Inaugural-Dissertation

zur Erlangung der Doktorwürde der

Gesamtfakultät für Mathematik, Ingenieur- und Naturwissenschaften

der

Ruprecht-Karls-Universität
Heidelberg

vorgelegt von

John Ziegler (M.Sc.)

aus Schwäbisch Gmünd

Tag der mündlichen Prüfung:

Temporal, Network-Based Media Analytics

From Model to Application

by

John Ziegler

Supervisors: Prof. Dr. Michael Gertz
Prof. Dr. Christian Schulz

Abstract

Given our evermore complex world, keeping track of important events, developments, and interdependencies is increasingly time-consuming or even infeasible. To a large part, this complexity is caused by the systems' underlying dynamics and connectedness – characteristics found in various domains. Media, characterized by its high volatility and interwoven network of content and actors, is such a domain. Due to its ever-growing importance, not only from an academic but also from a societal perspective, understanding media-related phenomena is of huge importance. However, deriving insights from media data is not trivial, and the mentioned characteristics must be considered. Instead of simply asking a question like *Which topics are currently discussed?*, to gain a holistic perspective, one must also consider the underlying dynamics and ask *Which topics are gaining in popularity?* or *How does the relevance of a topic change over time?*. Similarly, it is insufficient to only investigate individual media actors without considering their social connectedness. To account for these requirements, in this thesis, we leverage temporal, network-based methods for analyzing media data. We do not limit ourselves to the development of novel methodological approaches but also put these into practice by bridging the gap between model and application.

First, a unifying model that allows for coping with heterogeneous data sources is required. Based on the concept of temporal networks, we develop such a model that particularly reflects the data's time-sensitivity and structural interdependencies. Its capabilities are demonstrated in several media analytics studies. These studies include the investigation of trends – a phenomenon that is integral to the dynamics of media. In particular, we focus on examining long-term trends, which keep their large prevalence over longer periods compared to short-lived trends. Also, we connect the analysis of trends with that of social network analysis by investigating the actor-networks underlying trends. Further, we show that the model can also be applied to online conversation analysis. Given its conception based on temporal networks, we can approach respective analysis by incorporating the conversations' content, dynamics, and structural properties. Finally, after laying the theoretical foundation of this work in the form of the proposed model and successfully leveraging it for several analytics use cases, we shift our focus to its technical implementation. For that, we showcase two real-world applications, the EPINetz platform and the TrendTracker app, for the temporal and network-based exploration of media data. Also, the design of such applications at its core requires a performant graph data management and analysis system. Therefore, we benchmark various system setups and discuss an appropriate implementation strategy. In sum, this thesis demonstrates the benefits that come with approaching media analysis from a temporal and network-based perspective. Our contributions are not limited to novel methods and techniques but also tackle challenges that occur when putting these approaches into practice.

Zusammenfassung

In unser zunehmend komplexen Welt ist es beinahe unmöglich, den Überblick über wichtige Ereignisse, Entwicklungen und Zusammenhänge zu behalten. Diese Komplexität ist größtenteils in den systemischen Dynamiken und Interdependenzen begründet – Charakteristika, die in verschiedenen Domänen zu finden sind. Medien, die sich durch eine hohe Volatilität und ein verflochtenes Netzwerk an Inhalten und Akteuren auszeichnen, stellen eine solche Domäne dar. Deren wachsende Bedeutung führt dazu, dass die Untersuchung medialer Phänomene, nicht nur aus akademischer, sondern auch aus gesellschaftlicher Sicht, immer bedeutsamer wird. Allerdings ist das Gewinnen von Erkenntnissen aus Mediendaten nicht trivial. Für eine ganzheitliche Betrachtung genügt eine Frage wie *Welche Themen werden derzeit diskutiert?* nicht aus, sondern die zugrunde liegende Dynamik der Daten muss ebenfalls berücksichtigt werden, zum Beispiel durch Fragen wie *Welche Themen werden immer populärer?* oder *Wie ändert sich die Relevanz eines Themas über die Zeit?*. Ebenso reicht es nicht aus Medienakteure isoliert zu betrachten, ohne deren soziale Beziehungen zu berücksichtigen. Um genannte Dynamik und Interdependenzen miteinzubeziehen, nutzen wir in der vorliegenden Arbeit netzwerkbasierte und zeitsensitive Methoden zur Analyse von Mediendaten. Zunächst ist hierzu ein vereinheitlichendes Modell erforderlich, das die Integration von heterogenen Datenquellen ermöglicht. Basierend auf dem Konzept zeitlich veränderlicher Netzwerke entwickeln wir ein solches Modell. Seine Einsetzbarkeit wird in mehreren Medienanalysen demonstriert. Diese umfassen die Untersuchung von Trends – ein Phänomen, das integraler Bestandteil der Mediendynamik ist. Wir konzentrieren uns insbesondere auf die Untersuchung langfristiger Trends, die im Vergleich zu kurzlebigen Trends über längere Zeiträume hinweg eine hohe Prävalenz aufweisen. Darüber hinaus verbinden wir die Analyse von Trends mit der Analyse sozialer Netzwerke, indem wir die den Trends zugrundeliegenden Akteursnetzwerke untersuchen. Außerdem zeigen wir, dass das Modell auch für die Online-Konversationsanalyse verwendbar ist. Aufgrund seiner Konzeption können wir Analysen durchführen, die den Inhalt, die Dynamik und die Struktur der Konversationen miteinbeziehen. Im Anschluss verlagern wir unseren Fokus auf die technische Umsetzung der entwickelten Methoden. Dazu stellen wir zwei Anwendungen zur zeitlichen und netzwerkbasierten Erkundung von Mediendaten vor, die EPINetz-Plattform und die TrendTracker-App. Das Design solcher Anwendungen erfordert im Kern ein leistungsfähiges System zur Verwaltung und Analyse von Netzwerkdaten. Dazu vergleichen wir verschiedene Systeme und diskutieren eine geeignete Implementierungsstrategie. Zusammenfassend zeigt die vorliegende Arbeit die Vorteile auf, die sich aus zeitsensitiven und netzwerkbasierten Medienanalysen ergeben. Nicht nur entwickeln wir neue Methoden und Techniken, sondern wir gehen auch auf Herausforderungen ein, die bei der Umsetzung dieser Ansätze in die Praxis auftreten.

Acknowledgements

First and foremost, I would like to express my sincere gratitude towards my supervisor, Prof. Dr. Michael Gertz. He gave me the great opportunity to conduct the research leading up to this PhD thesis in his working group at Heidelberg University. Not only did he provide an excellent research environment in terms of facilities, equipment, and administrative support, but he also offered personal guidance and professional advice whenever required. Similarly, thanks go to Prof. Dr. Christian Schulz, who serves as co-supervisor for this thesis. Also, thank you, Prof. Dr. Peter Bastian and Prof. Dr. Jürgen Hesser, for being part of the PhD committee.

Next, I would like to thank the Klaus Tschira Stiftung, which financed the EPINetz project in which framework this thesis has been conducted. This financial foundation has allowed me to concentrate fully on our research objectives.

Further, I want to highlight the great support and helpful exchange with all my colleagues who have accompanied me over the past years, in particular, Dennis Aumiller, Satya Almasian, Jayson Salazar, Ashish Chouhan, Nicolas Reuter, and Philip Hausner – not to forget the delightful and entertaining conversations during the lunch breaks we spent together. This collegial exchange also applies to the current and former members of the EPINetz team, especially Prof. Dr. Wolf J. Schünemann, Tim König, Alexander Brand, Erik Wolfes-Wenker, Marina Walther, Raeesa Yousaf, Robin Khanna, Julian Freyberg, Derya Emer, Marco Gronewold, and Quentin Bukold. The collaboration with such enthusiastic and skilled researchers was not only rewarding but also allowed me to gain insights into research domains different from my own.

It must also be emphasized that this work would not have been possible in this form without the support of several students who assisted me in various ways. In particular, I want to mention Fabian Kneissl and Johannes Sindlinger – I have thoroughly enjoyed our collaboration. In this regard, the technical support provided by Milena Bruseva, Holger Wünsche, and Saif Mandour should also not be forgotten. Similarly, thanks go to Natalia Ulrich and Catherine Proux-Wieland for the administrative help and guidance they provided me whenever necessary.

Even though proofreading a PhD thesis can be a cumbersome task, my fellows have also proven their extraordinary support in this regard. Special thanks go to Satya Almasian, Elias Arnold, and Moritz Althaus for their extensive reviewing efforts and for providing me with so much valuable feedback.

Last but not least, I want to express my great gratitude to my friends and family for their continuous support, love, and companionship – it is a great gift to share my life with you!

"*If you want to go fast, go alone.*
If you want to go far, go together."
— African proverb

Contents

I Introduction

We live in an increasingly complex and constantly evolving world, driven by two fundamental characteristics: *dynamism* and *connectedness* (see Bianconi (2018, pp. 79–82)). However, even if these characteristics are very noticeable today, they are not unique to our time. Instead, they can be seen as fundamental building blocks of our environment. Already in 1769, Gotthold Ephraim Lessing described this phenomenon very aptly (Lessing (1769, p. 140), as cited in Quote Investigator (2022)):

> "In nature everything is connected, everything is interwoven, everything changes with everything, everything merges from one into another."

Named characteristics open up two perspectives on our world: How do things *evolve* and how are they *interrelated*? In order to find ever more precise answers to these questions, our real-world observation methods and modeling approaches must be improved. Nowadays, many observations are available in the form of unstructured datasets. Extracting valuable insights from these datasets and improving our real-world understanding based on these is a key challenge. This work intends to make a contribution to overcoming the stated task.

1.1 Why to Use Temporal Networks?

In line with the characteristics revealed by real-world systems, for the analysis of collected observation data, its dynamism and connectedness have to be taken into account. In this regard, networks as data structure represent a natural choice to model the interrelatedness of objects. Leveraging such networks for an appropriate data model satisfies the first "connectedness" requirement. Further, stated dynamics also need to be incorporated. Therefore, the simple network-based model must be extended to reflect the dynamics of the analyzed data as well. Approaches from the field of *temporal networks* do so. Hence, they satisfy the "dynamism" requirement for sought-after data model and can be employed to study real-world phenomena. In fact, numerous studies have already shown their broad applicability across various domains, such as route planning (Chabini and Lan, 2002; Wang et al., 2015), social network analysis (Hanneke et al., 2010; Huo and Tsotras,

2014), finance (Somin et al., 2020) or biology and medicine (see Hosseinzadeh et al. (2023)). Particularly relevant for this work are use cases from the field of (social) media analytics, as detailed in the following section. Notably, the term "network", as used in this work, must not be understood in a technical sense, e.g., such as used to refer to "computer networks". Instead, it refers to an abstract data structure for the modeling of interrelated objects (see Section 2.1.1).

1.2 Media Analytics as Use Case

The current historic era is termed "Information Age" or "New Media Age"[1]. This naming already highlights the great importance of media in today's world. Given the societal importance of media, from an academic perspective, studying the related phenomena is of great relevance. Traditionally, media has mainly referred to analog content-sharing services like newspapers, radio, or television. With the advent of the World Wide Web, online media in the form of social media platforms, blogs, and streaming services gained significance (see Wikimedia Foundation, Inc. (2023)). As of October 2023, each of the seven biggest social media platforms had over one billion active users each month (Statista, Inc., 2023). Therefore, analyzing and understanding such online media content also becomes more relevant. One might be interested in investigating discussed topics, trends, conversations, or from an actor-centric perspective, e.g., which social media users are very influential or frequently spread information. In the case of social media, Zeng et al. (2010) define its analysis as follows:

> "*Social media analytics* is concerned with developing and evaluating informatics tools and frameworks to collect, monitor, analyze, summarize, and visualize social media data, usually driven by specific requirements from a target application."

With regard to this definition, multiple aspects must be highlighted. First and foremost, the application-driven nature of social media analytics is stated. Most often, the context in which a respective analysis study is conducted determines leveraged techniques and methodology. In this work, we also follow this approach. The development of the temporal, network-based analysis model (see Chapter 3) is primarily driven by use case-related requirements. Additionally, its applicability to study various media-related phenomena is demonstrated, e.g., by investigating trends in Chapter 4 or conversations in Chapter 5.

Most applications presented in this thesis are conducted based on data collected from the Twitter[2] platform. Even though Twitter has recently been renamed to "X", for historic consistency, we keep

[1] Information Age – Wikipedia: https://en.wikipedia.org/wiki/Information_Age (accessed 2023-11-27)
[2] X. It's what's happening / X: https://twitter.com (accessed 2023-11-02)

referring to their information-sharing service and social media platform as "Twitter". Additionally, in subsequent parts, "media analytics" is understood as a generalization of the more specific "*social* media analytics" approach by also taking into account media services without additional social interaction features, e.g., news outlets (without commenting functionality).

EPINetz project

This thesis has been conducted in the context of the EPINetz project[3]. As indicated by its name, "Exploration of Political Information Networks", the project aims to leverage network approaches for the analysis of politically relevant data collected from news and social media (see Section 6.2). Along with that, a web application is developed that allows EPINetz users to explore the gathered data and derive insights. Therefore, by its conceptual design, the EPINetz project aims not only to develop novel theoretical approaches for analyzing political media data but also to put these into practice. In line with this agenda, in this thesis, we bridge the gap between theoretical contributions and actual technical implementations. As such, a model for analyzing media data is not only developed and applied to various use cases, but technical challenges that come with its implementation are also approached.

Even though within the EPINetz project, *political* information is studied in particular, this thesis is not tied to the political domain. Instead, a domain-independent temporal, network-based data analysis model (see Chapter 3) is developed, and its capabilities are demonstrated by means of non-political applications, e.g., the study of trends in Chapter 4. Due to the lack of the author's political domain expertise, politically related work is not included in this thesis. For the development of respective computational methods, this expertise is not necessarily required. However, during the EPINetz project, collaborations with researchers from the political sciences allowed for a transfer of developed computational methods to the political science discipline. Several contributions to the field of computational political science can be mentioned in this regard:

- Wolf J. Schünemann, Alexander Brand, Tim König, and John Ziegler. Leveraging Dynamic Heterogeneous Networks to Study Transnational Issue Publics. The Case of the European COVID-19 Discourse on Twitter. *Frontiers in Sociology*, 7, 2022.

- Tim König, Alexander Brand, Wolf Schünemann, John Ziegler, and Michael Gertz. Wer treibt hier wen an? – Temporale Diskursverschiebungen zwischen News-Agenda und Parteikommunikation auf Twitter. *DVPW Kongress*, 2021.

[3] EPINetz: `https://epinetz.de` (accessed 2023-11-02)

In general, the described interdisciplinary research environment influenced the conducted research efforts in various ways. Most notably, the selected application use cases are strongly motivated by a real-world need, and developed methods are tailored with practical requirements in mind. For example, maintaining high data quality standards is essential in the political science discipline. Therefore, we have made a great effort to comply with these standards. Also, we focused on data analysis methods that guarantee reproducibility and do not reveal black box characteristics.

1.3 Aims and Scope

Even though this work aims to contribute to the research area of media analytics in a way that reaches from theoretical approaches, such as the modeling of analyzed data, to the technical implementation, e.g., how a scalable temporal graph[4] processing and data management system needs to be designed, of course, not all related use cases and application scenarios are covered. Instead, the effort is focused on a subset of topics. This selectivity starts with the leveraged media data. Not the complete spectrum of online and offline media sources is covered, but the emphasis is placed on online news outlets and social media platforms, especially Twitter. Further, semi-structured data is analyzed in particular. This data involves a combination of textual data and metadata. Data of other modes is not considered, such as visual data (e.g., images and videos). Also, concerning the analysis itself, not all aspects are covered. From an abstract perspective, by leveraging the developed temporal, network-based analysis model, the data's dynamism and connectedness characteristics are studied in particular, such as how different content (e.g., social media posts) is related to one another or how it evolves over time. However, the content itself is less studied, e.g., in the form of semantic or sentiment analysis. Finally, concerning the domain of application, even though this thesis originates from a politics-centered context and some of the applications are demonstrated based on political media data, e.g., the study about social media conversations presented in Chapter 5, the theoretical approaches are not tight to the political domain, but can be applied to others as well.

[4] Even though "graphs" are the mathematical representation of the (real world) "networks" (Aleta and Moreno, 2019; Blanco and Lioma, 2012). Both terms are used interchangeably in this work as this is common practice in the field of network science. The same is true for the terms "object", "vertex", and "node", as well as "edge", "link", and "relationship" (see Latora et al. (2017, p. 3)).

1.4 Contributions

Several contributions to different research disciplines are part of this thesis. In the following, these are outlined in line with the order of subsequent thesis chapters. Related publications are listed along with respective contributions:

- Chapter 3: A model for analyzing media data is developed. It is built on concepts known from temporal graph analysis and allows the extraction of informative insights from the analyzed data, taking its time-dependency and structural (i.e., topological) characteristics into account. Further, due to its modeling flexibility, which allows data integration from various sources and its reproducibility capabilities, central requirements arising from the media analysis use case are satisfied. The model's applicability is show-cased in subsequent studies, such as the investigation of actor-networks underlying trends on Twitter presented in Section 4.2. Further, challenges related to the technical implementation of an analysis system based on this data model are approached in Section 6.3. To the best of our knowledge, the developed approaches make up the most comprehensive network-based media analytics framework, and the conducted application studies show how it can successfully contribute towards a better understanding of the highly dynamic and interwoven media landscape.

- Chapter 4: Trends characterize the dynamics of media in general and social media in particular. However, trends are not studied in all their details, but numerous research questions still need to be answered. In this work, we focus on two specific topics. First, long-term trends prevalent over a longer time with respect to typical media information cycles are studied. Methodologies for detecting long-term trends and visualizing respective analysis results are developed for this. Secondly, actor-networks underlying trends are studied, which contributes to a better understanding of the question "Who is behind a trend?". Use case-specific approaches for detecting trend phases, and the temporal sampling of networks are developed. Generally, this chapter contributes to a blend of time series analysis in the form of trends and network analysis. Presented insights are based on the following publications:

  John Ziegler and Michael Gertz. No Mayfly: Detection and Analysis of Long-term Twitter Trends. In *BTW 2023*, pages 353–364. 2023.

  John Ziegler and Michael Gertz. Who Is behind a Trend? Temporal Analysis of Interactions among Trend Participants on Twitter. *Proceedings of the International AAAI Conference on Web and Social Media*, 17:960–969, 2023.

- Chapter 5: With the results presented in this chapter, we contribute to an improved understanding of conversations taking place on social media. We argue that a holistic perspective on conversational data analysis is required for that – structure with regards to network topology, content, and the conversation's dynamics need to be considered. Therefore, various aspects of online conversations are studied. First, the users' posting activity is modeled, and a specific sampling technique for creating temporal network snapshots is developed based on that. This approach allows to investigate the structural evolution of the conversation networks. A novel metric, the temporal Wiener index, is developed and leveraged for that. Finally, the dynamics concerning the hashtags used in conversations are studied. Found insights are made publicly available to the research community:

  John Ziegler, Fabian Kneissl, and Michael Gertz. CODY: A graph-based framework for the analysis of COnversation DYnamics in online social networks. *arXiv preprint arXiv:2310.08140*, 2023.

- Chapter 6: Unlike the previous chapters, this one is focused on realization and technical applicability. Thereby, four main building blocks are covered. First, a temporal network media dataset is presented, which serves as the foundation of subsequently presented implementations and studies. It is also leveraged for various studies presented in the other chapters of this thesis, such as the study of long-term trends presented in Section 4.1. For the discussion of the dataset, a large emphasis is placed on its statistics, network schema, and covered dynamics. Secondly, the developed EPINetz platform is presented. It shall give interested users access to parts of the mentioned media dataset and allows them to explore the dataset in a self-service manner. In line with the developed analysis model, users can explore the data based on temporal and network-centric methods. Thereby, the platform is intended to provide capabilities that support users in improving their media literacy and learning about political topics. Nevertheless, implementing such a system in practice does not come without its challenges. Developing a performant (temporal) graph data management and analytics system is required. By benchmarking various systems and technical setups, we contribute to the research discipline related to the performance analysis of graph analysis systems. Finally, another technical contribution is presented in the form of the TrendTracker application. It allows users to interactively explore the studied long-term trends mentioned above. Network-based and time series visualizations are core to the application and allow to gain insights into the context of analyzed trends, as well as their dynamics. In a broader sense, the TrendTracker application also contributes to improved trend visualization and exploration techniques. Outlined work is rooted in various publications:

John Ziegler, Alexander Brand, Julian Freyberg, Tim König, Wolf Schünemann, Marina Walther, and Michael Gertz. EPINetz: Exploration of Political Information Networks. *INFORMATIK 2021*, pages 1603–1609, 2021.

John Ziegler, Johannes Sindlinger, Marina Walther, and Michael Gertz. TrendTracker: Temporal, network-based exploration of long-term Twitter trends. In *Proceedings of the 2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Forthcoming.

In sum, the contributions of this work cannot be limited to a single research discipline but span from the modeling of media data to a performant implementation of a graph processing system. Also, they range from methodological approaches to more technically-focused insights. Several publications make the lessons learned available to the research community.

1.5 Structure

Structurally, this thesis is separated into seven chapters, including this one. Logically, however, they can be grouped into four main parts. First, the objective of this work is introduced, and the theoretical foundation is provided. Then, methodological contributions and studies are covered before the focus is shifted towards more technical and implementation-centric work. Finally, the thesis is concluded by summarizing the efforts and discussing their limitations. In detail, the following topics are covered in the subsequent chapters: First of all, in Chapter 2, the theoretical background is introduced. This introduction primarily includes concepts from the field of network science, media, and socio-semantic networks. Also, for the technical Chapter 6, different aspects related to database management systems are described. After that, in Chapter 3, a temporal, network-based media analysis model is introduced. For that, its requirements are discussed first. Then, its capabilities concerning the network-based modeling, the modeling of dynamics, network projections, and different aspects related to granularity are discussed. Further, this proposed model is applied to different use cases in the subsequently discussed studies. In Chapter 4, the objectives of these studies are trends. In particular, two aspects of trends as a social media phenomenon are examined: long-term trends and actor-networks behind trends. Next, the model is applied to study online conversations in Chapter 5. Several experiments about the posting activity, the structural evolution of conversation networks and the usage of hashtags are conducted. Furthermore, in Chapter 6, the focus is shifted towards technical objectives such as the leveraged media dataset, the EPINetz platform, which allows interested users to explore politics-related media information, a benchmark on the performance of different graph data management and analysis systems, as well as the TrendTracker application offering advanced trend exploration capabilities. Also, these tech-

nical contributions showcase the applicability of the proposed model (see Chapter 3) in real-world solutions. Finally, Chapter 7 concludes this thesis by summarizing done work and discussing its limitations.

2 Fundamentals

This chapter provides the necessary theoretical basics for the studies and methods presented in subsequent chapters. Thereby, various academic disciplines are covered, and methods from different topics are presented. First and foremost, Section 2.1 introduces fundamental network science concepts, covers different network modeling approaches, explains community detection methods, and highlights approaches for studying the dynamics of networks. Then, in Section 2.2, media platforms are introduced based on the Twitter example, and a specific focus is placed on working with data collected from such platforms, e.g., by examining issues related to data quality. Furthermore, a specific network modeling approach called "socio-semantic networks" is described in Section 2.3. It follows the idea that for a comprehensive understanding (social) media data needs to be analyzed from an actor-centric *and* a semantic perspective. In the subsequent chapters, we consider both of these perspectives. Finally, in a more technically-focused Section 2.4, various concepts related to systems for managing data are explained. These concepts are particularly relevant for the benchmark on graph data management and analysis presented in Section 6.3.

2.1 Network Science

In this section, the theoretical background needed for the subsequent chapters with regard to the network science discipline is covered. For that, Section 2.1.1 first introduces the theoretical foundations of graph theory. Important concepts are explained, such as various graph representations (e.g., adjacency matrices and edge lists), random networks, weighted networks, and centrality measures. Further, in Section 2.1.2, different aspects of community detection and evolution are examined. Then, in the following, two specific network modeling approaches are discussed in detail: heterogeneous information networks (HINs) in Section 2.1.3 and multilayer networks in Section 2.1.4. Finally, Section 2.1.5 covers various aspects related to the dynamics of networks, such as different temporal network representations or the concept of network journeys.

2.1.1 Graph theory

Given the enormous size of the graph theory research field, this section in no case claims completeness. However, it is meant as an introductory covering some basic concepts to understand the methods developed in subsequent chapters. It is largely based on the work by Latora et al. (2017), which gives an excellent overview of the network science discipline. In detail, this section covers the graph formalism in Section "Formalism", the area of weighted networks in the subsequent section, random networks, network motifs, ego-networks and node centrality measures in Section "Centrality".

Formalism

According to Latora et al. (2017, p. 3), "[a] graph is defined by giving a set of elements, the graph nodes, and a set of links that join some (or all) pairs of nodes." This definition already indicates that networks are an abstract concept and are not tied to any real-world phenomenon, rather they can be leveraged to model complex systems from a variety of domains.

