proceedings of the 25th ACM International Conference on Information and Knowledge Management (CIKM)*. 2016, pp. 2425–2428. DOI: 10.1145/2983323.2983324.

229. K. Zhao, L. Chen, and G. Cong. "Topic Exploration in Spatio-Temporal Document Collections". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 2016, pp. 985–998. DOI: 10.1145/2882903.2882921.

230. Y. Zhu, Z. Guan, S. Tan, H. Liu, D. Cai, and X. He. "Heterogeneous Hypergraph Embedding for Document Recommendation". *Neurocomputing* 216, 2016, pp. 150–162. DOI: 10.1016/j.neucom.2016.07.030.

231. Y. Zuo, J. Zhao, and K. Xu. "Word Network Topic Model: A Simple but General Solution for Short and Imbalanced Texts". *Knowl. Inf. Syst.* 48:2, 2016, pp. 379–398. DOI: 10.1007/s10115-015-0882-z.

232. K. A. Zweig. "Are Word-Adjacency Networks Networks?" In: *Towards a Theoretical Framework for Analyzing Complex Linguistic Networks*. Ed. by A. Mehler, A. Lücking, S. Banisch, P. Blanchard, and B. Job. Springer, Berlin, Heidelberg, 2016, pp. 153–163. DOI: 10.1007/978-3-662-47238-5_7.

233. S. Zwicklbauer, C. Seifert, and M. Granitzer. "Robust and Collective Entity Disambiguation through Semantic Embeddings". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2016, pp. 425–434. DOI: 10.1145/2911451.2911535.