Starting from the basics, in its simplest form, a network consists of nodes that are linked by undirected edges, which is called an *undirected network*:

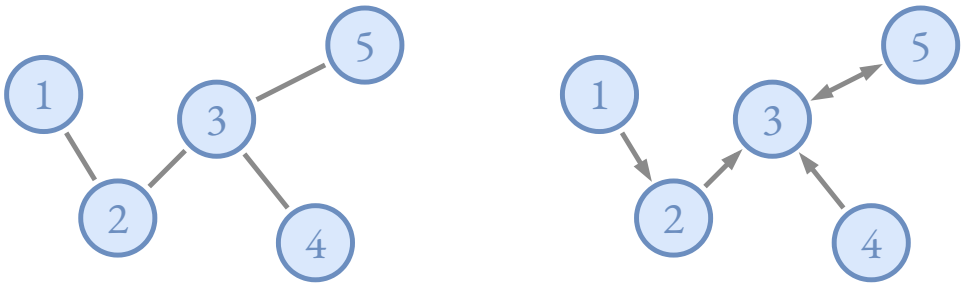
Definition 2.1 (Undirected network)**.** (Latora et al., 2017, Definition 1.1, adjusted). An undirected network $G \equiv (V, L)$ is a tuple of its nodes and edges. Both the nodes V and the edges L are sets of distinct elements. Each edge $l_{ij} = (v_i, v_j)$ (or $l_{ji} = (v_j, v_i)$) with $l_{ij}, l_{ji} \in L$ and $v_i, v_j \in V$ is an unordered pair of the adjacent nodes. In total, $n_V = |V| \neq 0$ nodes and $n_L = |L|$ edges are part of the network.

In some use cases, one might be interested in only a subset of a graph's nodes and/or edges. This subset defines a specific part of the network, also referred to as *subgraph*:

Definition 2.2 (Subgraph)**.** (Latora et al., 2017, Definition 1.4, adjusted). A subgraph $G' = (V', L')$ of a graph $G = (V, L)$ is a part of the whole network defined by the two subsets $V' \subseteq V$ and $L' \subseteq L$. If the subgraph G' consists of all links from the original graph G among the set of nodes V', one calls it an *induced* subgraph.

Not all phenomena are well-modeled by an undirected network, but occurring relationships might imply some directionality instead. In these cases, *directed* networks are better suited as graph representations. In contrast to the undirected networks for the directed networks, the inequality $l_{ij} \neq l_{ji}$ holds. Figure 2.1 gives an example of an undirected and a directed network with five nodes

each. In the case of the directed network, one can see that both directions are present only for the link between node 3 and node 5. All the other links are uni-directional.



(a) Undirected (b) Directed

Figure 2.1: Two exemplary networks illustrating the difference between directed and undirected networks

Formally, the same network can be represented in multiple ways. Next to a visual representation as in Figure 2.1, networks are commonly described by their *adjacency matrix*.

Definition 2.3 (Adjacency matrix)**.** (Latora et al., 2017, Definition 1.15, adjusted). A graph $G = (V, L)$ is represented by a binary $n_V \times n_V$ square matrix, termed adjacency matrix $\mathbb{A}$. Its entries $\mathbb{a}_{ij}$ are defined by the following condition:

$$\mathbb{a}_{ij} = \begin{cases} 1 & \text{iff } (v_i, v_j) \in L, \\ 0 & \text{otherwise.} \end{cases}$$

In Equation 2.1, the two adjacency matrices belonging to the networks of Figure 2.1 are shown. Thereby, $\mathbb{A}_u$ is the according matrix of the undirected network of Figure 2.1a and $\mathbb{A}_d$ is the corresponding matrix of the network shown in Figure 2.1b. As one can see, matrix $\mathbb{A}_u$ is symmetric and includes $2n_L$ non-zero values. In contrast, the matrix $\mathbb{A}_d$ has n_L non-zero values and is not symmetric. This asymmetry is caused by the uni-directional edges present in the corresponding network. For an uni-directional edge between the nodes v_i and v_j, only the value $\mathbb{a}_{ij}$ or the entry $\mathbb{a}_{ji}$ is non-zero.

$$\mathbb{A}_u = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \qquad \mathbb{A}_d = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}. \tag{2.1}$$

Notably, the definition of an adjacency matrix given above and the discussed examples do not contain self-loops. These would be represented by non-zero values on the diagonal of the adjacency matrix: $\mathbb{a}_{ii} = 1$.

Further, an adjacency matrix is not a graph's only formal representation form. In cases where the network is sparse ($n_L \ll n_V^2$), most entries in the corresponding adjacency matrix are zero. A more compact representation form called *edge list* is commonly used in such scenarios. In Equation 2.2, the edge lists belonging to the networks of Figure 2.1 ($\mathbb{L}_u$ belongs to Figure 2.1a) are shown.

Definition 2.4 (Edge list)**.** (Latora et al., 2017, Definition 1.17, adjusted). Two vectors **i** and **j** make up the edge list representation of a graph. These vectors store integers that indicate the row and column indices of the graph's adjacency matrix non-zero entries. Both vectors contain $2n_L$ elements in case of an undirected graph and n_L elements otherwise.

$$\mathbb{L}_u = (\mathbf{i}, \mathbf{j}),\ \mathbf{i} = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 3 \\ 3 \\ 3 \\ 4 \\ 5 \end{pmatrix},\ \mathbf{j} = \begin{pmatrix} 2 \\ 1 \\ 3 \\ 2 \\ 4 \\ 5 \\ 3 \\ 3 \end{pmatrix}, \qquad \mathbb{L}_d = (\mathbf{i}, \mathbf{j}),\ \mathbf{i} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix},\ \mathbf{j} = \begin{pmatrix} 2 \\ 3 \\ 5 \\ 3 \\ 3 \end{pmatrix}. \tag{2.2}$$

Furthermore, the nodes in a network might have different topological characteristics. One of the metrics to quantify the "connectedness" of a node is the so-called *degree*:

Definition 2.5 (Degree)**.** (Latora et al., 2017, Definition 1.7, adjusted). Given a node v_i, its degree k_i is defined as the number of incident links. For directed networks, one differentiates between the number of ingoing links that make up the in-degree k_i^{in} of a node v_i and the number of outgoing links that define the out-degree k_i^{out} respectively. In total, the degree of a node v_i in a directed network is given as $k_i = k_i^{in} + k_i^{out}$.

Additionally, on the network level, instead of a single node level, the average degree $\langle k \rangle$ characterizes the overall connectedness of the nodes in the network. It is defined as

$$\langle k \rangle = \frac{1}{n_V} \sum_{i=1}^{n_V} k_i = \frac{2n_L}{n_V}. \tag{2.3}$$

Further, for directed networks the equality $\langle k_{in} \rangle = \langle k_{out} \rangle = \frac{n_L}{n_V}$ holds for the average in- and out-degree.

Similar in a sense to also describe the topology of a network are the concepts of *walks* and *paths*. These concepts characterize traversals of the according graph and are closely related to node reachability.

Definition 2.6 (Walk & path)**.** (Latora et al., 2017, Definition 1.8, notation adjusted). A walk $\mathscr{W}(i, j)$ from node i to node j is an alternating sequence of nodes and edges [...] $\mathscr{W} = (i \equiv v_0, l_1, v_1, l_2, \dots, l_{n_\mathscr{W}}, v_{n_\mathscr{W}} \equiv j)$ that begins with i and ends with j, such that $l_i = (v_{i-1}, v_i)$ for $i = 1, 2, \dots, n_\mathscr{W}$. Usually a walk is indicated by giving only the sequence of traversed nodes: $\mathscr{W} = (i \equiv v_0, v_1, \dots, v_{n_\mathscr{W}} \equiv j)$. The length of the walk, $n_\mathscr{W}$ [...], is defined as the number of edges [...] in the sequence. [...] A path is a walk in which no node is visited more than once. A shortest path (or geodesic) from node i to node j is a walk of minimal length from i to j, and in the following will be denoted as $\bar{S}(i, j)$.

Notably, for an undirected graph, a walk or path that connects two nodes following one edge direction also represents a walk or path respectively in the other direction. To give an example based on the directed network shown in Figure 2.1b, the path $\mathscr{W} = (1, 2, 3, 5)$ links the two nodes 1 and 5.

Weighted networks

Weighted networks are a natural extension of the network types presented in the previous paragraphs. According to Latora et al. (2017, p. 374), a weighted network is "[...] a network in which each link is associated with a numerical value, in general, a positive real number, representing

the strength of the corresponding connection." Formally, a weighted network is defined as follows:

Definition 2.7 (Weighted network)**.** (Latora et al., 2017, Definition 10.1, adjusted). Next to the set of nodes V and edges L, a weighted network $G^W \equiv (V, L, W)$ consists of a set of weights $W = \{w_1, w_2, \dots, w_{n_L}\}$ with $w_i \in \mathbb{R}^+$ that are attributed to the edges in the network.

Furthermore, to represent a weighted network in the form of a matrix, a binary adjacency matrix is insufficient as the edge weights must also be encoded. Therefore, a *weighted adjacency matrix* is leveraged that is defined as follows:

Definition 2.8 (Weighted adjacency matrix)**.** (Latora et al., 2017, Definition 10.2, adjusted). The weighted adjacency matrix, $\mathbb{A}^W$, of a graph, $G = (V, L)$, is a $n_V \times n_V$ matrix. Thereby, the weights of the links are equal to the matrix's values, i.e., the weight of the link between node i and j determines the value a_{ij}^W in the weighted adjacency matrix. Further, $a_{ij}^W = 0$ holds if the two nodes i and j are not connected, as well as $a_{ii}^W = 0 \; \forall i$.

Next to the adjacency matrix of a graph, the weighted links should also be considered for the node degrees. Therefore, a node's *strength* quantifies the node's connectedness in a weighted network and functions as a natural extension of the degree metric in the case of weighted graphs.

Definition 2.9 (Node strength)**.** (Latora et al., 2017, Definition 10.3, notation adjusted). The strength k_i of a node i is the sum of the weights of the edges incident in i:

$$k_i = \sum_{j=1}^{n_V} a_{ij}^W . \tag{2.4}$$

If the graph is directed, the strength of the node has two components: the sum of weights of outgoing links k_i^{out}, referred to as the out-strength of the node, and the sum of weights of ingoing links k_i^{in}, referred to as the in-strength of node i:

$$k_i^{out} = \sum_{j=1}^{n_V} a_{ij}^W \qquad k_i^{in} = \sum_{j=1}^{n_V} a_{ji}^W . \tag{2.5}$$

The total strength of the node is then defined as $k_i = k_i^{out} + k_i^{in}$.

Random networks

Random networks make up a specific class of networks. According to Latora et al. (2017, p. 69), they are characterized by "[...] the disordered nature of the arrangement of links between different

nodes." Even though most real-world networks do not reveal random graph properties (see Latora et al. (2017, p. 252)), they still play a crucial role when it comes to theoretical considerations. In this regard, various random network models exist. These describe not only a single network but instead a complete set of potential random network instances. In the emergence of the field studying random networks, the two researchers Erdős and Rényi (ER) played a significant role (Erdős and Rényi, 1959, 1960). Two of the well-known random network models are named after them:

Definition 2.10 (ER model A: uniform random graphs)**.** (Latora et al., 2017, Definition 3.1, notation adjusted). Let $0 \leq n_L \leq X$, where $X = \frac{n_V(n_V-1)}{2}$. The model, denoted as $G^{ER}_{n_V,n_L}$, consists in the ensemble of graphs with n_V nodes generated by connecting n_L randomly selected pairs of nodes, uniformly among the X possible pairs. Each graph $G = (V, L)$ with $|V| = n_V$ and $n_L = |L|$ is assigned the same probability.

Definition 2.11 (ER model B: binomial random graphs)**.** (Latora et al., 2017, Definition 3.2, notation adjusted). Let $0 \leq \mathrm{p} \leq 1$. The model, denoted as $G^{ER}_{n_V,\mathrm{p}}$, consists in the ensemble of graphs with n_V nodes obtained by connecting each pair of nodes with a probability p. The probability P_G associated with a graph $G = (V, L)$ with $|V| = n_V$ and $|L| = n_L$ is $\mathrm{P}_G = \mathrm{p}^{n_L}(1-\mathrm{p})^{X-n_L}$, where $X = \frac{n_V(n_V-1)}{2}$.

Even though both models can be leveraged to create random networks, the sampling strategy differs. For the ER model A, a fixed number of links is randomly selected to be "materialized" in the generated network. In contrast, for the ER model B, the probability of a potential edge to be "materialized" is fixed. Therefore, for the latter model, multiple network generation processes might lead to networks with varying numbers of edges. Notably, more random network models exist. Nevertheless, covering all of these would go beyond the scope of this thesis. We refer the interested reader to the Chapters 4, 5, and 6 of Latora et al. (2017).

Network motifs

For specific use cases, detecting and analyzing certain topological patterns occurring in a network is highly interesting. Thereby, so-called *network motifs* play a crucial role. Latora et al. (2017, p. 324) define them as "[...] the recurrent structures representing the building blocks of a given network."

Definition 2.12 (Network motifs)**.** (Latora et al., 2017, Definition 8.6, notation adjusted). An undirected (directed) subgraph G' with $n_{V'}$ nodes and $n_{L'}$ links is said to be a network motif if it occurs in an undirected (directed) graph G at a number significantly higher than in randomised ver-

sions of the graph, i.e. in graphs with the same number of nodes and links, and degree distribution as the original one, but where the links are distributed at random.

Given Definition 2.12, one still needs to clarify how the occurrences of a subgraph G' in the graph G are detected. For that, the subgraph itself and its topologically equivalent subgraphs are checked. Then, based on the number of occurrences of all topological variations of G', it is decided whether they appear significantly more frequently compared to similar random networks.

Ego-networks

This paragraph is based on the book of Crossley et al. (2015) that gives a comprehensive overview on the topic of social network analysis based on *ego networks* (a.k.a. "ego-nets"). Notably, the concept of ego-nets is closely tied to social network analysis in which actors and their relationships are analyzed (see Section 2.3). Therefore, ego-net analysis is also referred to as "actor-centered" analysis (Crossley et al., 2015, p. 1). Thereby, an ego-net consists of a central actor node, the ego, and all the other actors the ego is related to. Commonly, the relationships among the non-ego nodes are also considered part of the ego network. Nevertheless, this is not a strict requirement. Generally, the concept of ego-nets is not tied to a particular type of actor or relationship. Instead, arbitrary phenomena, such as the economic exchange between companies or the emotional closeness between people, can be modeled by leveraging ego networks.

Centrality

Some use cases benefit a lot from gaining insights into which nodes in a network are particularly "important" and play a key role. Nevertheless, measuring the importance of a node is a challenging task, and numerous metrics have been developed to "[...] characterize and rank the nodes of a network." (Latora et al., 2017, p. 31) One quite simple measure is the *degree centrality* which is based on the node's degree:

Definition 2.13 (Degree centrality)**.** (Latora et al., 2017, Definition 2.2, notation adjusted). In an undirected graph, the degree centrality of node i ($i = 1, 2, \dots, n_V$) is defined as:

$$c_i^D = k_i = \sum_{j=1}^{n_V} \mathbb{A}_{ij}, \tag{2.6}$$

where k_i is the degree of node i.

In a directed graph, in-degree and out-degree centrality of node i are respectively defined as:

$$c_i^{D^{in}} = k_i^{in} = \sum_{j=1}^{n_V} \mathbb{a}_{ji}, \quad c_i^{D^{out}} = k_i^{out} = \sum_{j=1}^{n_V} \mathbb{a}_{ij}, \tag{2.7}$$

where k_i^{in} and k_i^{out} are respectively in- and out-degrees.

The normalised degree centralities C_i^D, $C_i^{D_{in}}$, $C_i^{D_{out}}$ are obtained by dividing the three quantities above by $n_V - 1$. For instance, in the undirected case we have:

$$C_i^D = \frac{c_i^D}{n_V - 1}. \tag{2.8}$$

Next to the centrality measures based on the node's degree, another set of metrics exists. These focus on determining a node's centrality based on the paths in the network. As such, the *betweenness centrality* takes into account the number of shortest paths the analyzed node is part of.

Definition 2.14 (Betweenness centrality)**.** (Latora et al., 2017, Definition 2.7, notation adjusted). In a connected graph, the shortest-path betweenness centrality of node i is defined as:

$$c_i^B = \sum_{\substack{j=1 \\ j \neq i}}^{n_V} \sum_{\substack{k=1 \\ k \neq i,j}}^{n_V} \frac{n_{jk}(i)}{n_{jk}} \tag{2.9}$$

where n_{jk} is the number of geodesics from node j to node k, whereas $n_{jk}(i)$ is the number of geodesics from node j to k, containing node i. The normalised quantity, defined as:

$$C_i^B = \frac{c_i^B}{(n_V - 1)(n_V - 2)} \tag{2.10}$$

takes values in the range $[0, 1]$.

2.1.2 Communities: detection and evolution

This section builds on Chapter 9 of the book by Latora et al. (2017), which gives a good overview of the *community detection* topic. They understand communities in networks as "[...] clusters of nodes such that nodes within the same cluster are more tightly connected than nodes belonging to two different clusters." Therefore, communities make up "mesoscopic structures" whose scale are between a single node and that of the complete graph. In general, studying the communities of a network allows one to gain insights into the graph's internal structure. Further, for time-

dependent networks (see Section 2.1.5), the temporal *evolution* of contained communities needs to be considered as well.

In particular, this section addresses three topics related to network communities. First, modularity is discussed as a metric to evaluate the partitioning of a graph with respect to community structures. Secondly, Infomap as an algorithm for the detection of communities is examined, and finally, we focus on the evolution of communities in time-varying graphs.

Modularity

Commonly, community detection algorithms aim at finding graph partitions that are likely to reflect the actual community structure of the network. Nevertheless, to quantitatively measure how "good" a given network partition is, a so-called *quality function* is necessary. Ideally, optimizing this quality function should lead to partitioning the network into real communities. *Modularity* is such a quality function.

Definition 2.15 (Modularity)**.** (Latora et al., 2017, Definition 9.11, notation adjusted). Given a graph G and a partition $\ddot{P}_V = \{V_1, V_2, \dots, V_{n_{\ddot{P}}}\}$ of its nodes into $n_{\ddot{P}}$ sets, the modularity $\mathcal{Q}_{\ddot{P}}$ of partition $\ddot{P}_V$ can be written as:

$$\mathcal{Q}_{\ddot{P}} = \sum_{i=1}^{n_{\ddot{P}}} \left[\frac{k_{ii}}{n_L} - \left(\frac{k_i}{2n_L} \right)^2 \right] \tag{2.11}$$

where k_i is the degree of set V_i, defined as the sum of the degrees of all the nodes in set V_i, k_{ii} is the total number of links joining nodes in set V_i, and $2n_L$ is the total number of links in G.

In other words, the modularity metric measures if the number of internal edges of a node cluster is larger than expected in a random network with the same degree for each partition set as in the analyzed network. The larger the modularity value, the more the network reveals clustered node structures. It is defined to be smaller than one and can also take on negative values.

Infomap

Infomap is an algorithm to be used for network clustering (Edler et al., 2023) and is based on the "map equation" (Rosvall et al., 2009). Thereby, the map equation allows to determine the theoretical limit of how efficiently the movement of a random walker on the network can be encoded. Efficiently encoding the trajectories can then be used to partition the network: "If we can find an optimal code for describing places traced by a path on a network, we have also solved the dual problem of finding the important structural features of that network." (Rosvall et al., 2009) Naively, the random walks could simply be encoded as a sequence of unique node

identifiers. Nevertheless, this encoding approach might not be the most efficient one as the node IDs would need many bits. Alternatively, the sequence can be encoded by introducing codes of specific modules/clusters/communities in the network. In the path sequence, one then needs to record which modules are visited by the random walker and when it leaves a module again. Even though this additional information leads to some encoding overhead, it also allows a more concise encoding of the node identifiers, given that these can be reused across the different modules and, therefore, need fewer bits. Finally, testing multiple ways to cluster the network and using the map equation to derive the theoretical description length limits leads to the optimal partitioning of the network.

In contrast to other community detection approaches that leverage modularity maximization to find the optimal partitioning of a network (e.g., Newman and Girvan (2004)), the Infomap approach "maps the flow" on the network which is restricted by the underlying network structure. Therefore, the focus is placed "[...] on the interdependence of links and the dynamics on the network once it has been formed [...]" (Rosvall et al., 2009).

Community evolution

Communities of nodes might not be static if the underlying network is evolving. To conduct community analyses in such a temporal network context, it is required to leverage specialized algorithms that can handle *community evolution*. In the past, such approaches have been applied to investigate the evolution of topics, e.g., Lorenz-Spreen et al. (2018), or to examine trends, e.g., Cazabet et al. (2012). We refer the interested reader to the survey of Rossetti and Cazabet (2018) for a comprehensive overview of the research field dealing with community detection in temporally evolving networks.

2.1.3 Heterogeneous information networks

The following section introduces heterogeneous information networks (HINs) as a special category of networks. For that, their formalism is introduced at first in Section "Formalism". This introduction not only includes the fundamental concepts covering the network-related definitions but also derived concepts such as meta paths. Finally, in Section "Comparison to related concepts", HINs are compared to related concepts such as complex networks and knowledge graphs.

Formalism

The concept of HINs is not new but has already been introduced a few years ago (see Sun et al. (2009)). Essentially, a HIN is a type of network that consists of *multiple* types of objects and/or relationships. Still, its fundamental building blocks are nodes V and links L. To formally define HINs, one has to start clarifying the concept of *information* networks:

Definition 2.16 (Information network)**.** (Shi et al., 2017, Definition 1, notation adjusted). An information network is defined as a directed graph $G = (V, L)$ with an object type mapping function $\varphi : V \rightarrow A$ and a link type mapping function $\psi : L \rightarrow R$. Each object $v \in V$ belongs to one particular object type in the object type set A: $\varphi(v) \in A$, and each link $l \in L$ belongs to a particular relation type in the relation type set R: $\psi(l) \in R$. If two links belong to the same relation type, the two links share the same starting object type as well as the ending object type.

HINs are different from *homogeneous* networks as they contain more than one type of object and/or relationship. For a more detailed comparison to related concepts, see the following section. Based on Definition 2.16 the differentiation can be formalized as follows:

Definition 2.17 (Heterogeneous/Homogeneous information network)**.** (Shi et al., 2017, Definition 2, notation adjusted). The information network is called *heterogeneous information network* if the types of objects $|A| > 1$ or the types of relations $|R| > 1$; otherwise, it is a *homogeneous information network*.

To model the overall structure of a HIN, meaning how objects of a specific type are related to other objects by which relationship type, one has to resort to a more abstract level. This abstraction requirement is why so-called network schemas have been introduced:

Definition 2.18 (Network schema)**.** (Shi et al., 2017, Definition 3, notation adjusted). The network schema, denoted as $S_G = (A, R)$, is a meta template for an information network $G = (V, L)$ with the object type mapping $\varphi : V \rightarrow A$ and the link type mapping $\psi : L \rightarrow R$, which is a directed graph defined over object types A, with edges as relations from R.

Therefore, HINs that follow the structure of a given network schema are called *network instances* of the same network schema. As part of that, the schematic dyad of two object types a_1 and a_2 that are related by the link type r is written as $a_1 \xrightarrow{r} a_2$. Further, given that the network is constrained by its schema, one can classify network instances as semi-structured networks (Shi et al., 2017).

Figure 2.2 shows an exemplary network schema based on Twitter data that is modeled as HIN. Both the nodes, indicated as circles, and the relationships are labeled. These labels indicate the respective

types (A and R) as used in the HIN. The "Reply" and "Retweet" self-loops indicate relationships among "Tweet" nodes. In contrast, the other two relationship types link nodes of different types, e.g., the "Mention" relationship links "Username" nodes and "Tweet" nodes.

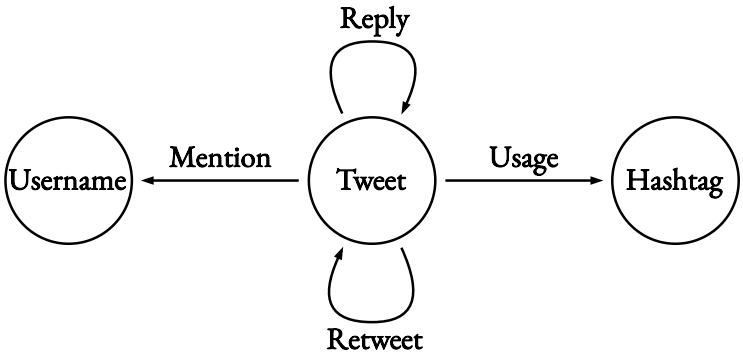


Figure 2.2: Exemplary network schema based on a HIN extracted from Twitter data

In HINs, specific paths that traverse a semantically meaningful sequence of nodes are called *meta paths*, which are defined as follows:

Definition 2.19 (Meta path)**.** (Sun et al., 2011, Definition 3, notation adjusted). A meta path P is a path defined on the graph of network schema $S_G = (A, R)$, and is denoted in the form of $a_1 \xrightarrow{r_1} a_2 \xrightarrow{r_2} \dots \xrightarrow{\eta} a_{l+1}$, which defines a composite relation $r_P = r_1 \circ r_2 \circ \dots \circ \eta$ between type a_1 and a_{l+1}, where $\circ$ denotes the composition operator on relations.

Furthermore, according to Sun et al. (2011), if only one relationship type exists between the same pair of objects, denoting the sequence of objects is sufficient to unambiguously define a meta path, $P = (a_1, a_2, \dots, a_{l+1})$, and can be used as simplified notation. Also, the length of P is defined by the number of links in P. A concrete realization of a meta path as occurring in a network instance is called a *path instance* p. Given such a path instance $p = (v_1, v_2, \dots, v_{l+1})$ in the network G, which follows the schema S_G, the following holds true: $\forall i\ \varphi(v_i) = a_i$ and $\psi(l_i) = r_i$ with $l_i = (v_i, v_{i+1})$.

Remark. A path as defined by Definition 2.6 requires that none of the nodes is passed more than once. However, no such restriction exists for the meta path Definition 2.19 and the related definition of a path instance. Following a specific meta path instance, the same node might be visited multiple times. For example, take the following meta path, which links the hashtags used in the original tweet with these used in the retweets: $hashtag \xrightarrow{used\ in} tweet \xrightarrow{retweet\ of} tweet \xrightarrow{uses} hashtag$. Now, if the start and end hashtag is checked to be the same, one obtains all the hashtags

that are used in both the original and the retweeted post. This request is still valid under the less restricted meta path definition.

Described meta paths can not only be used to define latent relationships but also to project HINs. In the projected network, nodes are directly linked that were only latently connected by a specific meta-path in the original network. Milani Fard et al. (2019) propose a network projection method based on meta-paths. They call the projected network an "augmented reduced graph". It is formally defined in the following:

Definition 2.20 (Augmented reduced graph)**.** (Milani Fard et al., 2019, Definition 5, notation adjusted). Given a HIN graph $G = (V, L)$ and a target meta path $P(a_i, a_j)$ between nodes of type a_i and a_j, an augmented reduced graph $G^P = (V^P, L^P)$ is a graph, where $V^P \subseteq V$ and nodes in V^P are of type a_i and a_j, and edges in L^P indicate relationships of type P in G.

Comparison to related concepts

For a broader picture, the concept of HINs needs to be compared and differentiated from related network concepts. Such comparison is, to some extent, already outlined in previous work by Shi et al. (2017) and Shi and Yu (2017, p. 5).

Homogeneous network: As already formalized in Definition 2.17, homogeneous networks are different from HINs as they only contain a single type of object and relationship, which makes them a special case of heterogeneous networks. Homogeneous networks can be extracted from HINs by network projections. Given a HIN $G = (V, L)$ with network schema S_G such a network projection $G_r^a = (V', L')$ by the relation $a \xrightarrow{r} a$ is defined as: $V' = \{v \in V | \varphi(v) = a\}$ and $L' = \{l \in L | \psi(l) = r\}$.

Multi-relational network: Multi-relational networks as used by Yang et al. (2012) are again a special case of heterogeneous networks as they are restricted to networks with multiple link types but only a single object type.

Social network: According to Myers et al. (2014), information and social networks have different characteristics, which help to classify a given network instance as information or a social network. A list of network characteristics for information and social networks is shown in Table 2.1. For example, information networks are typified by a lack of reciprocity and large two-hop neighborhoods. In contrast, social networks generally have a high degree of reciprocity and small shortest path lengths.

Table 2.1: Characteristics of information and social networks according to Myers et al. (2014)

Information network	**Social network**
lack of reciprocity	high degree of reciprocity
large two-hop neighborhoods	small shortest path lengths
large vertex degrees	high degree assortativity
	large connected components
	high clustering coefficients

In addition to the characteristics shown in Table 2.1, one can say that the dominant interaction within information networks is the spreading of information, while social networks are mainly built on social ties (Myers et al., 2014).

Apart from the characteristics of the interaction(s) within the network, social network data, generally, can be modeled in several ways, e.g., by using HINs. In this case, nodes might be social actors and edges the relationships among those. Multiple types of relationships and/or actors might be modeled by using different node/edge types accordingly.

Complex network: According to Kim and Wilhelm (2008) *complex* in contrast to *random* networks are characterized by specific topological features. These features include a high clustering coefficient combined with a small characteristic path length (Watts and Strogatz, 1998), a power-law distribution of node degrees (Barabási and Albert, 1999), a (inverse) correlation of node degrees (Newman, 2002), a, compared to random networks, significantly higher presence of network motifs (Milo et al., 2002) and a division into subgraph modules (Newman, 2006). In contrast to complex networks, HINs are not defined by their topological features and, therefore, the study of complex networks also places a different focus on investigating named network structures (Shi and Yu, 2017, p. 5) as opposed to extracting semantically meaningful information. Many real-world networks from the field of biology, social sciences, or economics can be classified as complex networks (Kim and Wilhelm, 2008). Given that many of them are also semantically heterogeneous, one can assume that many real-world HINs are also complex in nature (Shi and Yu, 2017, p. 5).

Network of networks: Opposed to the model of HINs, the concept of "network of networks" places its focus on the interdependence or rather coupling of *multiple* networks (D'Agostino and Scala, 2014, pp. 3–7). Still, those individual networks and their dependence could also be modeled as a single, large HIN, which contains all the different object and relationship types. In

Section 2.1.4, the model of a network of networks is discussed in more detail in the context of multilayer networks.

Knowledge Graph: In the field of knowledge graph research, the concept of a knowledge graph is used inconsistently, which is why Ehrlinger and Wöß (2016) propose a unified definition: "A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge." Given their definition, it becomes clear that comparing the concept of HINs and that of a knowledge graph is not meaningful. Compared to HINs, knowledge graphs are much more than just a knowledge base and describe the complete system, which implies integrating information from different sources and reasoning to generate additional knowledge (Ehrlinger and Wöß, 2016).

Better suited would be the comparison to ontologies, which are defined as follows: "An ontology is a formal, explicit specification of a shared conceptualization that is characterized by high semantic expressiveness required for increased complexity." (Feilmayr and Wöß, 2016) In the same line, HINs can also be used as models with high semantic expressiveness. Also, given that ontologies do not only refer to the pure schema but to populated instances of the knowledge base as well (Ehrlinger and Wöß, 2016), the meaning is similar to network instances of a given HIN schema. Nevertheless, ontologies tend to be very schema-rich and are meant for complex modeling scenarios, which is less true for HINs, which typically have less complex network schemas (Shi and Yu, 2017, p. 8).

Labeled property graph: The terminology of labeled property graphs (LPGs), commonly referred to as *property graphs*, is described by Rodriguez and Neubauer (2010) and more formally, they are defined by Angles (2018). In general, a property graph can be characterized as "[...] directed, labeled, attributed, multi-graph [...]" (Rodriguez and Neubauer, 2010), which means that its relationships are directed, its nodes and edges are typed/labeled and come with additional attributes, as well as multiple edges might connect the same pair of nodes. Thereby, the additional node and edge attributes, referred to as *properties*, allow the storage of specific features, such as the first name belonging to a "person" node or the date belonging to a "published in" relationship (Angles, 2018). Given its large degree of flexibility, various Graph Database Management Systems (GDBMSs) leverage property graphs as database model (Angles, 2018). Compared to the formal definition of HINs, they also come with typed nodes and edges, as well as explicitly model node and edge attributes. Still, they miss derived features such as meta paths even though these could also be formalized via typed paths, as leveraged by the pattern matching syntax, i.e., the `MATCH` clause, used to query property graphs (see Angles (2018)).

2.1.4 Multilayer networks

The following section about *multilayer networks* is based on the book by Bianconi (2018, pp. 79–114) with the respective notation being adjusted. Generally, multilayer networks describe a composition of multiple interacting networks. Thereby, a single layer in a multilayer network represents one of the composing networks. Modeling a composition of networks as a multilayer network facilitates understanding the interplay between the different network layers.

Multilayer networks are often referred to as *multidimensional networks*. In this case, dimensions correspond to layers with respect to the multilayer network terminology. Nevertheless, given that the term "dimension" has a special meaning in the field of mathematics and physics, one should preferably refer to multidimensional networks as multilayer networks (Kivelä et al., 2014).

In the following, the mathematical formalism and the most general form of multilayer networks are described in Section "Formalism". Further, special subtypes of multilayer networks, namely multiplex networks, multi-slice networks and networks of networks, are explained. Also, examples of multilayer networks are given, and in the final paragraph, multilayer networks are compared to the concept of HINs.

Formalism

Each layer of the multilayer network represents a network of a certain type. Interactions among nodes of the same layer are referred to as *intralinks*, whereas interactions between nodes of different layers are called *interlinks*. Given that a multilayer network consists of $n_{\mathscr{M}}$ layers, the same amount of networks exists that describe the interactions among nodes in the same layer. Furthermore, for every pair of layers in the multilayer network, another network represents the interactions among nodes of the paired network layers. Therefore, in addition to the $n_{\mathscr{M}}$ networks covering the intralinks, $\frac{n_{\mathscr{M}}(n_{\mathscr{M}}-1)}{2}$ networks of the interlinks are part of a $n_{\mathscr{M}}$-layer network. According to Bianconi (2018, p. 100) the most general description of a multilayer network M can be given as follows:

$$M = (\mathscr{M}, \dot{\mathscr{G}}, \ddot{\mathscr{G}}). \tag{2.12}$$

As Equation 2.12 shows, a multilayer network is defined as a triple consisting of the network layers denoted as $\mathscr{M}$,

$$\mathscr{M} = \{m | m \in \{1, 2, \dots, n_{\mathscr{M}}\}\} \quad \text{with} \quad |\mathscr{M}| = n_{\mathscr{M}}, \tag{2.13}$$

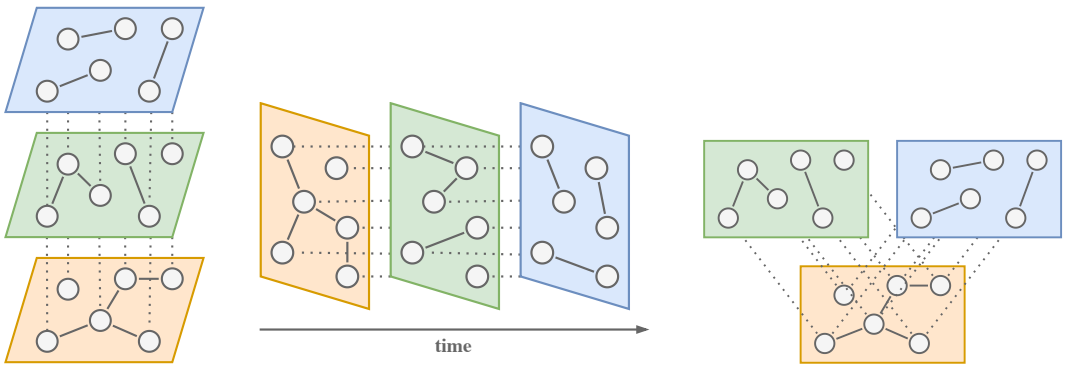
as well as the ordered list of networks that are made up of the interactions within each of the network layers

$$\dot{\mathcal{G}} = (G_1, G_2, \dots, G_m, \dots, G_{n_{\mathcal{M}}}) \quad \text{with} \quad G_m = (V_m, L_m). \tag{2.14}$$

As described above, the nodes V_m of the layer m are thereby connected by intralinks L_m. Further, a multilayer network also comprises networks of interlinks that connect nodes of different layers. Formally, those interlink networks are a $n_{\mathcal{M}} \times n_{\mathcal{M}}$ list $\ddot{\mathcal{G}}$ of bipartite networks $\ddot{\mathcal{G}}_{m,m'}$ whereby the indices of those networks indicate the pair of layers that are connected by the interlinks:

$$\ddot{\mathcal{G}}_{m,m'} = (V_m, V_{m'}, L_{m,m'}) \quad \text{for each } m < m' \text{ and } m, m' \in \{1, 2, \dots, n_{\mathcal{M}}\}. \tag{2.15}$$

In their most general form, nodes of different layers are of different types and do not have any one-to-one mapping regarding nodes of other layers. Sill, various categories of multilayer networks that come with additional characteristics and constraints, such as a one-to-one mapping between nodes of different layers, exist (see Figure 2.3).



(a) Multiplex network (b) Multi-slice network (c) Network of networks

Figure 2.3: Illustrations of different kinds of multilayer networks according to Bianconi (2018, Figure 4.1)

Multiplex networks

According to Bianconi (2018, pp. 102–106), *multiplex* networks are defined as outlined in the following. In short, they are characterized by two key properties:

- There exists a one-to-one mapping of nodes of different layers. Those corresponding nodes are called *replica nodes*.

- Interlinks can only connect replica nodes.

In their simplest form, multiplex networks do not contain any interlinks. Instead, the same set of nodes is simply replicated across different layers, each representing a different kind of interaction. In this case, multiplex networks are similar to multi-relational networks, with each type of relationship represented in a different layer of the multiplex network. In contrast, a multiplex network *with* interlinks can be seen in Figure 2.3a. As one can observe, links connecting different network layers also exist in these multiplex networks. Regardless of whether interlinks are part of the multiplex network or not, its set of vertices V can be defined as

$$V = \{i | i \in \{1, 2, \dots, n_V\}\}. \tag{2.16}$$

Based on that, for each of the $n_{\mathcal{M}}$ layers of the network a set of replica nodes V_m can be defined:

$$V_m = \{(i, m) | i \in \{1, 2, \dots, n_V\}\}. \tag{2.17}$$

Note that the nodes with the same label i but different layer indices correspond to each other as replica nodes. Based on that, the set of interlinks $L_{m,m'}$ which connects replica nodes, i.e., nodes with the same label but of different layers (m and m'), is defined as follows:

$$L_{m,m'} = \{[(i, m), (i, m')] | i \in \{1, 2, \dots, n_V\}\}. \tag{2.18}$$

Concerning Equation 2.18, one can see that only links between nodes with the same label i are allowed.

Multi-slice networks

The notion of *multi-slice* networks is also outlined in the book of Bianconi (2018, pp. 106–110). Briefly described, the following features characterize multi-slice networks:

- In multi-slice temporal networks, a one-to-one mapping relates nodes of different layers as replica nodes. Each layer in the network represents a different temporal snapshot.
- Interlinks between nodes of different snapshots only exist in temporal succession.

Accordingly, multi-slice networks can also be seen as temporal snapshot-based networks. In more detail, given a multi-slice network M covering the interactions within a time period T and the time interval Δt chosen as snapshot size, there exist $n_M = \frac{T}{\Delta t}$ layers. Thereby, the layer m captures the interactions that occur in the time window $[(m-1)\Delta t, m\Delta t)$. Further, given that interlinks only occur between replica nodes of subsequent snapshots, the set of interlinks $L_{m,m'}$ is only a non-empty set in case of $m' = m + 1$:

$$L_{m,m'} = \begin{cases} \{[(i,m),(i,m')] | i \in \{1,2,\dots,n_V\}\} & m' = m+1, \\ \emptyset & \text{otherwise.} \end{cases} \tag{2.19}$$

In Figure 2.3b, one can see a multi-slice network consisting of three layers, i.e., temporal snapshots. Only links between subsequent temporal layers exist.

Furthermore, an *aggregated* network $\tilde{G}$ can be constructed from a multi-slice network. For such an aggregation, the temporality of the edges contained in the network is neglected, and all edges, no matter from which network slice, are aggregated into a single network (layer).

Network of networks

Network of networks as discussed in Section 2.1.3 can also be seen as a special kind of multilayer network. With respect to the multilayer network model, each layer represents a network of the "network of networks" model. Interactions between multiple networks are determined by a so-called *supernetwork*, which specifies the layers, i.e., networks, that interlinks can connect. As a visualized example of such a network consisting of three layers, i.e., individual networks, see Figure 2.3c.

Networks of networks are formally defined by Bianconi (2018, pp. 112–114). Accordingly, the supernetwork $G^s = (V^s, L^s)$ is composed of the supernetwork nodes V^s that represent layers of the original multilayer network,

$$V^s = \{m | m \in \{1,2,\dots,n_M\}\}, \tag{2.20}$$

and the set of supernetwork edges L^s. These supernetwork edges determine which pairs of layers in the original network are connected by interlinks. Interlinks in the multilayer network can only exist between nodes of layers also connected in the supernetwork.

Multilayer network examples

Different complex systems can be modeled using the multilayer network framework. Bianconi (2018, Chapter 4) provides examples of versatile application scenarios ranging from social network analysis to the analysis of brain networks. Depending on the specific use case and analysis perspective, different types of multilayer networks are more appropriate to use. Table 2.2 shows a list of exemplary multilayer networks leveraged for various application scenarios. The respective type of the multilayer network is shown as well.

Table 2.2: Exemplary use cases for multilayer networks as outline by Bianconi (2018, Chapter 4)

Description	**Multilayer network type**
social network with multiple interaction types	multiplex
transportation network with multiple connection types	multiplex
scientific interaction network	multiplex
temporal brain correlation network	multi-slice
face-to-face interaction network	multi-slice

Also, in this thesis, concepts from the multilayer network model are leveraged. In particular, temporal network snapshots are applied, aligning with the multi-slice network approach. A detailed comparison with regards to the data analysis model developed in this thesis can be found in Section 3.3.3. The respective temporal network model is employed for different use cases, such as to analyze long-term trends (see Section 4.1.4).

Multilayer network vs. heterogeneous information network

According to Kivelä et al. (2014) multilayer networks can also be seen as HINs with the different node and edge types being mapped to different layers of the multilayer network. Still, it is essential to note that multilayer networks should not be understood as single large networks containing all the interaction types, but rather, the focus is placed on treating the different interaction types separately (Bianconi, 2018, p. 4). This separation contrasts the model of HINs in which all interactions are modeled within a single network. In the HIN model, information regarding the different nature of the nodes and their relationships is encoded in the network's link and edge type mappings as opposed to the different layers of a multilayer network.

2.1.5 Network dynamics

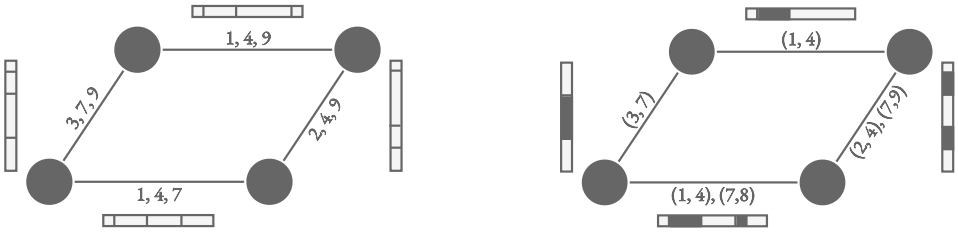
Commonly, in a graph-based data analysis setting, the topology of the underlying network is time-dependent. To consider this time dependency, e.g., for evolving social or time-dependent biological networks, the leveraged network model needs to be adjusted. This adaptation can be done by leveraging a time-dependent network model that considers the *network's dynamics* accordingly. Nevertheless, temporal networks are appropriate only for some dynamic data analysis use cases. According to Holme and Saramäki (2012), only if the temporal evolution of the network topology influences the dynamics of the studied phenomenon it makes sense to resort to a temporal network model. For a counter-example, studying the movement of data packets on the internet does not require a temporal network model, given that the packets traverse the network much faster compared to how the network itself is evolving. It is, therefore, sufficient to treat the network as being static.

In the past, temporal graphs have been referred to in multiple ways, such as time-varying graphs, temporal networks, time-dependent graphs or evolving graphs (Rost et al., 2022). Multiple surveys give a comprehensive overview of the temporal networks topic, e.g., Kostakos (2009), Casteigts et al. (2012), Holme and Saramäki (2012), or Wang et al. (2019). In the following, different aspects of the field are highlighted. A particular focus is placed on concepts again leveraged in other parts of this thesis. In detail, we cover different forms of representing temporal networks, the differentiation between "temporal" and "evolving" metrics, as well as the concept of temporal network journeys.

Temporal network representations

In the past, time-dependent networks have been represented in various ways (see Holme and Saramäki (2012); Wang et al. (2019); Xu (2021)). These representations include *contact sequences*, *interval graphs*, *link streams* and *snapshot networks*. Each of them emphasizes different aspects of the network's temporal evolution and is more or less suited for different analysis use cases. Figure 2.4 illustrates two of these temporal network representations, contact sequences and interval graphs. For both representation forms, the temporal information is modeled as edge property. Additionally, Figure 2.5 visualizes the other representation forms, snapshot networks and link streams, highlighting the difference between continuous and discrete temporal network representations.

Contact sequence: According to Holme and Saramäki (2012), modeling the dynamics of a network in the form of a contact sequence is appropriate if the durations of the interactions are negligible. In these cases, representing the edges as a triple of the source node, the target node, and



(a) Contact sequence (b) Interval graph

Figure 2.4: Two representations of time-dependent networks for which the temporal information is modeled as edge property according to Holme and Saramäki (2012, Figure 4)

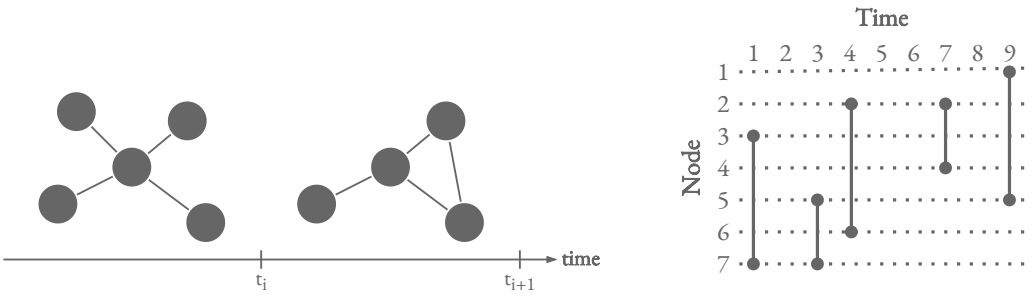
a simple timestamp is sufficient. Exemplary temporal networks that can be modeled as contact sequences are e-mail networks or networks based on physical proximity data. Also, see Figure 2.4a for a visualization of a contact sequence. The timestamps of the contacts are indicated as black lines on the grey timelines. Representing a temporal network as a contact sequence allows to capture its dynamics in a continuous manner.

Interval graph: As outline by Holme and Saramäki (2012), in contrast to contact sequences, interval graphs are the appropriate modeling approach if the durations of the interactions are *not* negligible. For interval graphs, time intervals instead of simple timestamps are assigned to the network edges to indicate in which time windows the respective interaction is active. Figure 2.4b shows an exemplary interval graph. The active periods of the edges are marked in black on the grey timelines. Real-world applications of interval graphs are seasonal food webs or infrastructure networks. Like contact sequences, an interval graph is also a continuous representation form.

Link stream: A link stream represents the time points or intervals of interaction between a fixed set of nodes (Xu, 2021; Latapy et al., 2018). In a way, they are similar to contact sequences and interval graphs, which also keep track of the timestamps, respectively time periods, a specific edge is active. Nevertheless, for link streams, the activity of edges is explicitly represented with regards to a time axis (see Figure 2.5b) instead of being indicated by edge properties (see Figure 2.4). Further, Latapy et al. (2018) extend link streams to the concept of *stream graphs*. For these, not only are the edges dynamic, i.e., only active at certain times, but also the nodes are temporally dependent. In any case, these time-dependent network representation forms can be leveraged for studying the network's continuous dynamics.

Snapshot networks: For some network analysis use cases, discretizing the time-dependent network might facilitate unveiling the network's dynamics. Such network discretization can be achieved by modeling the network as a sequence of static network snapshots (Wang et al., 2019; Xu, 2021). Each of the temporal snapshot networks then contains only those edges that are ac-

tive during the time (period) covered by the respective snapshot. In contrast to the previously described continuous temporal graph representations, discrete network snapshots are unsuitable for continuously analyzing graph properties (Xu, 2021). Also, the network's evolutionary changes should not be obliterated by the selected network sampling strategy.



(a) Network snapshots (b) Link stream

Figure 2.5: Two representations of time-dependent networks: network snapshots (discrete) and a link stream (continuous) (according to Xu (2021, Figure 9))

Temporal vs. evolving

Even though the terms "temporal" and "evolving" might commonly be used interchangeably, it has to be noted that in the context of studying network dynamics, they might come with different meanings. In this regard, Rost et al. (2023) who propose novel degree metrics in their work understand a *temporal* metric with respect to a certain *point in time*. In contrast, an *evolutionary* metric comes in the form of a time series describing the metric over a given *period of time*. Therefore, understanding "temporal" as referring to a specific moment and "evolving" as referring to a time window is required.

Journey

Not all concepts that apply to static networks can be transferred to the context of time-varying networks without further adjustment. As such, the concept of walks/paths must be re-defined for time-dependent networks. In this regard, *journeys* are defined as the temporal-sensitive equivalent (Xuan et al., 2003; Santoro et al., 2011). Even though a journey is also realized as a sequence of edges, given that these are time-dependent, additional conditions must be fulfilled. These conditions include the necessity for each edge part of the journey to be active when it is traversed and that each edge needs to follow in temporal succession of the previous edge (Santoro et al., 2011). As a result, "[...] journeys cannot go to the past." (Xuan et al., 2003)

2.2 Digital Media

As already outlined in Section 1.2, media plays a central role in today's world. So do the digital platforms on which users consume respective media. In the first step, understanding how these platforms work is essential because, in the next step, during the analysis of the collected media data, this knowledge is indispensable. Therefore, the following Section 2.2.1 introduces media platforms using the example of Twitter. Secondly, in Section 2.2.2, special attention is drawn to data quality, again in the case of Twitter social media data. Issues related to data quality are discussed because these should be considered during the subsequent media analysis studies.

2.2.1 Platforms

Various digital media platforms, ranging from online services of news outlets, such as CNN[1] or BBC[2], to social media services like Facebook[3] or Instagram[4] exist. From an academic perspective, the emergence of these platforms opens up new research opportunities. Questions about how users behave on these platforms, which content is gaining popularity or how it is spread are still not fully answered. Given that this thesis also deals with the objective of media analytics, it might be helpful to give an introduction to named media platforms. In the following, this is done based on the Twitter platform because Twitter data is analyzed in numerous ways in the subsequent chapters.

Twitter

Twitter can be described as an information sharing platform with additional social interaction features. Zubiaga et al. (2015) give a good summary of the Twitter platform. This section largely builts on their work.

Remark. As already explained in Section 1.2, to be consistent with past work, we still refer to the renamed platform by its old name "Twitter" even though it has been renamed to "X" (Dey and Conlin, 2023).

This section is separated into multiple paragraphs: First of all, various interaction features available to the users on the Twitter platform are described. Subsequently, because trends are especially relevant for the next parts of this thesis, e.g., Chapter 4, Twitter trends are also covered. Finally, Twitter data accessing capabilities are discussed.

[1] Breaking News, Latest News and Videos | CNN: https://edition.cnn.com (accessed 2023-10-27)
[2] BBC – Homepage: https://www.bbc.com (accessed 2023-10-27)
[3] Facebook – log in or sign up: https://www.facebook.com (accessed 2023-10-27)
[4] Instagram: https://www.instagram.com (accessed 2023-10-27)

Interaction features: On the Twitter platform users can interact in various ways (see Zubiaga et al. (2015)). These interaction features and the way they are used on the platform are shown in Table 2.3. Some of these features can also be combined in a single tweet, e.g., hashtags can also be used in a reply message. Also, the content users share on the platform can be of various types, like URLs, images, videos, or simple text messages. As of January 2023, the length of these text messages is constrained by a 280 character limit (Twitter, Inc., 2023a).

Table 2.3: Interaction features of the Twitter service along with a description of the corresponding syntax (Zubiaga et al., 2015)

Feature	Description	Example
user mention	In a tweet another Twitter account can be mentioned or referred to by placing its name after an @-sign.	Please check @BBC for the latest news.
reply	If a tweet is meant as direct reply to a previous tweet, the @-mentioning of this username needs to be placed at the beginning of the reply tweet.	@BarackObama That's great news.
retweet	User "A" can share a tweet of another user "B" on his own profile by retweeting it. This is indicated by placing "RT @UserB:" in front of the original tweet. Retweets can also be nested.	RT @POTUS: This is a public health risk.
hashtag	Certain terms in a tweet can be marked as keywords by prefixing a #-sign to them. These hashtags work as tagging system on the Twitter platform and link tweets to a certain topic.	Great #soccer game!

From an abstract perspective, the interactions on the Twitter platform shown in Table 2.3 can be assigned to two different categories depending on the type of relationship they enable: social and topical/semantic. This differentiation again highlights that the Twitter platform reveals characteristics of an information sharing as well as a social media platform. Theoretically, the discipline of socio-semantic networks also deals with the discoursive interactions of actants and Twitter data is often analyzed leveraging concepts developed in socio-semantic network research (see Hellsten and Leydesdorff (2020)). We refer the interested reader to Section 2.3 for more details.

Trends: The content shared on Twitter is constantly evolving, and the attention paid to different topics is shifting all the time. Thereby, some topics gain a significant level of popularity and become trending. To keep track of these trends, Twitter provides its users with a list of trending terms on the homepage. According to Twitter, by default, these trends are personalized, and the algorithm to determine the trends prefers currently popular topics and such that are currently emerging over topics that are popular on a regular basis or have already been popular for some time (Twitter, Inc., 2023b). Figure 4.1 shows an example of this list of trending topics. Next to the topic itself, the

corresponding tweet volume is also given in some cases. Also, as already stated, topics might only be trending with respect to a particular geographic region. This information is indicated in the referenced list of trends where appropriate.

Data access: Twitter provides an API to access their platform's data programmatically, such as tweets or details about user profiles. Exemplary data that can be retrieved through Twitter's API is shown in the listings given in Section 3.2.4. Access to this data lays the foundation for studying various phenomenons, such as the evolution of trends (see Section 4.1) or specific types of actor-networks (see Section 4.2). Unfortunately, at the beginning of 2023, Twitter severely limited their free API offering (Dotson, 2023).

2.2.2 Data quality

Working with social media data does not come without hurdles. One of these is the issue of appropriate data quality. A lack of good data quality inevitably leads to suboptimal analysis results and should, therefore, be considered for any data analysis project (see Batrinca and Treleaven (2015)). Given that the research related to data quality covers a broad spectrum of domains and aspects, at this point, we focus on social media data quality and, more specifically, on Twitter data quality. For details on how the Twitter media platform works and what features it covers, we refer to Section 2.2.1.

For social media data to achieve high data quality, upfront cleaning is required. According to Batrinca and Treleaven (2015), this cleaning process involves the removal of "[...] incorrect, inconsistent or missing information." Practically, steps like spell-checking, the removal of duplicates or fixing inconsistent dates might be part of that.

Regarding Twitter data quality, some specific issues are highlighted in the following. Still, it should be noted that this section does not claim completeness. Additional data quality issues might arise according to specific use-case requirements and should be considered accordingly. For example, one issue related to political data collected from Twitter is that the published content primarily comes from people with strong opinions (Huberty, 2015). Less opinionated users might not publish political content as actively. A dominance of "heavy users" is also stated by Tumasjan et al. (2010). Furthermore, when dealing with Twitter data, one should take into account that one is not analyzing a static system, and therefore, the reproducibility of analysis results might not be guaranteed. In particular, the platform's highly volatile user community should be considered. "Rapid growth and customer churn will very quickly alter the profile of its users, often in ways which are difficult to measure from the outside." (Huberty, 2015) Even though the root causes of these dynamics cannot be eliminated, the time-dependent nature of the data can and should be

considered during data analysis. Therefore, the network analytics model proposed in this thesis explicitly models the data's dynamics (see Chapter 3). In addition to the issues related to the evolving social media platform, analyzing its content also comes with specific hurdles. As such, one has to consider that the offered platform features might not be used consistently. For example, hashtags might be used inconsistently, i.e., in different variations or to refer to different topics (Cazabet et al., 2012). Furthermore, Twitter data quality issues concerning replies to private profiles or deleted tweets have been reported in the past (Cogan et al., 2012). Taking these issues into account is especially relevant for the analysis of conversations happening on Twitter. Section 5.4.1 gives more details on that.

2.3 Socio-Semantic Networks

In this section, the topic of socio-semantic networks is introduced. Conceptually, these kind of networks connect the semantic and the actor-centric perspectives on the modeled data. In later chapters, e.g., Chapters 4 and 5, an attempt is also made to adopt both of these perspectives. Figure 2.6 illustrates a socio-semantic network model and shows the duality of this approach. In a two-layer network, topics are modeled on one layer and users on the other, including their inter- and intra-links. Thereby, the semantic perspective (topics) and the actor-centric perspective (users) is represented.

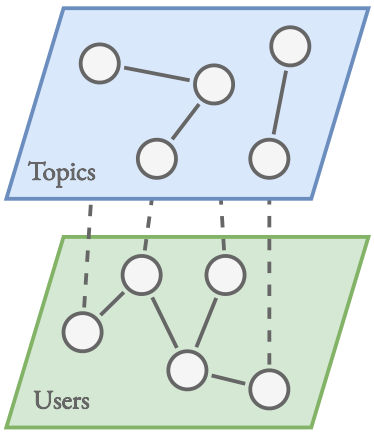


Figure 2.6: Multi-layer socio-semantic network illustration according to Logan et al. (2023, Figure 1)

In the following, after clarifying the used terminology, various studies from the socio-semantic network field are examined. For that, static and temporal analyses are taken into account.

2.3.1 Terminology

To clarify the terminology of socio-semantic networks, we refer to Gloor et al. (2009). They describe a socio-semantic network as a kind of two-mode network in which entities, such as individuals or groups, are linked by their shared discursive elements, e.g., words or sentences. As an example, in their work, Borge-Holthoefer et al. (2017) model Twitter data as a socio-semantic network with hashtags representing elements of discourse and Twitter users as the social entities accordingly. Socio-semantic networks can be seen as an extension of networks that solely represent text but do not explicitly model the underlying actor-network. Named text-representing networks, i.e., semantic networks, are largely based on the theories developed in the area of "Distributional Semantics"[5]. In this regard, Harris (1954) explains that semantic elements, e.g., words, are characterized by their context, i.e., co-occurring elements. Also, two elements with a similar meaning are more likely to occur in similar contexts than those with a less similar meaning. Such co-occurrences of linguistic elements are also leveraged in subsequent studies, e.g., for the trend networks based on co-occurring hashtags presented in Section 4.1. For additional background on semantic networks, we refer to the work referenced by Blanco and Lioma (2012).

In contrast, social networks model the interactions between actants. Latora et al. (2017) describe the basis of social network analysis as follows: "This discipline is based on representing a social system as a graph whose nodes are the social individuals or entities, and whose edges represent social interactions." (Latora et al., 2017, p. 31) Given the graph-based modeling approach, specific insights can be derived, such as finding important actors (Latora et al., 2017, p. 33).

2.3.2 Related work

In the following, work related to the study of socio-semantic networks is covered. It is separated into approaches that treat the networks as being static and those that specifically take their dynamics into account.

Static analysis: In their work Hellsten and Leydesdorff (2020) propose a socio-semantic network-based approach for the automated analysis of online debates. In line with the socio-semantic network methodology, they do not only consider the social networks among actors but also take the content or rather topics of the debates into account. Compared to related approaches, their focus is on the interactions of actors and topics. Further, they extend the existing actor-topic network model and also account for the authors of the content as another type of entity used during network analysis. By applying their methodology to two Twitter datasets, they can demonstrate the

[5] Distributional semantics – Wikipedia: https://en.wikipedia.org/wiki/Distributional_semantics (accessed 2023-10-30)

effectiveness of their approach. They leave it open for further research to extend their model in a way that takes even more entities and their interactions into account. Also, their method is not limited to the analysis of Twitter data but can be applied to related research fields as well. Further, building on the idea of heterogeneous networks, the work by Hellsten et al. (2020) studies scientific publications by leveraging multi-mode networks. In addition to the social and semantic dimensions, they also integrate epistemic information into their network model. More generally, their approach allows to consider any number of document attributes and their relationships. In their own words: "Our methodology is also relevant for combining heterogeneous nodes into networks of 'actants'." (Hellsten et al., 2020) This approach reveals similarities to the way documents and related entities are modeled in the network analytics model presented in Chapter 3. Specifically, it is similar to the entity networks proposed in Section 3.2.2. Still, the network's dynamics are not considered by Hellsten et al. (2020). Arroyo-Machado et al. (2021) build on the work of them and use scientific publications mentioned on Twitter to detect communities of social media users that share the same interests.

Temporal analysis: In an earlier work Gloor et al. (2009) mine web, blog, and online forum content to detect trends and the actors initiating these trends. For this, they mainly build on methods from the field of network science. Thereby, their approach consists of three main steps. First, they analyze the temporal betweenness centrality of concepts, such as companies or persons, as a measure of their importance. Secondly, to also account for the importance of the actors that talk about these concepts, they weigh different content according to the social network centrality of its originator. Finally, they keep track of the quality of the trends as extracted by the previously described steps by measuring the content's sentiment. Furthermore, Cointet and Roth (2009) in their work specifically investigate the dynamics of the socio-semantic network extracted from content of online blogs. They not only analyze how the knowledge distribution influences the topology of the social dimension of the network but also the other way around, i.e., how the topology of the network influences the spreading of information. The same authors also apply socio-semantic networks to study the temporal evolution of knowledge networks, in this case, a scientific community of embryologists and a group of political web bloggers (Roth and Cointet, 2010). Similarly, Gaumont et al. (2018) conduct an extensive study based on dynamic socio-semantic networks and analyze Twitter content related to the 2017 French presidential election. Thereby, they investigate the dynamics of opinions, as well as how political communities evolve. Furthermore, they look at the engagement of political actors, how political communities impact information diffusion, especially in the context of fake news, and also measure echo chamber effects. Also within the political context, Radicioni et al. (2021) study the discourse on migration policies taking place on the Italian Twittersphere and leverage socio-semantic networks for that. They not only analyze

the user community structures of the Twitter retweet network but also investigate the semantic hashtag networks related to these communities. Furthermore, they do not stick to a static analysis of the mentioned networks but also look at their temporal development. This way, they are able to analyze the dynamics of interactions between different political forces.

2.4 Data Management

Not only if one wants to build a customer-facing application to present data analysis results, but also for the analysis itself, the respective data needs to be managed in a functional way. Commonly, database management systems (DBMSs) are leveraged for that. These systems usually come with features to optimize the data management for the use case on hand. In this regard, the following Section 2.4.1 covers specific data indexing capabilities and Section 2.4.2 explains different data partitioning concepts. Further, Sections 2.4.3 to 2.4.6 cover various DBMSs that are later benchmarked in Section 6.3 to find an efficient and performant temporal network data management system.

2.4.1 Database indices

In a DBMS, specific subsets of data might be queried more frequently than others, or the stored data might be retrieved based on specific data accessing patterns, such as filtering the data with respect to selected properties. To improve the query performance in such scenarios, specialized data indexing techniques have been developed. In the following, two such data indexing methods, namely B-tree indices and GIN indices, are described.

B-tree index

The following section is largely based on Chapter 2 of the book by Petrov (2019) in which he gives an introduction to the topic of B-trees. Additional to that, the work by Comer (1979) gives and systematic overview of different B-tree variants. Originally, B-trees were developed by Bayer and McCreight (1970) and referenced work is largely based on this original work.

To better understand the benefits of B-trees used as efficient index structure within multiple database systems, one has to start at the related concept of binary search trees (BSTs) which are used for efficient in-memory key-value lookup. For an example of a BST see Figure 2.7. Each node in a BST consists of a key, an associated value and two pointers to its child nodes. The starting node at the top of the tree is called the "root node". As subtrees are divided into a (left) part with values lower than the parent value and a (right) part with values larger than the parent value, traversing

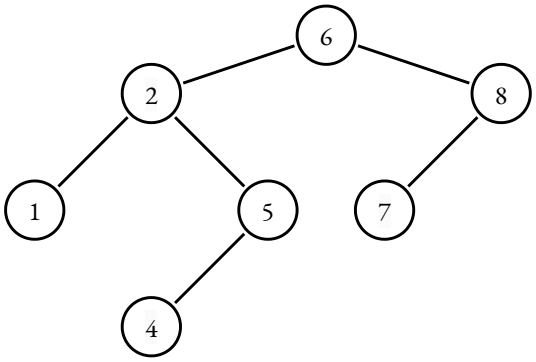


Figure 2.7: Exemplary binary search tree

the tree allows to efficiently search for a given value. For this lookup to be efficient it is necessary that the tree is "balanced", meaning that its hight follows $\log_2(n)$ with n being the total number of inserted elements. As random additions to the tree lead to an unbalanced situation, elements have to be shifted around in these cases to rearrange an balanced state again.

For disk-based storage, random seeks following pointers are quite expensive. By using BSTs as on-disk data structures, these seeks lead to several performance issues. This is especially relevant because disk-based storage is often used for database systems today. First of all, due to the low "fanout" of 2 which describes that a maximum of 2 children is allowed per node, a BST has to be balanced rather frequently which leads to relocations of pointers and additional maintenance costs. Also, compared to higher fanout trees traversing a BST is more costly as the height of the tree also follows $\log_2(n)$. Further, it is not guaranteed that child nodes are in close proximity to their parent. On disk, this might lead to the situation that a child pointer spans across multiple pages. Such disk seeks during the traversal of the tree then become quite expensive. Therefore, to overcome named issues two properties are targeted which also leads to the concept of B-trees: 1. high fanout for an increased locality of neighboring keys 2. low height for less seeks during tree traversal.

As shown in Figure 2.8 B-trees have a larger fanout compared to BSTs and again three hierarchy levels: a root node, internal nodes, and leaf nodes. Compared to BSTs they also have a lower height. For a B-tree, the keys of the nodes (the indexed values) are sorted, and each node contains dozens of items. Therefore, only for level jumps disk seeks are needed and the lookup within the nodes itself can be done efficiently, for example by using binary search. B-trees work efficiently not only for point queries (=), but also for range queries ($<$, $>$, $\leq$, $\geq$). In general, the complexity of searching elements within B-trees can be stated to be $O(\log(n))$.

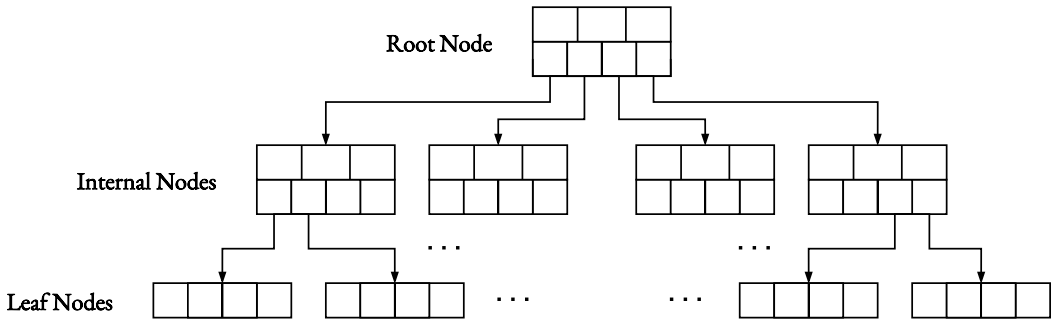


Figure 2.8: Illustrated structure of a B-tree according to Petrov (2019, Figure 2-9)

Next to the high performance, the B-tree index implementation of PostgreSQL comes with additional advantages. These advantages include the capability of defining the ordering of the indexed data, the option to create multi-column indices, clusterability, and the possibility to configure how `NULL`s are handled within the index (Rogov, 2019).

GIN index

The term GIN index stands for "Generalized Inverted Index" (PostgreSQL Global Development Group, 2023b), referring to its capability of indexing complex data types, such as arrays and JSON documents, in contrast to simple data types like integers and strings, which makes it a more "generalized" form of an inverted index. It is tailored to support search queries of element values occurring in the indexed composite types, such as a search for a word occurring in a set of text documents. Internally the GIN index stores tuples of named element values and a so-called "posting list". This posting list itself is again a set of row IDs that identify the composite data in which the according element appears. Thereby, each element value referred to as "key" appears only once in the GIN index. In contrast, the same row ID might occur multiple times in different posting lists, such as a word occurring in multiple text documents.

As shown in Figure 2.9, the mentioned posting list is stored in different ways depending on its size. If its size, together with the indexed key, does not surpass the defined size of the leaf node tuple, a list of heap pointers to the row IDs is stored (PostgreSQL Global Development Group, 2023a). In contrast, a heap pointer leading to a B-tree of heap pointers is stored if the posting list is too large to be stored in the leaf node page. Further, the GIN index, as implemented in PostgreSQL, comes with a pending list of queued GIN index updates. With that list of pending updates, these can be applied batch-wise instead of individually, which improves the update efficiency. This

[6] GIN Internals: `https://www.postgresql.org/docs/current/gin.svg` (accessed 2023-12-22)

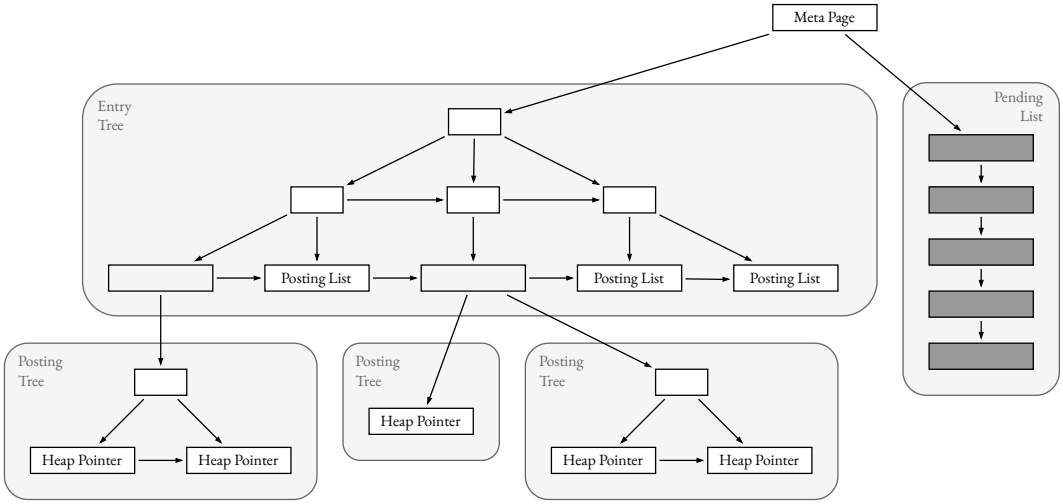


Figure 2.9: Structure of a GIN-index according to PostgreSQL Global Development Group (2023a, Figure 70.1.)[6]

performance improvement is crucial, given that updates to the GIN index are pretty expensive. Even a single added item/document might cause many posting lists to be updated, given that many keys might be contained in the added item.

2.4.2 Database partitioning

"Partitioning in database design is the process of assigning a logical object (relation) from the logical schema of the database to several physical objects (files) in a stored database." (Navathe et al., 1984) Informally, during partitioning logically related data is separated into multiple chunks and stored in physically different places. These techniques are relevant to multiple computing environments. Such use cases include the distribution of data across multiple computing nodes in a distributed database design, partitioning of data according to primary and secondary memory, as well as distributing data across multiple physical devices (Ceri et al., 1982; Navathe et al., 1984). It aims at faster data access and improved query performance. Therefore, to partition data in a way that data access performance is improved, it is crucial to investigate typical data access patterns (Ceri et al., 1982). In a relational database setting, where data is stored in multi-columnar tables of records, data can be partitioned horizontally (Ceri et al., 1982), i.e., into sets of records, or vertically (Navathe et al., 1984), i.e., by sets of columns. Navathe et al. (1984) describe the two partitioning techniques as follows: "*Vertical partitioning* subdivides attributes into groups and assigns each group to a physical object. *Horizontal partitioning* subdivides object instances (tuples) into groups, all having the same attributes of the original object."

2.4.3 PostgreSQL

PostgreSQL[7] (a.k.a. "Postgres") (Stonebraker and Rowe, 1986) is an open source object-oriented RDBMS. It is a popular database system (solid IT gmbh, 2023) and is commonly used in industrial, as well as academic settings (see Ding et al. (2019) and Miler et al. (2014)). PostgreSQL comes with a rich set of features, such as B-tree-based access methods (see Section 2.4.1) and partitioning of data according to different memory levels (Stonebraker and Rowe, 1986). Further, it is explicitly designed to work with time-varying data (Stonebraker and Rowe, 1986). Not only current tuples, but also historical data can be retrieved. Also, parts of the time-varying data can be materialized and stored in the form of snapshots for faster access performance of subsequent queries. Different use cases might come with different data requirements, such as geographic information systems require to work with geographic objects (Stonebraker and Rowe, 1986). PostgreSQL facilitates such customization and extensions in multiple ways. First of all, its open source nature allows to inspect the source code and helps to understand the software's internal workings. Secondly, PostgreSQL allows to define custom object types, along with specialized operators and access methods (Stonebraker and Rowe, 1986). Multiple PostgreSQL extensions to deal with domain-specific use cases exists, such as TimescaleDB to work with time series data.

2.4.4 TimescaleDB

TimescaleDB is a PostgreSQL extension to work with time series data (Timescale, Inc., 2023). It enriches the native PostgreSQL capabilities to deal with time-varying data with more advanced time series and analytics features. These features include among others auto-updating materialized views, enhanced data compression, and data distribution across multiple databases. As core concept, TimescaleDB leverages so called "hypertables" which are native PostgreSQL tables that are horizontally partitioned into subsets of tuples, so called "chunks". Partitioning is based on a given time attribute. The size of these chunks are defined by the `chunk_time_interval` parameter which determines the time window or snapshots size of respective data subsets. Hypertables can also be distributed across a cluster of multiple databases. In such a setup, hypertables are then called *distributed* hypertables. For a more visual understanding of how PostgreSQL tables are partitioned into time-based chunks in the form of hypertables, see Figure 2.10.

[7] PostgreSQL: The world's most advanced open source database: https://www.postgresql.org (accessed 2023-07-07)

[8] TimescaleDB hypertable visualization: https://assets.timescale.com/docs/images/getting-started/hypertables-chunks.webp (accessed 2024-01-09)

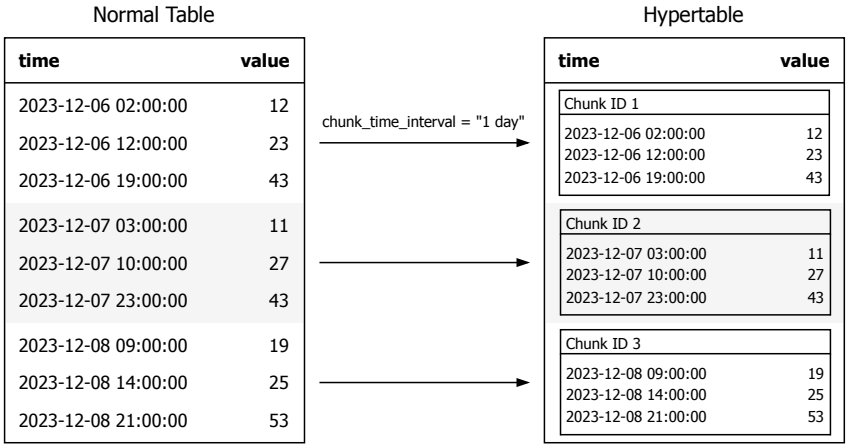


Figure 2.10: Visualization of the hypertable concept as used by the TimescaleDB extension according to Timescale Inc. (2024)[8]

2.4.5 Apache AGE

Apache AGE[9] is a PostgreSQL extension which enhances the database's native features by capabilities typically found in graph databases. Most notably, the extension allows the execution of openCypher[10] queries against the stored graph data. In addition to the PostgreSQL extension itself, the Apache AGE project provides users with a web application for the graph-based data exploration and analysis. The user interface is termed "Apache AGE Viewer". Technically the extension follows the LPG (Angles, 2018) model to represent stored graph data. In version `1.1.0` of the extension, each created graph is stored in a separate namespace and related objects (nodes and vertices) get assigned unique ids. These objects are redundantly stored in separate tables according to their labels, as well as in the tables `_ag_label_vertex` and `_ag_label_edge` which contain all the nodes and edges respectively (Farias, 2023). Properties of nodes and edges are stored in the form of JSONB data in an additional column within the same tables.

2.4.6 Neo4j

Neo4j[11] is a fully-featured graph analytics platform which also includes the Neo4j graph database. The GDBMS is offered in a free-to-use community edition, which can be self-hosted and run on commonly used platforms, such as Linux or Windows. Given the system's Java-based implementa-

[9] Apache AGE: `https://age.apache.org` (accessed 2023-07-07)
[10] openCypher · openCypher: `https://opencypher.org` (accessed 2023-07-25)
[11] Neo4j Graph Database & Analytics | Graph Database Management System: `https://neo4j.com` (accessed 2023-07-25)

tion, it is required to have a compatible Java[12] Virtual Machine (JVM) installed and set up on the respective operating platform. Independent of the underlying computing platform, the database itself comes with a rich set of configuration options, such as the JVM's heap size or the maximum number of memory a single transaction can use. Regarding data modeling, the Neo4j database follows the property graph model (Angles, 2018) and the stored data can be queried using the Cypher language. Neo4j's Cypher implementation itself is based on openCypher.

[12] Java | Oracle: https://www.java.com (accessed 2023-10-10)

3 Network Analytics Model

This chapter describes the data model used for the media analyses presented in subsequent chapters. The model builds on the theoretical basis outlined in Chapter 2 and mainly leverages concepts from network science, especially temporal graph analysis. Thereby, the model is tailored to the use case of media analysis and corresponding requirements.

In the following, Section 3.1 elaborates on these requirements. Subsequently, in Section 3.2, the network model is described concerning its topology. Then, in Section 3.3, this model is extended to capture the temporal evolution of modeled data. Furthermore, Section 3.4 describes network projections as a valuable tool for analyzing modeled networks. Next, Section 3.5 introduces a system of different network perspectives. According to different granularities, both topological and temporal, different points of view on the data can be defined. Finally, the chapter is concluded by discussing the proposed model in Section 3.6.

3.1 Requirements

Several requirements for a media analysis model can be formulated based on practical experiences gained during the EPINetz project. It is crucial to consider these requirements carefully as the model must be used for different data sources and in various analysis scenarios. Even though the requirements are based on practical insights, completeness is still not guaranteed, and additional requirements might arise from the specific use case. The general requirements of a target model are outlined in the following:

Flexibility and extensibility: The model must be flexible and extensible in a way that it needs to be applicable to different analysis use cases. Such use cases include, but are not limited to, trend analysis as described in Chapter 4 or the analysis of conversations (see Chapter 5). Further, flexibility is not only required concerning the usage of the model but also regarding the modeled data itself. The model needs to cope with *heterogeneous* kinds of data, and this heterogeneity comes in different forms. First, data might be heterogeneous regarding its formats, such as differently structured JSON files or various forms of HTML documents. This condition needs to be consid-

ered, especially in cases where data is collected from different media platforms, e.g., social media and news outlet websites. Secondly, with a more detailed perspective on the data, heterogeneity might also be related to how the collected data is structured. Some data might come in a structured format following a well-defined schema. In contrast, other data might only be semi-structured or even unstructured, which is especially true for social media data (Gerhardt et al., 2012). Finally, heterogeneity might come from semantic differences between the data. It might cover multiple domains, such as user profiles, social media posts, or publication metrics. Altogether, a target model should come with sufficient flexibility and extensibility to integrate multiple heterogeneous data sources and give the practitioner a more structured and homogeneous perspective on the modeled data to leverage it for various analysis use cases.

Time sensitivity: As another requirement of the data and analysis model, one can state its capability to work with *temporally evolving* data. Media data is inherently time-dependent as discussed topics, participating actors, or even the platforms themselves are constantly changing. Therefore, a model needs to capture the dynamics of the data. Temporal sensitivity is crucial for many use cases, such as information diffusion, trend and discourse analysis or event detection. In this regard, the use cases might rely on different temporal resolutions. For some scenarios, temporal information given on a daily basis might be sufficient, whereas other scenarios might demand higher temporal resolutions, such as hours or minutes. Consequently, the model should be able to cope with the different temporal resolution requirements.

Reproducibility: The model needs to guarantee *reproducibility*. Given the same data, based on deterministic analyses should lead to the same results. The same is true if analyses are done at different points in time or are re-produced after an initial run. Also, if data is processed in a way that less structured data is modeled to follow some schematic structure or if more fine-grained entities are extracted from given data, a data practitioner should still be able to trace back these entities to the original, "raw", data. This way, post-analysis investigations would allow checking whether processing was done as intended.

Information value: In contrast to data, *information* is understood as organized data that allows to answer simple questions about the world (Gartner, 2016, p. 10; Ackoff, 1989). In this sense, less organized data needs to be processed to be effectively used as information, e.g., in a decision-making process. Information, as opposed to data, is on a higher "knowledge" level. The proposed model should provide a framework to analyze processed data and effectively use it as information and, therefore, increase its "knowledge" level. For example, suppose one wants to ask questions about the mentioning activity of actors on a social media platform. In that case, it is of more

informative value to have the data modeled as a mention network, in contrast to the unprocessed social media posts.

Efficiency: For a data analytics model to be used in a practical setting, it should guarantee efficient usage of available computational resources. Still, in this chapter, the focus is placed on the layout of the proposed model and less on its efficient application. Its actual implementation is elaborated on in Chapter 6 regarding resource-saving storage and computation.

3.2 Network Model

For media analytics use cases, data often comes from different sources in an unstructured and heterogeneous way. Given their modeling flexibility, networks as introduced in Section 2.1 and especially HINs can cope with these conditions, which is why they are used as the elementary data model. The following sections describe in detail how unstructured social media data can be modeled as such an information network (Sections 3.2.1, 3.2.2, and 3.2.3). Also, in Section 3.2.4, the complete process is detailed based on a Twitter data example.

3.2.1 Document network

On an abstract level, a dataset used in a media analytics scenario can be described as a collection of *documents*. These documents might be of different types, such as news articles or social media posts. Nevertheless, abstractly, they are referred to as documents. Further, these documents might also be related, and the relationships might be relevant to the individual analytics use case. For example, a tweet might be a retweet of another tweet, or a news article might reference another. Defining relevant relationships is a manual task and requires domain-specific knowledge. Nevertheless, the present model is not restricted to a pre-defined set of documents and relationships but treats them as abstract concepts. Formally, the specification of a document network is given in Definition 3.1, and Figure 3.1 visualizes the corresponding network schema.

Definition 3.1 (Document network)**.** With a given set of documents D and links among these documents, referred to as L_d, a *document network* is defined as a directed graph $G_d = (D, L_d)$. Each of the network's links $l_d \in L_d$ connects two documents, $d_i \xrightarrow{l_d} d_j$, and is denoted as a tuple of these documents: $D \times D \rightarrow L_d$.

Already at this point, the document network can be described as HIN. Not only may the network contain different types of documents like news articles or tweets, but documents might also be

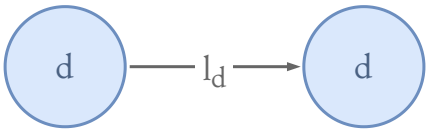


Figure 3.1: Document network schema. Document nodes are marked blue, and links in grey.

linked by different types of relationships. Table 3.1 lists examples of document node types and corresponding relationship types.

Table 3.1: Node and link types of exemplary document networks

Document node type	**Link type(s)**
tweet	retweet, reply
Twitter user profile	following
news article	reference
webpage	hyperlink
Instagram user profile	following

Remark. Notably, one has to distinguish between the node and relationship types described by the network model and those of actual network instances. Abstractly, analyzed documents are modeled as nodes connected by links in the according document network. Nevertheless, in an actual document network instance, these documents and links might be of different types (see Table 3.1). Still, the topological schema of the network might be the same. In the following sections, modeling follows the same approach. Keeping the model at this abstraction level allows different and heterogeneous datasets to be modeled and analyzed, as stated in the requirements Section 3.1.

3.2.2 Entity network

The documents described in Section 3.2.1 might contain additional data relevant to a given use case. This data should explicitly be modeled as well. For that, additional *entities* are extracted from these documents, e.g., hashtags and usernames contained in tweets, and are then added to the network. Different types of extracted entities are modeled as different node types. Thereby, we refer to entities as an abstract concept, similar to that of "objects". Depending on the exact use case, different entity types are relevant. Deciding which entities are relevant and should be extracted from the documents needs manual intervention and requires domain-specific knowledge. More detailed guidance on the selection of valuable entities is given in Section 3.2.2. Also, the entity

extraction process is not explicitly specified but treated as an abstract step with documents as input and extracted entities as output. The model only describes how extracted entities are incorporated into the network. For example, methods known from natural language processing (NLP) might be used to extract entities from text data contained in the documents. See the survey by Yadav and Bethard (2019) for an overview of the named entity recognition topic. Formally, the extraction process is specified in Definition 3.2, and Table 3.2 gives examples of documents, together with contained entities that might be extracted for analysis purposes.

Definition 3.2 (Entity extraction)**.** Given a set of documents D, entities denoted as E can be extracted from these. A single extracted entity is referred to as e: $e \in E$. Further, the extraction process of entities from documents is defined by a function that maps each document to a subset of entities: $\epsilon : D \to 2^E$.

Table 3.2: Examples of documents, contained entities, and corresponding relationship types

Document	**Entities**	**Relationship type(s)**
tweet	hashtag, username, URL	usage, mention, usage
Twitter user profile	country	residence
news article	keyword, tag	occurrence, description
webpage	URL	reference

Remark. Relationships, as outlined in Table 3.2, often imply some hierarchy and are, therefore, formulated in a way that implies directionality. Still, edge types that describe the relationship following one or the other direction can be formulated. For example, compare the two semantic descriptions of a relationship between a tag and a news article: "A tag *describes* a news article." vs. "A news article is *described by* a tag." To take this into account, links between entities and documents can, in general, be characterized as bidirectional. Nevertheless, only a single instance of the relationship is considered part of the network. The link going in the other direction can be derived as the inverse of that relationship. As the default choice, the edges from the document to the extracted entity are taken into account. Other network models also consider inverse relationship types, such as the work by Milani Fard et al. (2019) on meta path prediction in temporal HINs or the work by Bronson et al. (2013) in which they present a system to access the social graph data of the Facebook platform.

We refer to the document network extended by the extracted entities as *entity network*. In this network, the entities are linked to the documents they are extracted from. These relationships are labeled and come with a semantic meaning. Definition 3.3 formalizes the entity networks.

Definition 3.3 (Entity network)**.** The documents D and extracted entities E as nodes, together with the links among documents L_d and links between documents and entities L_e, make up the *entity network* G_e. It is defined as directed graph $G_e = (\{D \cup E\}, \{L_d \cup L_e\})$. Each link $l_e \in L_e$ connects an entity to the document it is extracted from: $d \xrightarrow{l_e} e$.

At this stage, the network contains the documents and entities as nodes, as well as the relationships among documents and between documents and entities as links. Given that the same entities might be extracted from different documents in the entity network, two documents might also be linked via shared entities. Such latent relationships can be described by meta paths as formerly defined in Definition 2.19. These meta paths follow the structure of $d \xrightarrow{l_e} e \xrightarrow{l_e^{-1}} d$.

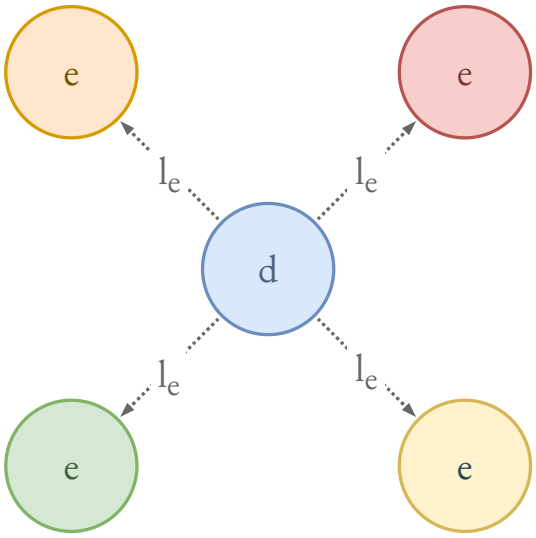


Figure 3.2: Schema of a single document entity network. Entity nodes are marked orange, red, green, or yellow.

As illustrated in Figure 3.2, a subnetwork describing a single document and extracted entities follows the topology of an ego-network with the document at its center. Linking extracted entities with the original document allows retracing the extraction process, which is essential for reproducibility purposes (see Section 3.1). Entities included in the network can be traced back to their origin even after the extraction is completed. This way, extraction procedures can be reviewed, and entities extracted from a specific document can be removed from the network at any time. Further, the entity extraction process and entity network model satisfy the "flexibility and extensibility", and "information value" requirements as outlined in Section 3.1. The entities can be extracted from differently structured, or unstructured data, which comes with much flexibility as the model can also deal with this kind of data heterogeneity. Also, even after an initial extraction process, other entities can extend the network. Further, the entity network model transforms the processed

data to be more structured, which is crucial for many analytics use cases and transforms the data into more valuable information.

Which entity types should be considered for entity networks?

The selection of entities that should be considered during the entity network construction process depends on the exact use case and analysis scenario. Therefore, at least to some extent, domain-specific knowledge is required. Nevertheless, some guidelines regarding the appropriate entity types can be outlined. First, only entity types with a limited number of possible values also denoted as *nominal data*, should be considered. This restriction helps to avoid polluting the network with many entity nodes but instead focuses on the most semantically expressive ones. These might, at least on average, be linked to more document nodes, leading to higher connectedness within the network. Such entities might be contained in the data itself, or its associated metadata which is typically described as "[...] data about data." (Gartner, 2016, p. 2) For example, raw Twitter data comes with associated metadata such as the username of the account that posted the tweet, its date and location, and statistics about the user, such as the follower count (Gartner, 2016, Figure 1.1). Especially *structural* metadata, which is defined as "[...] metadata designed to help us discover and locate the data it refers to" (Gartner, 2016, p. 6) should be considered for the selection of appropriate entity types as it already provides a good indication of possible entity categories and filters to analyze the data.

3.2.3 Document attributes

Some of the information in the data might not be appropriately modeled as separate node or entity type but is better suited to be represented by an attribute assigned to related document nodes. Examples of such attributes include statistics like the number of followers related to a Twitter profile or the number of comments belonging to a news article. Table 3.3 shows more details about exemplary document node attributes. Formally, the attribution process can be described as mapping $\pi_{attribute}$ between the set of documents D and the domain of the respective attribute. Not all types of documents have the same attributes. Therefore, the attribute mappings are partial functions. They are undefined for the documents that do not have the respective document node attribute. For example, all news article document nodes are undefined for the attribute, which describes the number of followers on Twitter.

Especially time-related attributes, like the publication date of news articles or the timestamp of tweets, are essential to model the dynamics of described networks. This modeling step is described in detail in Section 3.3.

Table 3.3: Examples of attribute mappings with $\mathbb{T}$ denoting the domain of time.

Description	Function	Codomain
number of Twitter follower	$\pi_{twitter\ follower}$	$\mathbb{N}$
publication date	$\pi_{publication\ date}$	$\mathbb{T}$
number of news article references	$\pi_{article\ references}$	$\mathbb{N}$
number of retweets	$\pi_{retweets}$	$\mathbb{N}$

3.2.4 Twitter network example

In the following, we illustrate the model described above using a network extracted from Twitter data. To begin with, Listing 3.1 shows data of a tweet as retrieved by the Twitter API v2. Different parts of the raw JSON data are highlighted as they represent the tweeted document and related entities modeled in the respective entity network. These node types are also given as examples in Table 3.2. Further, the publication date of the presented tweet, "created_at", is highlighted as this information is represented as an attribute belonging to the tweet document node (see Table 3.3). Leveraged colors align with the color coding used in Figure 3.2.

Listing 3.1: Excerpt of raw Twitter (tweet) data (Twitter, Inc., 2022, modified). The color coding corresponds to the coloring leveraged for the network illustrations.

```
"tweets": [
   {
      "conversation_id": "1304102743196356610",
      "id": "1304102743196356610",
      "public_metrics": {
         "retweet_count": 31,
         "reply_count": 12,
         "like_count": 104
      },
      "entities": {
         "mentions": [
            {
               "start": 146,
               "end": 158,
               "username": "suhemparack"
            }
         ],
         "hashtags": [
            {
```

```
                "start": 8,
                "end": 19,
                "tag": "TwitterAPI"
            }
        ]
    },
    "text": "The new #TwitterAPI includes some improvements to the Tweet
        ↪ payload. You're probably wondering – what are the main differences?
        ↪ 🧐\n\nIn this video, @SuhemParack compares the v1.1 Tweet
        ↪ payload with what you'll find using our v2 endpoints. https://t.co/
        ↪ CjneyMpgCq",
    "created_at": "2020-09-10T17:01:37.000Z",
    "author_id": "2244994945"
  }
]
```

The respective entity "ego-network" (see Section 3.2.2) extracted from the raw tweet data as shown in Listing 3.1 is given in Figure 3.3. The tweet document node is shown at the center with the tweet ID as the node identifier and its publication date assigned as an attribute. Further, two extracted entities, a hashtag, "#TwitterAPI", and a mentioned user, "@SuhemParack", are linked to the document node. The network only consists of a single document, and no links between documents exist. Nevertheless, as outlined in Section 3.2.1, the proposed model also deals with relationships between multiple documents, in this case, tweets. To illustrate these relationships Listing 3.2 gives an additional, fictitious tweet document excerpt following the structure of tweet replies used by the Twitter API v2 (Twitter, Inc., 2022).

Listing 3.2: Fictitious Twitter (tweet reply) data (Twitter, Inc., 2022, modified)

```
"data": [
  {
    "conversation_id": "1304102743196356610",
    "text": "See how @SuhemParack is using the new Twitter API.",
    "referenced_tweets": [
      {
        "type": "replied_to",
        "id": "1304102743196356610"
      }
    ],
    "entities": {
      "mentions": [
        {
```

```
                "start": 8,
                "end": 20,
                "username": "suhemparack"
            }
         ]
      },
      "id": "1304102743196356689",
      "public_metrics": {
         "retweet_count": 9,
         "reply_count": 3,
         "like_count": 26
      },
      "author_id": "2544984325",
      "in_reply_to_user_id": "2244994945",
      "created_at": "2020-09-11T16:05:31.000Z"
   }
]
```

Figure 3.4 shows the network of Figure 3.3 extended with the data extracted from Listing 3.2. Now, the network contains another document node connected to the first one by a "reply" relationship. Further, the two document nodes are indirectly linked by their shared user

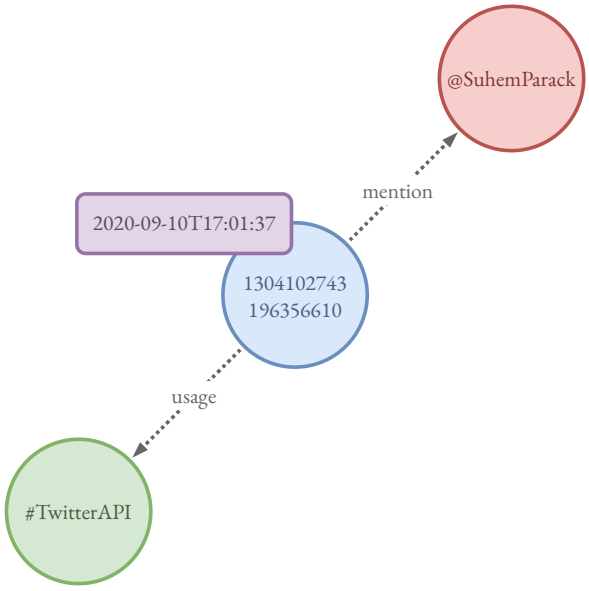


Figure 3.3: Entity network extracted from tweet data as shown in Listing 3.1. Timestamps are marked purple.

node, labeled as "@SuhemParack". Latent relationships like these are described by the meta path $tweet \xrightarrow{mentions} user \xrightarrow{mentioned\ by} tweet$ or, in this case, the actual meta path instance $tweet_1 \xrightarrow{mentions} SuhemParack \xrightarrow{mentioned\ by} tweet_2$. The two connected tweets are further specified by their IDs given as respective document node attributes: $\pi_{id}(tweet_1)$ = 13[...]10 and $\pi_{id}(tweet_2)$ = 13[...]89.

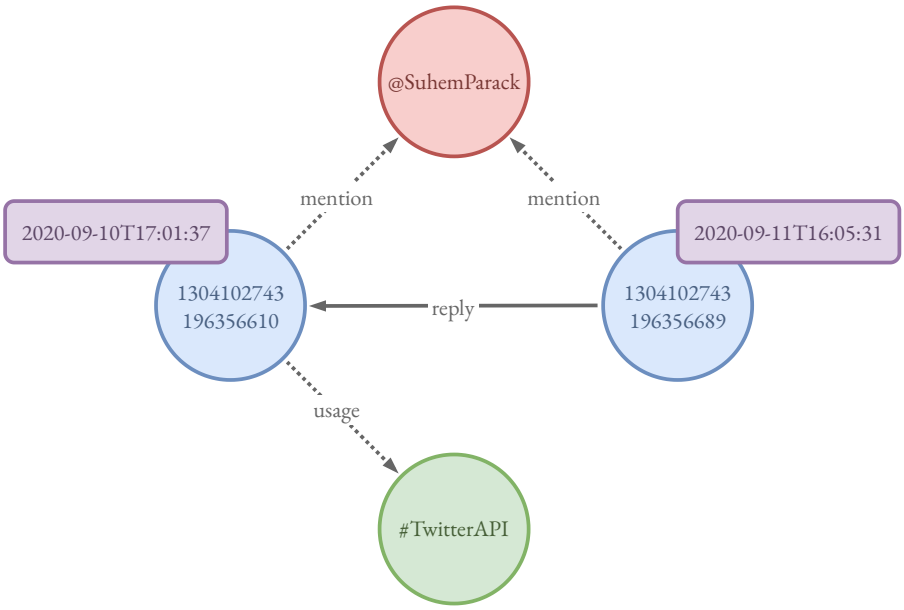


Figure 3.4: Extended tweet entity network

3.2.5 Related work

Many document corpora, such as social media posts, research publications, or web pages, can be modeled as document networks. In this sense, a document network representation models the documents as nodes connected by use case-specific relationships (e.g., social media following, citations, or hyperlinks). Notably, the terminology around "document networks" is not used consistently throughout different academic disciplines. Some document network models represent the documents themselves as networks. Thereby, the networks consist of the words contained in the documents. Other node types, such as named entities, keywords, or URLs, might also be used. In the document networks, the nodes are then linked by use case-specific and semantically expressive relationships such as sentence distance or similarity. Works that leverage such document network model include Rafi et al. (2014), Rousseau and Vazirgiannis (2013), or Spitz and Gertz (2016). Described document networks are often referred to as *semantic networks* (see Section 2.3).

Further, Spitz (2019) describes in his thesis a graph-based approach to model document collections via implicit networks constructed based on term and entity co-occurrences. His model also covers enriching the extracted graph with data from external knowledge bases. Additionally, by proposing a generalization of his basic model, which leverages hypergraphs, higher-order co-occurrences are also taken into account. In contrast to the model proposed in this thesis, the model by Spitz (2019) is tailored to the use cases of natural language analysis, and it mainly leverages co-occurrence relationships. Our model specifically models various types of semantically meaningful relationships and focuses on entities extracted from documents, including their relationships. Also, we explicitly consider links on a document level.

Apart from terminological inconsistencies, document networks have in the past been successfully leveraged in different use case scenarios, such as topic modeling (Chang and Blei, 2009; Yang et al., 2016; Spitz and Gertz, 2018) or document clustering (Rafi et al., 2014). In their work, Schuhmacher and Ponzetto (2014) use the DBpedia knowledge graph (Auer et al., 2007) to model text documents as semantic networks based on occurring entities and their semantic relationships. Following their approach, a document's semantic network consists of the named entities mentioned in the document and which are known from DBpedia. Further, these entities are linked via relationships that are as well found in the knowledge base. Relationships could be direct links or indirect multi-hop paths between two entities. Schuhmacher and Ponzetto (2014) successfully apply their model to entity ranking tasks and to specify document similarity.

Furthermore, in their work, Sun et al. (2010) apply the task of community detection to dynamic HINs. Thereby, they focus on networks that follow a star-like schema. The objects in the center of the network are referred to as "target objects" (e.g., research publications), and the others are described as "attribute objects" (e.g., authors, conferences). This pattern is similar to the entity networks described in Section 3.2.2. The subgraph with the document at the center and the extracted entities on edge follows the same topology. In this sense, documents can be seen as target objects and the extracted entities as attribute objects.

Further related concepts regarding the document attributes outlined in Section 3.2.3 exist. Most notably, the approach of assigning additional attributes to nodes and edges in a graph is described by the property graph model (Rodriguez and Neubauer, 2010; Angles, 2018). Graph database systems commonly implement this model (Angles, 2018). Rodriguez and Neubauer define a property graph as "[...] directed, labeled, attributed, multi-graph [...]" (2010, p. 3). Property graph edges come with an orientation, and multiple edges can exist between the same two vertices. Vertices might be typed and/or might have attributes. These attributes correspond to the document attributes described in Section 3.2.3.

3.3 Modeling Dynamics

As already stated in Section 3.1, for a holistic analytics model, it is crucial to consider the temporal nature of social media data. Therefore, in this section we cover how the model presented in Section 3.2 is extended to deal with the data's temporal dynamics. For this, Section 3.3.1 covers an extension to the network model in the form of temporal network snapshots and Section 3.3.2 compares different temporal-sensitive analysis methods.

3.3.1 Temporal network snapshots

Documents, as described in Section 3.2.1, usually come with a characteristic timestamp as they are created, changed, or published at a given point in time. This temporal information is represented in the outlined model as document node attributes. The according attribute mapping assigns timestamps to each document in the analyzed dataset: $\pi_{timestamp} : D \rightarrow \mathbb{T}$, where $\mathbb{T}$ refers to the domain of time, which is presumed to be $\mathbb{R}^+$ (see Xu (2021)). Documents without temporal information are not considered since these cannot be localized in time, and it would be unclear when respective information should be included in the network model.

Remark. As an alternative to modeling the temporal dimension as a document node attribute, one might argue that a particular node type should represent time. Other nodes would be linked to respective time nodes to indicate their timestamps. Nevertheless, as time is a continuous variable and not of nominal scale, its values cannot, without restriction, be limited, e.g., to days, weeks, or months. Therefore, to preserve generalization, time should not be represented as a separate node type in the network model but instead as a node attribute to follow the argumentation outlined in Section 3.2.2.

The temporal information provided by the document timestamps allows us to extend the network model presented in Section 3.2 and to incorporate its temporal dynamics. For this, the document network is split into temporal snapshots. Each *document network snapshot* consists of only the subset of documents and their relationships with a timestamp, that falls into the time window the corresponding snapshot covers. Converting the temporal network into a temporal sequence of static snapshots usually facilitates their analysis (Holme and Saramäki, 2012).

Definition 3.4 (Document network snapshot)**.** The time T is split into multiple time windows to derive the document network snapshots. Thereby, a time window $w_i = [t_i, t_{i+1}]$ covers the time range between t_i and t_{i+1}. A document d is part of the respective network snapshot if $t_i \leq \pi_{timestamp}(d) \leq t_{i+1}$ holds. The document network snapshot of the time window w_i is referred to

as $G_d^i = (D^i, L_d^i)$ with D^i denoting its document nodes and L_d^i the respective links among these documents.

It is noted that the time windows used to split the network into snapshots can be of different sizes. The interval $t_{i+1} - t_i$ might not necessarily be as large as the interval $t_{i+3} - t_{i+2}$. Nevertheless, time windows have to be at least temporally ordered. For the two time windows $w_i = [t_i, t_{i+1}]$ and $w_{i+1} = [t_{i+2}, t_{i+3}]$, $t_{i+1} < t_{i+3}$ has to hold. Notably, this definition does not imply the condition $t_{i+1} <= t_{i+2}$. Therefore, two subsequent network snapshots might also be temporally overlapping. For example, the snapshots might be defined in an *accumulative* way with $t_i = t_{i+2} = t_{i+4} = ... = t_0$. Following such an approach means that for each snapshot, all past documents with respect to the snapshot's ending time are taken into account and are part of the network.

Abstractly, splitting the network into temporal snapshots is based on the discretization of time. This discretization can be defined either *directly* by specifying time windows that serve as the basis for the network aggregation or *indirectly* by splitting time according to another non-temporal metric. As such, numerous ways of defining network snapshots exist. In general, determining an appropriate sampling approach depends on the analysis use case and might require domain-specific knowledge. Multiple sampling methods are outlined in the following:

- **Fixed time window sizes:** Straight-forward, time can be split into equally sized time windows for discretization. Thereby, the time interval for each time window is of the same size: $t_{i+1} - t_i = t_{i+3} - t_{i+2}$. This approach can be seen as *direct* discretization strategy. In contrast, unequally sized time windows are unsuitable without another metric to define the discretization, as a comparison of networks across snapshots is invalid or at least less meaningful. However, fixed time window sizes that might be used are days, weeks, or months. Also, the same network might be sampled according to different time window sizes. Different time window sizes relate to different temporal resolutions, as smaller windows lead to a higher temporal resolution. For a more elaborate discussion on the implications of the different temporal resolutions, see Section 3.5.2. Further, it has to be noted that for the accumulative creation of network snapshots, sampling by fixed time windows is not possible. By definition, subsequent snapshots always cover a larger period than the previous snapshots. Nevertheless, keeping the additional period covered by the following network snapshot constant might be an appropriate extension of the fixed time window sampling employed in an accumulative setting: $t_{i+3} - t_{i+1} = t_{i+5} - t_{i+3}$.

- **Volume-based:** Given the document network of time window w_i as $G_d^i = (D^i, L_d^i)$, the volume-based approach can be defined by the constraint that $|D^i| = |D^{i+1}|$ needs to be fulfilled. Each temporal document network snapshot contains the same number of docu-

ments, e.g., tweets or news articles. Such an approach is especially suitable for use cases in which fixed time window sizes would lead to a highly skewed distribution of documents across the snapshots. In these scenarios, some snapshots contain a lot more documents than others. This unequal distribution of documents makes a comparison of snapshots less meaningful. With the volume-based sampling of the networks, their sizes are normalised, and comparison becomes useful again. *Indirectly*, taking the document count as a metric to split the network into snapshots also discretizes time. Respective time windows are defined so that each network contains the same number of documents. An unequal temporal distribution of documents inevitably leads to unequal time intervals, which sets this approach apart from the one using fixed window sizes. Furthermore, in an accumulative network setting, volume-based sampling, as described above, is not possible, at least if more documents are added to the dataset over time, and the removal of documents is not allowed. In such a scenario, $|D^i| <= |D^{i+1}|$ always holds. Nevertheless, the volume-based sampling has a natural extension to fit the accumulative setting. For this, the number of newly added documents per snapshot is kept constant: $|D^{i+1}| - |D^i| = |D^{i+3}| - |D^{i+2}|$. The CODY model presented in Section 5.3 employs such a sampling strategy.

- **Use case-specific:** Depending on the context, there might also be more appropriate use case-specific strategies to partition the temporal network data into snapshots that are neither based on fixed time windows nor the volume of analyzed documents. For example, in the work about actor-networks underlying trends examined in Section 4.2 the temporal sequence of detected trends is leveraged to define the time windows of the network snapshots. Given that trends might occur irregularly, the time windows of the snapshots might also be of different sizes and cover a variable number of documents. In contrast to the "fixed time window" method, these approaches also *indirectly* discretize time.

Remark. The present model does not consider links between documents of different snapshots. For every link $d_k \xrightarrow{l_d} d_l$ the linked documents d_k and d_l have to be part of the same snapshot: $d_k \in D^i \wedge d_l \in D^j$ with $i = j$. Other document links are not considered. Therefore, it is crucial to select an appropriate snapshot strategy carefully. Not too much information relevant to the analysis use case should be contained in the links between document snapshots. Instead, the most relevant information should be contained in the links within the temporal snapshots. As an example, if one is interested in the interactions among Twitter users and the main interactions are happening during weekdays, the time windows should be defined to go from Monday to Sunday and not from one Wednesday to the other, as this approach would omit valuable links between tweets that are posted on the first days of the week and those that are posted on the other days. Generally speaking, the snapshots should be defined so that links between these snapshots do not

play a significant role in the analysis use case. In this way, it is guaranteed that through discretizing the network, as little information as possible is lost.

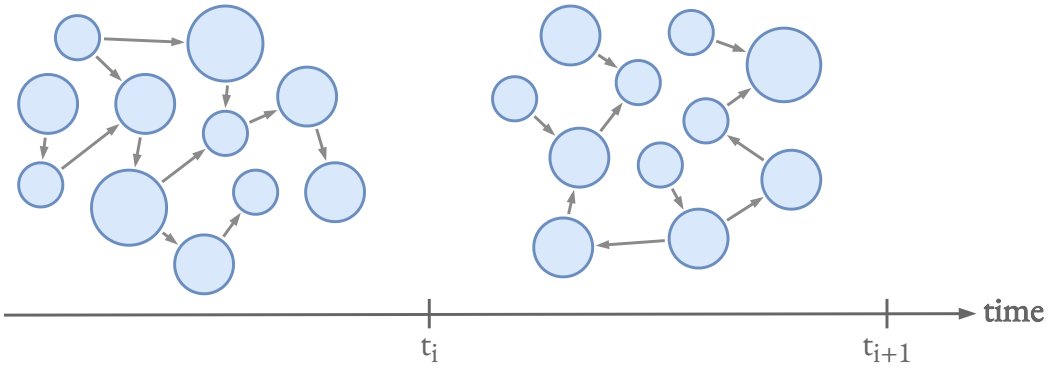


Figure 3.5: To model the dynamics of the network, it is split into multiple temporal snapshots.

Figure 3.5 illustrates the splitting of a document network into snapshots according to given time windows. The snapshot-based modeling approach allows observing temporal changes in the network structure. Not only temporal changes related to the documents but also related to extracted entities might be observed. Even though the network snapshots are constructed based on the timestamps of the document nodes, the entities extracted from these documents, as outlined in Section 3.2.2, are also part of the snapshot networks. Accordingly, Figure 3.6 visualizes the temporal document snapshots with the extracted entities as part of the networks. These networks are referred to as *entity network snapshots*. Each entity network snapshot comprises the document network's nodes and relationships and the entities and links extracted from these documents.

Definition 3.5 (Entity network snapshot)**.** An entity network snapshot of time window w_i is defined as $G_e^i = (\{D^i \cup E^i\}, \{L_d^i \cup L_e^i\})$ with E^i and L_e^i referring to the entities extracted from the documents D^i and the belonging links respectively.

With the entities as part of the temporal network snapshots, analyses based on certain entity types can also consider their temporal changes. For example, one might be interested in the entities that are added or removed from one snapshot to the other or how the occurrence of an entity changes over time.

3.3.2 Temporal vs. evolution analysis

During analysis, temporal changes can be investigated from different perspectives. As described in Section 3.3.1, the analyzed data is split into temporal snapshots. Each snapshot represents an excerpt of the ongoing evolution of the constructed network. Therefore, the first perspective

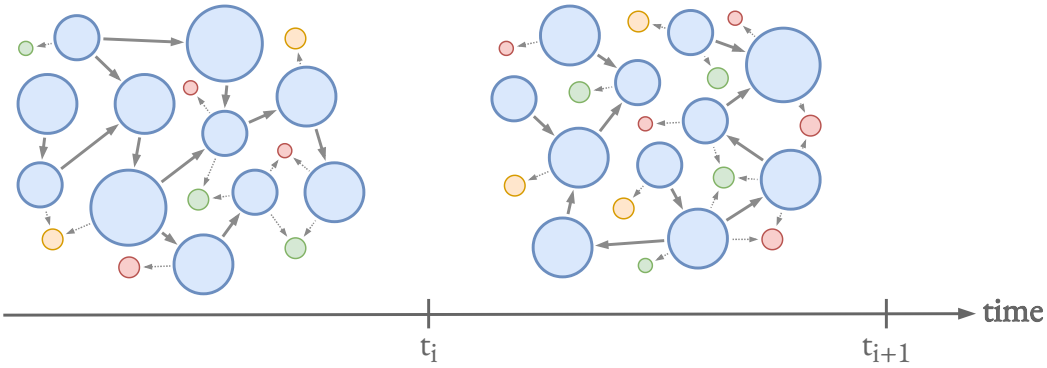


Figure 3.6: Document network snapshots with extracted entities

on the temporal evolution of the network is centered around single snapshots. Questions asked with this perspective in mind focus on the state of the network at this point or target temporal properties of individual nodes, edges, or subgraphs. This data analysis method is referred to as *temporal analysis*.

In contrast to focusing on a single snapshot, one might also be interested in the *evolution* of a particular property or metric across time. Such a property or metric might be the degree of a specific node or its centrality, the number of documents included in the snapshot network or the number of interactions among these. An evolution-based analysis starts by deriving the property values for each network snapshot. This results in a temporally ordered list or time series of values and allows investigating the temporal evolution of the property across the entire period covered by the network snapshots. Formally, if one refers to the property calculated for the i-th snapshot as x_i, mentioned time series of temporal property values is given as $X = (x_1, x_2, \dots, x_i, \dots, x_n)$ with n being the number of network snapshots. This formalism assumes that the property value can be derived for each network snapshot. If not, the corresponding null value is used as a property value for respective snapshots.

Throughout this thesis, both analysis approaches are applied. For example, the analyses conducted in the context of long-term social media trends detailed in Section 4.1.5 illustrate the difference between *temporal* and *evolution* analysis. While comparing the state of the trend networks at different points in time corresponds to a *temporal* analysis (see Figure 4.4), does the investigation of how the trend prevalence change over time fit with the characteristics of an *evolution* analysis (see Figure 4.5).

3.3.3 Related work

Different approaches for modeling the temporal evolution of networks exist, and the terminology used varies a lot. For more details on this topic, please see Section 2.1.5. Nevertheless, commonly, the modeled temporal information is attributed to the networks' edges. This attribution, however, contrasts the model presented in Section 3.3.1. In line with the data structure of document corpora, timestamps are modeled as document node attributes. Furthermore, the differentiation between temporal and evolution analysis proposed in Section 3.3.2 and previously outlined in Section 2.1.5 aligns with the understanding applied by Rost et al. (2023) who also differentiate between temporal and evolutionary metrics. According to them, the first mentioned metrics are defined with regard to a specific moment, and the latter ones capture the change within a given period in the form of a time series.

In contrast to snapshot-based approaches, which aggregate the temporal data into static networks according to pre-defined time windows, Kostakos (2009) presents a different approach. Each node becomes a directed chain of temporal node occurrences in his temporal network model. For each time a node is part of an edge, a respective node instance is added to the node's temporal occurrence chain. Links between the node chains represent the actual temporal edges. This model does not rely on snapshots, so it differs from the model presented in Section 3.3.1.

Several publications provide a good overview of the temporal network analysis topic, such as the one by Casteigts et al. (2012) or the survey by Wang et al. (2019). Also, more use case-specific works exist. For example, Spitz and Gertz (2018) leverage temporal projections, i.e., snapshots, of term and named entity co-occurrence networks to study the evolution of topics in document corpora. This approach is similar to the one presented in Section 3.3.1. Still, our approach specifically includes different network sampling strategies.

Multi-slice networks

An alternative approach to incorporate the temporal information into the network model presented in Section 3.2 would be multi-slice temporal snapshot networks. Therefore, these should also be discussed in the context of related work. To apply this approach, one would follow the formalism outlined in Section 2.1.4 but not consider replica nodes. Instead, a document node would be part of only the network layer that covers the time window in which the document's timestamp falls. With a given time interval Δt to define the network snapshots, the nodes of layer i would be given as $V^i = \{d \mid d \in D \wedge (i-1)\Delta t \leq \pi_{timestamp}(d) < i\Delta t\}$. This subset of documents would then be referred to as $D^i \subseteq D$. Further, links between two documents d_k and d_l would be intralinks in the case they connect documents with timestamps that fall into the same time win-

dow, $d_k \in D^i \wedge d_l \in D^j \wedge i = j$, or they would be interlinks in the case they connect documents of different time windows, $d_k \in D^i \wedge d_l \in D^j \wedge i \neq j$. However, such interlinks are not modeled by the document network snapshots described in Section 3.3.1. Instead, multi-slice networks with only intralinks would be more suitable for this model. Also, it should be noted that the multi-slice network approach relies on fixed time window sizes. In contrast, the model presented in Section 3.3.1 does not restrict the time windows used to define the snapshots. Also, the multi-slice network approach does not cover the accumulative creation of network snapshots.

Temporal HINs

As stated in Section 3.2.1, the network containing only document nodes can already be seen as HIN. It might consist of different document node types, such as tweets and news articles, but also different document relationship types, such as "retweet" or "reference". The network becomes even more diverse by including the entities extracted from the documents. At this step, additional node types like "URL" or "hashtag" might be added to the network. Now, by modeling the network as temporally evolving, it can be described as *temporal* HIN model. Various works have already dealt with such temporal information networks. For example, in their work Milani Fard et al. (2019) apply the task of meta path prediction to dynamic HINs. They understand dynamic HINs as typed networks with timestamped edges and model the network's evolution as a sequence of temporal snapshots, in line with the approach presented in Section 3.3.1. Going one step further, Sajadmanesh et al. (2019) not only try to predict relationships in the setting of temporal HINs but also propose a method to predict *when* these relationships will appear in the future. Focused on community detection, Sun et al. (2010) propose a method to detect heterogeneous communities in dynamic HINs. They also leverage sequences of temporally-aggregated network snapshots to analyze the temporal evolution of the studied networks. Using a similar methodology, Cuzzocrea and Folino (2013) focus on tracking communities over time and develop a method to detect changes in the community structure of the analyzed information network. Several other aspects are studied in the context of dynamic HINs, such as deriving node embeddings (Wang et al., 2022; Bian et al., 2019), sequential recommendation (Xie et al., 2021), network motifs (Li et al., 2018), or link inference (Jia et al., 2017; Aggarwal et al., 2012) and prediction (Sett et al., 2018).

3.4 Projections

For some analysis use cases, not all information contained in the networks presented in Sections 3.2 and 3.3 might be needed. Sometimes, the heterogeneous nature of the networks might hinder

or make the corresponding analysis more complex. Therefore, an appropriate method to translate the original, full-blown network into a more condensed version, which only contains the information needed for the given analysis use case, is needed. Network *projections* as presented in this section can be understood as such a "translation" approach. In this regard, Section 3.4.1 outlines the projection mechanism concerning the entity networks described in Section 3.2.2. Section 3.4.2 describes how node attributes are handled during the network projection process. Finally, Section 3.4.3 summarises related work.

3.4.1 Entity network projections

The model's current state does not incorporate links between entities. This lack of entity relationships is true for the entity network containing all documents and entity network snapshots. Nevertheless, much valuable information is contained in these entity-entity relationships. Some interesting questions to ask based on these relationships could be: What hashtags are frequently used within the same tweet? Which Twitter users are mentioned in the same context? How often are two websites referenced together? Even though the entity network model presented in Section 3.2.2 does not explicitly model links between the entities extracted from a document, these are still present implicitly. To manifest these latent relationships, the respective entity network has to be projected onto an entity network with these latent relationships explicitly modeled as links. These projected networks are referred to as *entity network projections*.

Formally, such a network projection $G_p = (E_p, L_p)$ is made up of the links L_p between a subset of those entities that are contained in the unprojected network: $E_p \subseteq E$. The subset of entities and respective links depends on the way the projection of the network is conducted. Generally, the projection is defined by a meta path, which starts and ends at an entity type. In other terms, with a given meta path of length n for each of its path instances, $p = (v_1, v_2, \dots, v_{n+1})$, $v_1 \in E$ and $v_{n+1} \in E$ should hold. Nevertheless, at least one node of the sequence $(v_i)_{i=2}^{n}$ has to be of type document as entities are not directly linked in the original entity network. In the simplest form, two entities are linked if they are extracted from the same document. In this case, the corresponding meta path instance is given as $e_k \rightarrow d \rightarrow e_l$. Given this meta path instance, the resulting entity network projection would contain a link between e_k and e_l: $(e_k, e_l) \in L_p$ and given the above meta path instance p the according projection would contain a link (v_1, v_{n+1}). A more general description of the entity network projections is given in Definition 3.6.

Definition 3.6 (Entity network projection)**.** If $\mathbb{P}$ is the set of all meta path instances that follow the meta path used to project the network and this meta path is of length n with $n \geq 3$, then the nodes E_p contained in the projected entity network are defined as $\bigcup_{p \in \mathbb{P}} \{v_1, v_{n+1}\}$ with $p =$

$(v_i)_{i=1}^{n+1}$. Further, the set of edges L_p contained in the projected entity network are defined as $\bigcup_{p \in \mathbb{P}} \{(v_1, v_{n+1})\}$.

By now, the edges of the projected network are not typed. To improve the semantic expressiveness of the links, they might be labeled. Finding relationship labels requires domain-specific knowledge, as the labels should describe the semantic meaning of the projected links. Table 3.4 gives some exemplary meta paths and respective labels.

Table 3.4: Examples of meta paths used to construct entity network projections

Label	**Meta path**
hashtag co-occurrence	$hashtag \xrightarrow{used\ in} tweet \xrightarrow{uses} hashtag$
co-mention	$username \xrightarrow{mentioned\ in} tweet \xrightarrow{mentions} username$
residence of hashtag user	$hashtag \xrightarrow{used\ in} tweet \xrightarrow{posted\ by} user\ profile \xrightarrow{residence} country$
same hashtag usage	$username \xrightarrow{belongs\ to} user\ profile \xrightarrow{posts} tweet_1 \xrightarrow{uses} hashtag \xrightarrow{used\ in} tweet_2 \xrightarrow{posted\ by} user\ profile \xrightarrow{belonging} username$

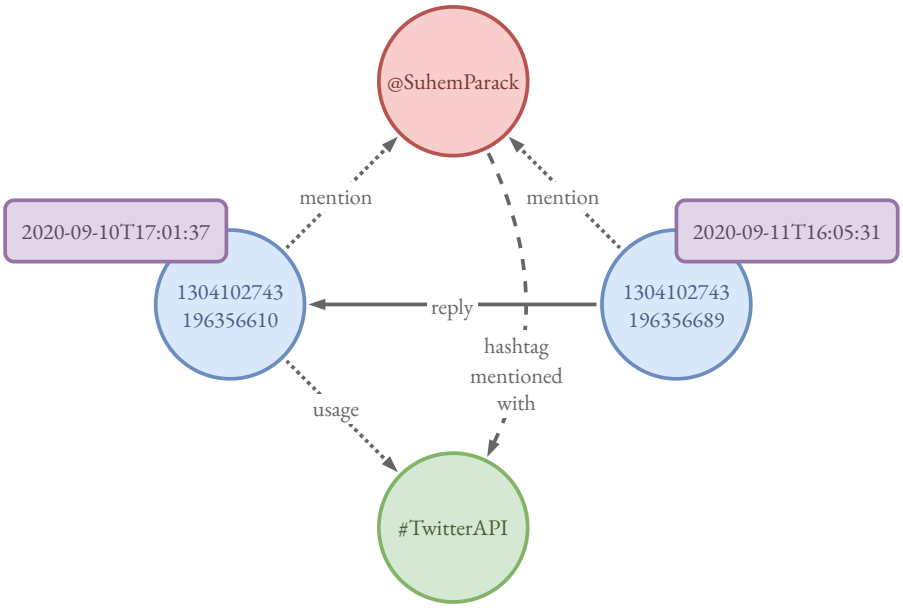


Figure 3.7: Twitter example entity network with added instance of $username \xrightarrow{mentioned\ in} tweet \xrightarrow{uses} hashtag$ meta path indicated by the dashed line

Figure 3.7 illustrates the extraction of *latent* relationships from entity networks. It shows the same entity network as Figure 3.4 but with the latent relationship between the extracted Twitter user

and hashtag explicitly represented as "hashtag mentioned with" relationship. This relationship follows the meta path *username* $\xrightarrow{\textit{mentioned in}}$ *tweet* $\xrightarrow{\textit{uses}}$ *hashtag* and is indicated by the dashed line.

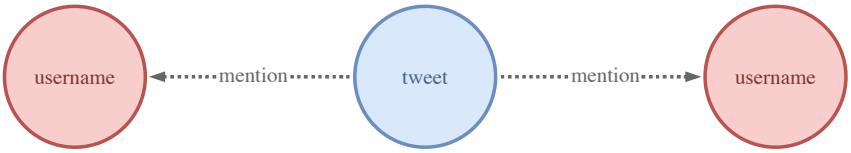
Taking the "hashtag mentioned with" meta path as indicated in Figure 3.7 to project the according entity network, one receives a simple entity network projection with the respective Twitter user and hashtag as nodes and a connecting "hashtag mentioned with" link. The described network projection is shown in Figure 3.8.

Figure 3.8: Entity network projection by meta path as shown in Figure 3.7

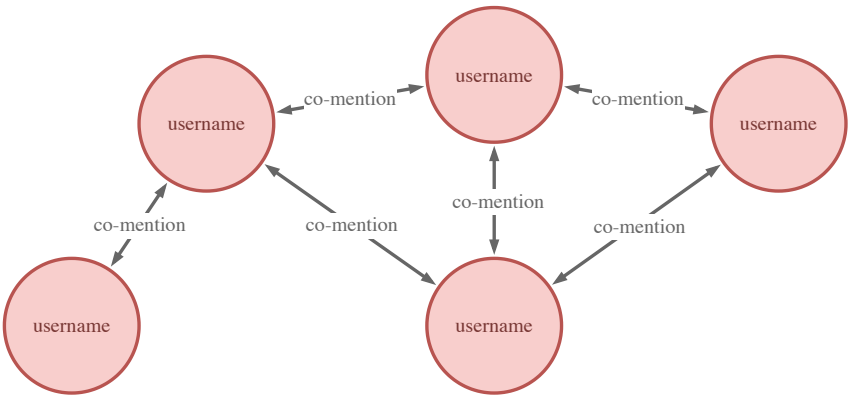
Remark. Similar to the discussion about directionality implied by the relationship labels outlined in Section 3.2.2, one might argue that some meta paths can also be formulated both ways. The link between the Twitter user and hashtag, as shown in Figure 3.7, might also go from the hashtag to the Twitter user and be labeled as "Twitter user mentioned with". Together, both meta path projections could be summarized as "Twitter user and hashtag co-occurrence" projection and be modeled as an undirected network with links between hashtags and Twitter users that occur within the same tweet. For the sake of simplicity, the model sticks with simple projections based on single meta paths and does not deal with the aggregation of multiple network projections. Still, these might be possible in an additional aggregation step.

Examples of entity network projections

Entity network projections can be used for different use cases and in various analytics scenarios. Accordingly, one has to decide which entity types and meta paths should be considered during the network projection. Two highly relevant use cases are analyzing topics and actors, especially while working with social media data. For these use cases, one has to target entities representing semantic entities, like hashtags, named entities, or terms, and actors (e.g., Twitter users or news article authors), respectively. In the according entity network projections, these entities might be linked by relationships such as co-occurrence for the topic-centered projections and co-mentioning or co-authorship for the actor-focused projections. Figure 3.9 illustrates such an actor-focused entity network projection based on Twitter user co-mentioning.



(a) Schema of the exemplary document network used for the actor-network projection



(b) In the exemplary projected actor co-mentioning network, the username nodes are linked by co-mention relationships.

Figure 3.9: Illustration of an entity network projection by the example of username co-mentioning. The co-mention meta path (see Table 3.4) is leverage for the projection.

Entity network projections are also leveraged for several studies presented in this thesis, such as to examine actor-networks underlying trends (see Section 4.2) or to analyze long-term trends (see Section 4.1).

3.4.2 Network attribution

Similar to document node attributes described in Section 3.2.3, the projected entity networks might also be extended by additional information modeled as attributes. Thereby, the attributes come from the original document nodes in the projected document network and might be assigned to nodes or edges. In line with the formalism outlined in Section 3.2.1, these attributes are defined via attribute mappings that either map entity nodes, $\pi_{node\ attribute}(e_p)$ with $e_p \in E_p$, or respective edges, $\pi_{edge\ attribute}(l_p)$ with $l_p \in L_p$, to their attribute values. Again, these mappings are partial functions and do not assign an attribute to nodes or edges for which it is not defined. Table 3.5 gives some examples of attributes that might be present in a projected entity network. The column "projected object" indicates whether the attribute is relevant for network nodes or edges in the projected network.

Table 3.5: Examples of attributes that might be used in entity network projections

Description	Projected object
number of Twitter follower	node (esp. Twitter user)
timestamp	edge
impact score	node (e.g., news outlet)
similarity	edge
weight	node/edge

During the network projection, attributes formerly assigned to document nodes might be assigned to nodes or edges in the projected network. The formalism of this assignment process is described by Definition 3.7.

Definition 3.7 (Document-based network attribution)**.** If a meta path instance $p = (v_i)_{i=1}^{n+1}$ traverses document d such that $d \in \bigcup_{i=2}^{n} \{v_i\}$, then the document's attributes might be used as node or edge attributes in the corresponding subgraph of the projected network: $\pi_{\textit{document attribute}}(d) = \pi_{\textit{node attribute}}(v)$ with $v \in \{v_1, v_{n+1}\}$ or $\pi_{\textit{document attribute}}(d) = \pi_{\textit{edge attribute}}(l_p)$ with $l_p = (v_1, v_{n+1})$.

To highlight, the projection of document attributes is essential for temporal information related to the document nodes. With that information also given in the projected networks, these networks can be analyzed from a temporal-sensitive point of view. Questions like "When did these two hashtags occur most frequently together?", and "Which Twitter users were frequently mentioned together during this period?" can be answered this way. Recalling the model requirements stated in Section 3.1, the entity network projections improve the information value of the modeled data. Simple filtering and aggregation-based queries on the projected networks allow us to answer questions like those above. This way, meaningful *information* can be derived from the projections instead of unprocessed and less informative data.

3.4.3 Related work

Network projections are well-known from bipartite networks (Newman, 2010, pp. 123–126). This kind of network contains two types of nodes, and each edge in the network links nodes of different types. In the so-called "one-mode projections", only the nodes of one type are contained. These nodes are linked if they are connected in the original bipartite network via a shared adjacent node of the other type. Such a one-mode projection can be seen as a particular case of the more general meta path-based projections outlined in Section 3.4.1. By discarding links between documents, the entity networks can be seen as bipartite networks with the two node types "docu-

ment", and "entity". Accordingly, entity network projections defined by meta paths that follow an *entity* → *document* → *entity* structure can be seen as one-mode projections of described bipartite document-entity networks. Still, in our case, the projected entities are not required to be of the same type.

More generally, projections of bipartite networks are leveraged in different network analysis scenarios and use cases, such as personal recommendation (Zhou et al., 2007), community detection (Melamed, 2014), and food pairing (Ahn et al., 2011). Further, some work also deals with network projections of HINs. For example, Shi et al. (2014) leverage HIN projections for their ranking-based clustering method. According to their definition of a HIN projection, one starts by selecting a node type, called "pivotal type", and other node types, called "supportive types", linked to the node type selected at first. With these selections, the projected network's schema is defined. It contains all nodes of selected types and links among these. As outlined by Shi et al. (2014), the projected networks are either bipartite or star-schema networks with potential self-loops. This definition of a heterogeneous network projection differs from the approach outlined in Section 3.4.1. The latter relies on meta paths as a basis to define the projection. In contrast, the approach by Shi et al. (2014) is centered around a single node type and its neighborhood. Accordingly, the schema of the projected network can be seen as a subgraph of the original network's schema. A more similar approach to the one presented in Section 3.4.1 is used by Grčar et al. (2013). They propose a framework to analyze document-enriched HINs and leverage network projections. In these projections, two nodes are linked if they share a common neighbor of a particular type in the original HIN. This approach is similar to the case of a projection that follows an *entity* → *document* → *entity* meta path. Further, as outlined in Section 2.1.3, Milani Fard et al. (2019) define a general method for projecting HINs based on meta-paths. Even though they do not build on the document-entity network schema as we do, their methodology is similar to our entity network projection if one defines an "augmented reduced graph" (Milani Fard et al., 2019) based on a meta-path that starts and ends with an entity type.

Focused on HIN embeddings, some existing work also uses network projections in a different sense. The network is represented in a different vector space depending on the relationship type. For that, the vector of each node of the original HIN is *projected* to the relationship-defined vector space. Works that follow this approach are, for example, Chairatanakul et al. (2021) and Chen et al. (2018).

3.5 Granularity

The networks described in Sections 3.2, 3.3 and 3.4 can be investigated from different perspectives and according to different levels, scales, or *granularities*. The following section elaborates on these granularities concerning network topology in Section 3.5.1 and regarding the temporal evolution of a network in Section 3.5.2. Finally, Section 3.5.3 covers related work of presented concepts.

3.5.1 Network granularities

So far, the proposed model focuses on topological patterns related to single nodes or relationships. Nevertheless, more coarse-grained structures like subgraphs of the constructed networks also provide valuable information from more than just the contained nodes and relationships. This section provides a more systematic overview of the different levels of granularity that might be used to analyze described information networks. Similar to zooming in and out while observing an image, a network can also be analyzed on different levels of detail. These levels of detail are referred to as *granularities*. A fine-granular perspective focuses on individual nodes and edges, whereas a less granular focus shifts towards a more coarse network perspective. Table 3.6 provides an overview of which granularities can be used when analyzing a network. According to their level of zoom, these levels are referred to as *micro*, *meso*, *macro* and *mega* perspectives.

Table 3.6: Different levels of granularity that can be used as perspectives on the analyzed network

Granularity	**Network perspective**
micro	individual node or edge
meso	two nodes and their relationship
macro	subgraph
mega	complete network

In the following, the described levels of granularity are examined in more detail:

- **Micro:** At this level of granularity, the analysis focus is placed on individual nodes and edges. Regarding the edges, this perspective does not involve the connected nodes but solely focuses on the information related to the edges themselves. Most importantly, one is interested in the attributes and metrics of given nodes or edges. These properties might include the node or edge type, a node's degree or the weight of an edge. For example, one might be interested in the number of hashtag occurrences by checking the hashtag node's degree in the Twitter network as shown by Figure 3.4.

- **Meso:** The meso level refers to a network edge *including* its involved nodes. In other studies, e.g., by Saveski et al. (2021), this network structure is also called a "dyad". Use cases for this perspective are the analysis of how often a particular link occurs in a network across time. Also, one might be interested in comparing adjacent nodes, e.g., by investigating their similarity. For example, one might want to investigate the similarity of frequently co-mentioned Twitter users (see Figure 3.9b).

- **Macro:** In the most general sense, the macro perspective investigates network subgraphs. These subgraphs consist of a subset of the nodes and edges of the complete network. Therefore, more topologically complex network structures can be analyzed. Such structures include network communities or motifs. For example, in the context of Twitter actor-networks, one might be interested in analyzing groups of Twitter users that make up communities.

- **Mega:** The most coarse-grained perspective on the network is to investigate it as a whole. Every node and edge contained in the network is considered at this level. One might be interested in global network metrics such as the number of nodes and edges or the average node degree. For example, when analyzing networks extracted from social media data (e.g., Tweets), one might want to investigate the distribution of node degrees to see whether highly connected "hubs" exist.

Complementary to the list of granularities as shown in Table 3.6, Figure 3.10 visualizes the different levels of detail that might be used during the analysis of a network. Notably, the presented systematic is valid for different types of networks like the document networks presented in Section 3.2.1, the entity network projections as outlined in Section 3.4 or even temporal network snapshots (see Section 3.3.1). In all cases, the network can be examined with different levels of detail.

3.5.2 Temporal resolution

Different granularity levels can be defined not only concerning topological network structures but also related to the temporal-sensitive analysis. In the context of time, these levels of granularity are better referred to as *resolution* levels and correspond to varying approaches of discretizing $\mathbb{T}$. Accordingly, the temporal network snapshots outlined in Section 3.3.1 can be created with respect to different resolution levels. Most importantly, this is relevant for snapshots defined by a fixed time window size. Snapshots can be generated based on different time windows, such as 15 minutes, 24 hours, or three months. Different time windows come with different resolution levels. The larger the time window, the lower the time resolution. Also, different resolution levels

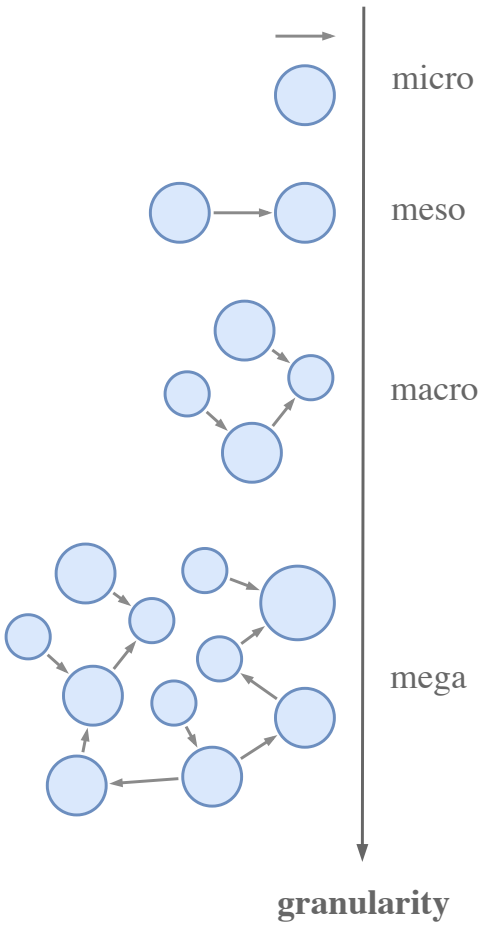


Figure 3.10: Visualization of the different network granularities

must be used depending on the phenomenon being studied, and their determination requires domain-specific knowledge. As outlined by Zhou et al. (2020), different resolution levels come with computational trade-offs. On the one hand, a higher resolution leads to more snapshots and, therefore, higher computational costs. On the other hand, a too coarse temporal granularity might save computational costs but might also lead to a loss of "[...] fine-grained temporal context information [...]" (Zhou et al., 2020).

3.5.3 Related work

In their survey about temporal graph concepts, Casteigts et al. (2012) also differentiate between various "point of views" when analyzing the temporal evolution of a network. They outline three categories: "edge-centric", "vertex-centric" and "graph-centric". Accordingly, the analysis focuses

on individual edges, vertices, or the network as a whole. Similar to the concepts outlined in Section 3.5.1, their categorization is based on different network granularities. Further, they also apply this kind of distinction to the temporal dimension and differentiate between "fine-grain dynamics" and "coarse-grain dynamics" depending on the time scale used for the analysis. Again, this is similar to the temporal resolution concept outlined in Section 3.5.2. Similarly, Roth and Cointet (2010) in their work differentiate between micro- and macroscopic network perspectives. In line with the categorization proposed in Section 3.5.1, the microscopic scale focuses on individual nodes. In contrast, the macroscopic scale, as defined by Roth and Cointet (2010), considers the network as a whole. This perspective corresponds to the "mega" granularity described above (see Table 3.6). Furthermore, Latora et al. (2017, p. 332) refer to communities in networks as "mesoscopic structures". Following our categorization, communities would be macroscopic structures. Farther fetched, the differentiation between micro-, meso- and macro-level analysis is also commonly used in the social sciences. See Serpa and Ferreira (2019) for a discussion on that. According to the specific level, analyses range from the study of face-to-face interaction to the study of society as a whole or even inter-societal systems (Turner, 2010, pp. 12–20).

3.6 Discussion

In this chapter, the media analysis model leveraged for the studies presented in subsequent parts is developed. It is built on concepts known from temporal network analysis and tailored to the use case of media analytics. A summary of its properties and capabilities is given in the following Section 3.6.1. Furthermore, even though the model is successfully leveraged in subsequent studies, its limitations must also be mentioned. Therefore, the model's shortcomings are discussed in the final Section 3.6.2.

3.6.1 Summary

The media analytics model proposed in this chapter is developed with regard to several requirements outlined in Section 3.1. Among these requirements, integrating and modeling heterogeneous data from various sources plays an outstanding role. It led to leveraging networks as the central modeling approach (see Section 3.2). In line with the implicit structure of a dataset consisting of documents that themselves contain entities, a respective network model is developed. Once the data is processed and a network representation is derived, it can be accessed and analyzed in a more standardized and homogeneous way. Further, a specific emphasis is placed on the model's capabilities to reflect the underlying data's dynamics. Snapshots are leveraged as the temporal network representation for that (see Section 3.3). In practice, defining appropriate time windows

for these snapshots can be challenging and at worst, much information is lost due to the network's discretization. Thus, we discuss different temporal sampling strategies in detail. Only by employing an appropriate sampling technique valuable information can be extracted from the data. In this regard, the proposed model comes with two additional capabilities, respectively properties, that facilitate gaining informative insights into the data. First, the network projections introduced in Section 3.4 allow data investigation from different entity-centric perspectives. Based on given meta-paths, "condensed" versions of the original network can be derived. Further, the examined network granularities (see Section 3.5) serve as a framework to guide the network analysis based on different topological and temporal perspectives. Finally, the model takes the "reproducibility" requirement into account (see Section 3.1). In the network, extracted entities are linked to their original document, which allows to retrace derived results and to investigate where gained information comes from.

3.6.2 Limitations

Even though the proposed data analysis model has been developed with the outlined requirements in mind, it also comes with its limitations. Two such limitations arise from the way the network's dynamics are modeled. First, in the snapshot-based temporal network model, interlinks connecting nodes of successive snapshots are not considered. Based on the assumption that an appropriate sampling technique is leveraged, this should not lead to a significant information loss. Still, our approach might not be sufficient for analyses that require continuous modeling of the network dynamics. Secondly, the proposed model considers the data's temporal information in the form of timestamps and not in the form of intervals. Leveraging intervals during which a respective node or edge is active might be more suitable in certain application scenarios. Next, the model is currently limited with respect to the modes of the analyzed data. Primarily, textual data is considered. Nevertheless, media content might also come in other forms, such as videos or images. Extending the model to follow a multi-model perspective would be a valuable path for future work. Finally, in this chapter, no information is provided on the efficient implementation of the proposed model as demanded by the outlined requirements. Section 6.3 catches up on this by establishing a benchmark on the performance analysis of several temporal graph management and analysis systems.

4 Trends

The attention to specific topics is rapidly changing on (social) media, and trends are integral to these dynamics. Very prominently, this is demonstrated by the ever-changing list of trends on Twitter (Twitter, Inc., 2023b) as shown by Figure 4.1. Thereby, the meaning of the term "trend" should not be confused with its meaning in the context of time series analysis. Trends in that context refer to a characteristic of a time series, whereby a time series shows a trend if its mean is dependent on time (Cryer and Chan, 2008, pp. 27–54). In a social media analysis setting, however, trends come with a semantic meaning and are similar to topics (Asur et al., 2011; Budak et al., 2011). Further, for a topic to become a trend, it needs to "[...] capture the attention of a large audience [...]" (Asur et al., 2011) or, in other words, it needs to gain a certain level of popularity. Thereby trends can be triggered in various ways. On Twitter, trends are often related to news, ongoing events, memes, or commemoratives (Zubiaga et al., 2015).

In contrast to trends driven by news or ongoing events that are prevalent for only a short time, long-term trends keep their prevalence for a longer period, e.g., the news coverage of the COVID-19 pandemic. As such, the phenomenon of long-term trends is studied in Section 4.1. Not only does the study contain an examination of long-term trends extracted from a dataset covering the German political Twittershere between January 2021 and July 2022, but it also proposes a network-based analysis framework that is build upon the model developed in Chapter 3.

Independent of how a trend comes into existence and for how long it is present, another interesting dimension of a trend to be studied are the actor-networks among trend participants. Such a study is conducted and presented in Section 4.2 based on a Twitter dataset that covers the discussion around the UEFA EURO 2020[1] soccer championship. Next to the analysis results, the proposed methodology, again based on the model developed in Chapter 3, is presented in this section. In sum, this study showcases another application of the proposed network analytics model.

[1] Season 2020 | UEFA EURO 2020 | UEFA.com: https://www.uefa.com/uefaeuro/history/seasons/2020 (accessed 2023-05-11)
[2] Trends / Twitter: https://twitter.com/i/trends (accessed 2023-06-19)

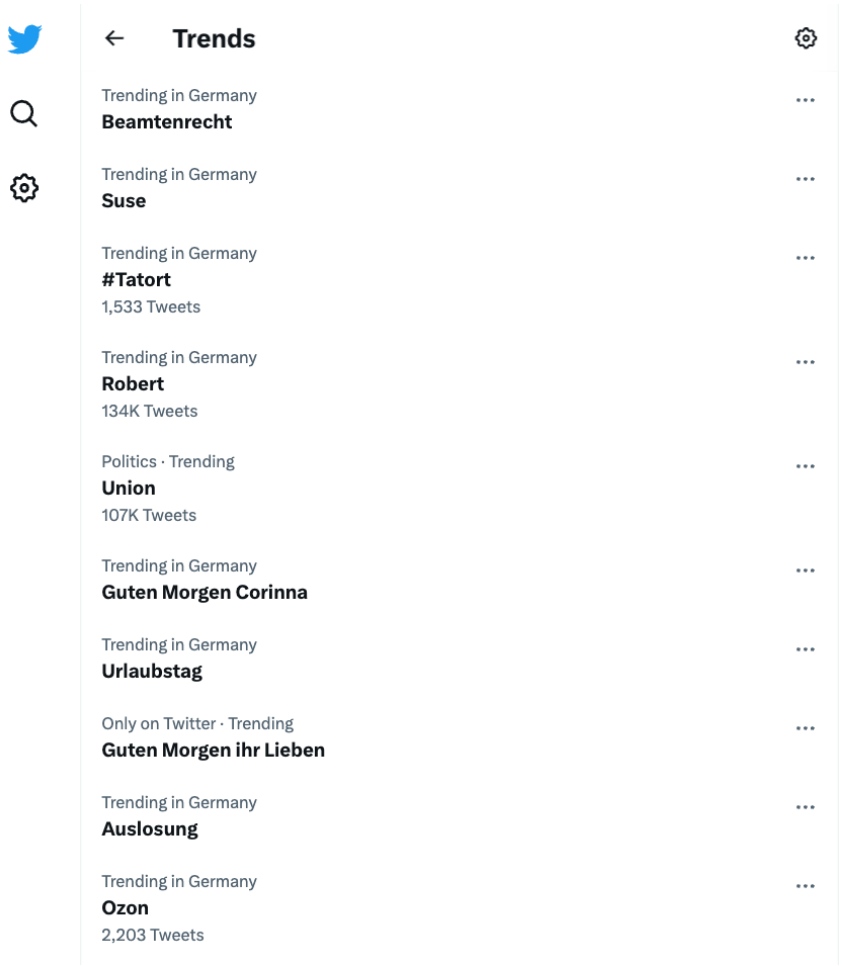


Figure 4.1: Screenshot of a trends list published on the Twitter platform[2]. Displayed trends are also influenced by the user's location, such as Germany, as in the shown example.

4.1 Long-Term Trends

Stories about short-lived breaking news characterize the focus and dynamics of social media. Often, such "mayflies" make it hard to keep track of more profound topics that are prevalent over a long period of time. To provide such capabilities, this section presents a method to detect long-term trends based on temporal networks and community evolution. Connecting these methods with trend analysis approaches allows to study the temporal development of trends, their contextual information, and how they are interrelated over time, thereby overcoming deficiencies of existing work. Results obtained from a Twitter case study are discussed in detail and evaluated based on real-world event linkage, proving the proposed method's effective functionality.

Reference: This section is based mainly on the following peer-reviewed publication:

John Ziegler and Michael Gertz. No Mayfly: Detection and Analysis of Long-term Twitter Trends. In *BTW 2023*, pages 353–364. 2023.

The remaining part of this section is structured as follows: After outlining the contributions of this study in Section 4.1.1 and a coverage of the leveraged terminology in Section 4.1.2, related work is described and compared in Section 4.1.3. Section 4.1.4 then covers the methodology concerning the detection of long-term trends. The proposed method is applied to a collected political Twitter dataset, and the according analysis is described and evaluated in Section 4.1.5. Finally, Section 4.1.6 summarises the present framework and describes future work.

4.1.1 Contributions

In this study, a framework to detect and analyze long-term social media trends is proposed. It builds on existing work that is adopted and extended to fit the use case requirements. These extensions include:

1. Leveraging a temporal network model to study long-term trends,
2. Pruning of less prevalent nodes based on a power law degree distribution model,
3. Temporal tracking of hashtag communities via a core of central nodes, and
4. Visualizations to analyze the temporal development of found trends.

The proposed methodology is applied to the German political Twittersphere to analyze long-term political trends. Thereby, the network-based approach allows us to intuitively represent detected trends within their semantic context. In contrast to related work, e.g., the work by Chae and Park (2018), the present analysis specifically investigates semantic shifts of detected trends over time.

4.1.2 Terminology

Asur et al. (2011) describe trends as topics that "[...] capture the attention of a large audience [...]". We follow this definition and start by taking Twitter hashtags as representatives of topics, which is in line with previous work, e.g., Asur et al. (2011) and Budak et al. (2011). According to Bhulai et al. (2012), these hashtags might also be clustered. As a result, we extend the previous definition of a "topic" and do not refer to it as a single hashtag but as a community of hashtags. Tracking those communities of hashtags over time results in "temporal topics". A topic can be

said to make up a "trend" if its popularity is large enough (see Asur et al. (2011)) and is further called a "long-term" trend if it is prevalent over a sufficiently long time period. Combined, we denote them as "long-term topical trends". Further, to distinguish between short- and long-term trends, we refer to the concept of "news cycles" or rather "political information cycles" as described by Chadwick (2011). These cycles describe news production processes and typically cover a time span of a few days. Topics discussed in the context of such short-lived media attention cycles are defined as short-term trends. In contrast, long-term trends describe topics that are prevalent in media for several weeks, months, or even years. This distinction is in line with past work, e.g., by Hashavit et al. (2016).

4.1.3 Related work

Most studies related to social media trend analysis focus on short-lived and mostly event-driven scenarios, e.g., Asur et al. (2011) and Budak et al. (2011). Nevertheless, Chae and Park (2018), as an example, apply topic detection to a long-term Twitter dataset and investigate trends within the corporate social responsibility domain. They study how the popularity of topics changes over time and how topics are interrelated. In contrast to their work, the present analysis focuses on the political domain, explicitly aiming to analyze topical shifts over time and following a temporal, network-based approach. In another work, Cazabet et al. (2012) use network analysis methods to extract trends from social media content. They apply a dynamic community detection method to detect trends. Their method does not only allow for studying bursting events and short-lived topics but also steady trends that are prevalent over a long period. Regarding the semantic evolution of these trends, they look at the terms that become relevant in respective contexts during different times. The authors leave it for future work to adjust their method to other social media platforms such as Twitter. In our work, trends are extracted from Twitter data and specifically for the German political domain. Also, the semantic evolution of found long-term trends is analyzed in more detail by investigating the complete co-occurrence networks at different points in time and by looking at the interrelation of different topics across time. Also related to trend analysis, Annamoradnejad and Habibi (2019) study the trends published by Twitter itself. Thereby, they analyze the trending time as well as the trend's re-occurrence over time.

Further, Majdabadi et al. (2020) propose a graph-based Twitter trend extraction method and do not only take hashtags but also terms into account. Still, they do not track those trends over long time periods. Similarly, the work by Khan et al. (2021) deals with detecting and ranking trends based on Twitter data. Also, Cui and Kertész (2023) analyze the pre-history of trending hashtags on the Chinese microblogging platform Sina Weibo. They find two dominating categories of popularity progression. On the one hand, there is the "Born in Rome" category for which

superhubs are involved at an early stage, and hashtags immediately gain high popularity. On the other hand, hashtags that follow the "Sleeping Beauty" pattern need more initial time until they reach a certain level of attention. Even though their work is also concerned with the temporal dimension of trends, they do not investigate long-lasting trends.

Some existing work from the field of information retrieval also approaches trend-related use cases. For example, Hashavit et al. (2016) propose an online method to detect trends in a user search context. Notably, they also use social network communities as trend candidates and distinguish between short- and long-term trends. Further, focusing on classifying trends on Twitter, the work by Zubiaga et al. (2015) outlines a classification system of Twitter trends, along with methods to correctly identify a trend's category at its initial stage. For trends, they rely on the official trends shown on the Twitter platform, which are short-living (Twitter, Inc., 2023b).

4.1.4 Methodology

The following paragraphs outline the methodology underlying the detection of long-term trends. For this, we first introduce the leveraged dataset in Section "Dataset", then continue by describing the temporal, network-based model formalism in Section "Temporal networks" and outline the processing of the underlying hashtag co-occurrence networks in Section "Network processing". Finally, Sections "Detection of hashtag communities" and "Long-term trend detection" cover the detection of topics and their tracking over time, which also leads to the extraction of topical long-term trends.

Dataset

The EPINetz Twitter Politicians Dataset 2021 provides "[...] Twitter accounts of German parliamentarians, minsters [*sic*], state secretaries, parties, and ministries on a state, federal, and European Union level for the year 2021" (König et al., 2022). We rely on the Twitter search API v2[3] to gather the raw tweets based on those user accounts. Tweets posted by the 2,449 accounts are collected for the time range from January 2021 until July 2022 without filtering. In total, the dataset contains about 1.8 million tweets. Retweets are not considered. Instead, only the original tweets are taken into account. Collected tweets are primarily in German and cover topics from the political domain. Not every Twitter user we track via the Twitter API is contained in the dataset, most likely due to a lack of Twitter activity in the analyzed time frame. In sum, the dataset contains posts of 2,097 unique users. It is a subset of the complete EPINetz dataset described in Section 6.1.5. Hashtags

[3] Twitter API Documentation | Docs | Twitter Developer Platform: https://developer.twitter.com/en/docs/twitter-api (accessed 2023-05-11)

used in the tweets are taken as representatives of topics, which corresponds to the procedure of other works (see Asur et al. (2011) and Budak et al. (2011)). We extract timestamped information about the (co-)occurrence of the hashtags from the unprocessed tweets and use them as the basis for detecting long-term topical trends. Table 4.1 gives a comprehensive overview of the dataset statistics.

Table 4.1: Rounded statistics of the collected Twitter dataset used to analyze long-term trends in the German political Twittersphere. "Unique hashtags" refers to the number of unique hashtags that occur in at least one of the collected tweets. Similarly, "unique tweet authors" refers to the number of unique Twitter users that published a tweet contained in the dataset.

Description	**Count** (in thousand)
tweets	1,824
unique tweet authors	2.097
unique hashtags	144.039
temporal hashtag co-occurrences	5,942

The timestamped hashtag co-occurrence information can also be described in the context of the model proposed in Chapter 3. Specifically, the hashtag co-occurrence network represents an entity network projection as outlined in Section 3.4.1. The according network projection is based on the meta path $hashtag \xrightarrow{used\ in} tweet \xrightarrow{uses} hashtag$ (see Table 3.4). Further, the timestamp of the edge is the result of a network attribution as described in Section 3.4.2.

Temporal networks

By leveraging the timestamped information about hashtag (co-)occurrences, the temporal networks are created as aggregations based on a given time window. To formally describe the temporal snapshot networks, we rely on the framework of multi-slice networks as outlined by Bianconi (2018, pp. 106–110). A multi-slice temporal network is a special kind of multilayer network, with each layer/slice representing a temporal snapshot of the complete network (see Section 2.1.4). As in our case, no interactions across snapshots exist, we focus on the intralink networks only, i.e., multi-slice networks without interlinks. Such a multilayer network M is defined as a tuple, $M = (\mathscr{M}, \dot{\mathcal{G}})$. It consists of the network layers $\mathscr{M}$ with $|\mathscr{M}| = n_{\mathscr{M}}$. A single layer is referred to as $m \in \mathscr{M}$. Additionally, $\dot{\mathcal{G}}$ describes the time-ordered list of networks that are made up of the interactions within each of the layers:

$$\dot{\mathcal{G}} = (G_1, G_2, \dots, G_m, \dots, G_{n_{\mathscr{M}}}) \quad \text{with} \quad G_m = (V_m, L_m). \tag{4.1}$$

Each network G_m consists of a set of nodes V_m, which are, in our case, hashtags and their co-occurrences as edges L_m. Given that a multi-slice network M covers the interactions within a time period T and the time interval Δt is chosen as snapshot size, e.g., one month, there are $n_M = \frac{T}{\Delta t}$ layers. Thereby, layer m captures the interactions that occur in the time window $[(m-1)\Delta t, m\Delta t)$. Within such a layer m, the degree of a node i is denoted as k_i^m. Further, for the *aggregated* network $\tilde{G}$ of the multi-slice network, the temporal nature of the interactions is neglected, and edges from all snapshots are taken into account.

Model reference. The proposed temporal network model based on multi-slice networks aligns with the dynamic network model described in Section 3.3, which is based on temporal document network snapshots. We refer to the discussion in Section 3.3.3 for a detailed comparison.

Network processing

In contrast to mostly event- or breaking news-related short-term trends (Zubiaga et al., 2015), which are often represented by a single hashtag, long-term trends deal with more complex topics and can therefore be seen as communities of interrelated hashtags. To obtain more meaningful community networks and to further save computational costs during the community detection step, this analysis focuses on popular and highly connected hashtags. For this, less related hashtags, i.e., with a low co-occurrence degree, are removed from the temporal networks. We take the median node degree per snapshot as a reference and remove all hashtags with a degree below this threshold from the according temporal network. An investigation of the degree distribution reveals its power law nature ($k \propto k^{-\alpha}$). Therefore, we leverage the median as defined by Newman (2005):

$$k_{med} = 2^{1/\alpha - 1} k_{min} \tag{4.2}$$

Figure 4.2 shows an exemplary degree distribution. The fitting procedure reveals a power law exponent of 1.42 and, according to that, a median k_{med} of 5.31. For the fitting, the `powerlaw` package version 1.5 provided by Alstott et al. (2014) is used. In addition to the pruning step, the temporal snapshot networks are weighted. Ideally, respective edge weights reflect the strength of interrelations between hashtags. Therefore, we resort to Pointwise Mutual Information (PMI) (Role and Nadif, 2011) to be used for the edge weights as it sets the co-occurrence frequency of the two adjacent hashtags and their individual occurrence frequency into relation. With f_i^m describing the frequency of occurrence of node i during the timeframe covered by layer m and f_{ij}^m the frequency of co-occurrence of nodes i and j, the according PMI value is defined as:

$$\mathrm{PMI}_{ij}^{m} = \ln \frac{f_{ij}^{m}}{f_{i}^{m} \cdot f_{j}^{m}} = \mathrm{w}_{ij}^{m}. \tag{4.3}$$

As already indicated, those PMI values are used as co-occurrence edge weights w_{ij}^{m} between hashtag i and j in layer m of the temporal multi-slice network. This network weighting is also part of the model proposed in Chapter 3. Specifically, it relates to the network attribution as outlined in Section 3.4.2. Even though the weights are not derived from document attributes directly, they are still derived based on statistics extracted from the documents used for the network construction.

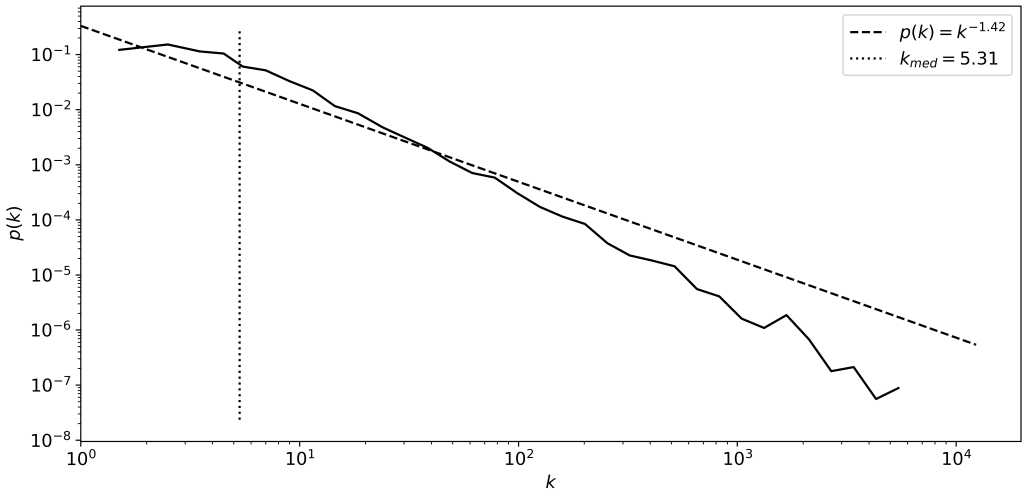


Figure 4.2: Degree distribution of the January 2021 network snapshot

Detection of hashtag communities

Hashtags, i.e., the nodes of the temporal networks, are taken as representatives of topics (Asur et al., 2011; Budak et al., 2011). Further, according to Bhulai et al. (2012), related topics should be clustered in a comprehensive trend analysis framework. Therefore, methods developed in the field of community detection are leveraged to find groups of densely interrelated hashtags. Those groups of hashtags then form a topic with all of its aspects, as multiple hashtags might describe different semantic dimensions of a topic. To be precise, we leverage the Leiden community detection algorithm by Traag et al. (2019) and use the implementation as provided by the `python-igraph` software package version 0.10.1 (Csárdi and Nepusz, 2006). The community detection is applied to all layers of the temporal network described in Section 4.1.4.

Long-term trend detection

Of course, temporal communities of hashtags, i.e., temporal topics, as described in Section 4.1.4 do not yet make up a long-term topical trend. Asur et al. (2011) describe trends as topics that "[...] capture the attention of a large audience [...]", which means that trends need to reach a certain level of popularity. For this to measure, we take the accumulated count of hashtag occurrences per community and time window as trend scores. A community i in the network layer/slice m is given as a subset of hashtag nodes: $\mathbb{C}_i^m \subseteq V_m$. Together with a mapping of those nodes to their respective occurrence counts for the given layer m, $o_m : V_m \to \mathbb{N}$, we define the trend scores τ as follows:

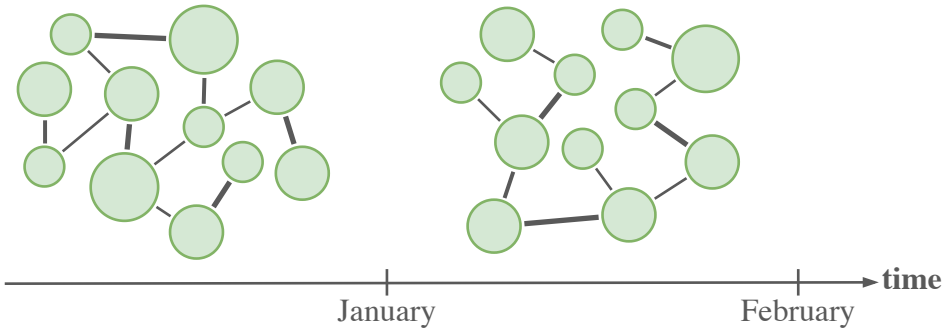
$$\tau(\mathbb{C}_i^m) = \sum_{v \in \mathbb{C}_i^m} o_m(v). \tag{4.4}$$

Those scores allow to rank detected trends by their popularity, and, for example, only the top-n trends can be investigated in subsequent steps.

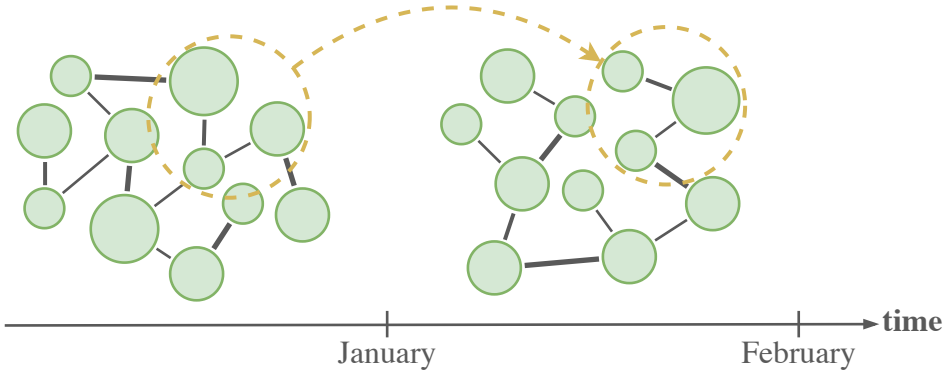
Long-term trends, as opposed to short-lived trends, need to be prevalent over a sufficiently large time span. Therefore, detected hashtag communities need to be tracked over time. In their work, Lorenz et al. (2017) specifically propose a method to capture the dynamics of weighted hashtag co-occurrence networks. Not only does their method allow to track communities of hashtags across subsequent time steps, but also across further distant snapshots. Considering higher-order memory, i.e., taking the networks of multiple previous snapshots into account, their approach allows to overcome issues related to temporal variations and instabilities of the single-layer (static) community detection process. We build on this existing work and leverage their approach to track popular temporal hashtag communities over time, which then form long-term topical trends. Figure 4.3 illustrates the temporal tracking of the communities across the temporal hashtag co-occurrence network snapshots. Even though a hashtag community might reveal slight temporal variations, it can be tracked based on maximum similarity linkage across the snapshots. The resulting sequences of temporal hashtag communities then make up the long-term trends.

4.1.5 Analysis and evaluation

To illustrate the long-term trend detection method described above, it is applied to the political Twitter dataset as outlined in Section "Dataset". Extracted hashtag co-occurrences are aggregated into monthly snapshots. For a global trend description, independent of time, the aggregated network is leveraged. As described in Section "Detection of hashtag communities", the Leiden



(a) Temporal network snapshots



(b) Temporal community tracking indicated by the added dashed lines

Figure 4.3: Illustration of the temporal community tracking, which links communities across multiple temporal network snapshots.

algorithm (Traag et al., 2019) is used for the community detection step. We use modularity as the objective function along with a resolution parameter of 1, $\beta = 0.01$, and 1000 iterations. Edge weights are taken into account. The algorithm is applied 10 times to circumvent suboptimal optimization results due to stochastic variations. Then, only the clustering that leads to the highest modularity score is taken to define the communities of hashtags, i.e., topics. Of course, due to the built-in randomness, repeated runs do not always lead to the same results, but slight variations might occur. Per community, the induced subgraph of the 10 nodes, i.e., hashtags, with the highest PageRank scores (Page et al., 1999) is taken to represent a trend. Trend networks consist of those hashtags as nodes and their weighted interactions. As many communities contain hashtags that are either used for only a short time on social media or are very specific, we focus on the set of the 25 most central nodes, according to their PageRank, and link communities according to the similarity between those sets. For this, we leverage the method proposed by Lorenz et al. (2017) as described in Section “Long-term trend detection”. Four months are used as memory for the

matching procedure to also link communities with temporal fluctuations and to focus on the long-term prevalence of a trend.

In the following sections, we present analysis results covering the prevalence of trends over time, their evolution, and temporal interactions. Finally, results are also evaluated.

Prevalence of trends

As topics are tracked over time, their prevalence and popularity can be investigated with a focus on their temporal development. Not all topics might be equally prevalent at a given time, nor might they occur across all time windows. Also, the popularity of a particular topic might change significantly over time. Figure 4.4 shows a temporal heatmap of the trend scores as outlined in Section 4.1.4. Trend scores are normalized on a trend basis, meaning that a value of 1 indicates the maximum popularity reached for an individual trend. The heatmap shows the 10 trends with the overall highest trend scores and visualizes their development over the 18 months of the entire dataset, from January 2021 until July 2022.

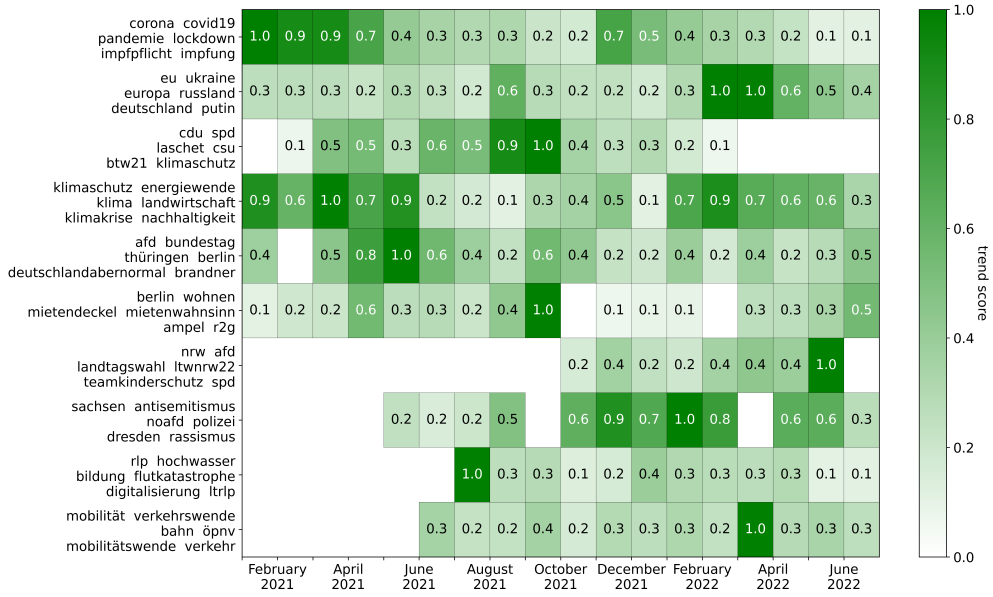


Figure 4.4: Temporal heatmap of trend scores. Larger trend scores indicate a higher level of popularity. The blank spaces show that some trends are not prevalent throughout the entire covered period.

First of all, it has to be noted that some trends, such as the one related to foreign policy and the European Union (row 2), are present across the entire time span, whereas for others, gaps in their prevalence over time become visible. That those gaps are occurring in the trend detection results

confirms that the used method is indeed capable of handling temporal fluctuations. The topic is tracked over time, even though it might not be detected in all intermediate snapshots. In contrast, some trends do not show gaps but are only present for a limited time span. As further described in Section 4.1.5, those trends are often related to some sort of event, e.g., the flood in the Ahr region (row 9). During the occurrence of that event, the trend's popularity is often at its high. All trend developments show periods of higher and lower prevalence. As an example, the COVID-19-related long-term trend (row 1) is most prevalent during the spring and winter of 2021, which might be due to a more tense pandemic situation during those periods.

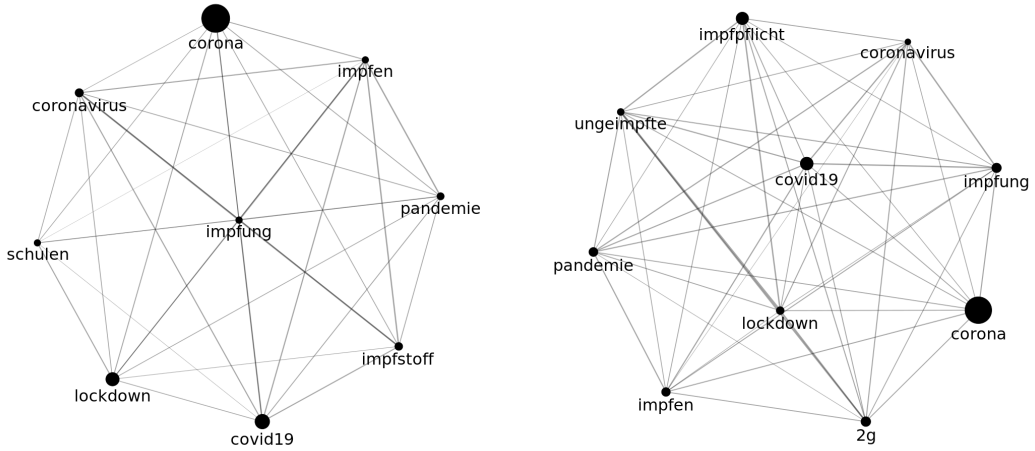
Further, some trends do peak at approximately the same time. Of course, one cannot conclude any causality or correlation from that, but at least the heatmap makes such patterns visible. Exemplary of this are the peaks of the trends related to the Russian invasion of Ukraine (row 2), which also triggered an ongoing media discussion about public transportation (row 10: "mobilität", "verkehrswende") and renewable energy (row 4: "klimaschutz", "energiewende").

Temporal evolution

Topical trends usually do not consist of only a single keyword but are instead described by multiple aspects. With the proposed trend networks, those aspects, represented by hashtags, and their interrelations are intuitively visualized. More interestingly, the temporal changes in the topical trends can be analyzed by tracking them over time. As an example, see Figure 4.5 that shows the trend networks related to the COVID-19 pandemic, as indicated by the respective hashtags, for the two time periods of January 2021 and November 2021. For the graph layout, the `igraph` (Csárdi and Nepusz, 2006) implementation of the Fruchterman and Reingold algorithm (Fruchterman and Reingold, 1991) is used. Even though some hashtags can be found in both networks, e.g., "corona" and "pandemie", other aspects and their importance change over time, e.g., "lockdown" and "impfstoff" vs. "impfpflicht" and "2g". Also, it seems as for this trend, hashtags are a lot more interrelated during November 2021 as more edges in the network show. Represented by their weighting, those edges also indicate relationships of different strengths.

Trend interrelation

Chae and Park (2018) already highlight the importance of topic interrelations. We go in the same direction and analyze *temporal* interrelations between topics. Topics do not co-exist independently of each other but might instead be merged over time or at least become more or less interrelated.



(a) January 2021 (b) November 2021

Figure 4.5: Trend networks related to the COVID-19 pandemic covering different time periods

Figure 4.6 visualizes the temporal interrelations between tracked trends from February 2022 until March 2022. The veins of the alluvial diagram (Rosvall and Bergstrom, 2010) represent the flow of nodes between two communities and, therefore, also the interrelation between topics across time. For the most part, topics seem to be relatively stable as the majority of nodes stays within the same community. Nevertheless, some topics, e.g., the one related to climate protection ("klimaschutz"), also influence numerous others, and nodes of these communities move to other topics. Most notably, a large portion of the climate protection topic shifts to the public transportation topic. To quantify these observations, 114 hashtags stay in the community, whereas 81 shift to the public transportation-related topic. Additionally, 21 shift to the foreign policy topic, and 17 go to the one covering the Ahr flooding. Those results indicate a context switch of certain topical aspects as they also become relevant for other trends.

Evaluation

To confirm that computed trends are meaningful, we leverage an event-based evaluation and manually check if detected trends are related to real-world events. For the top 10 most prevalent trends shown in Figure 4.4, the time frame of their highest popularity is taken as prediction and related events are checked for their temporal occurrence as kind of ground truth. In a subsequent step, the trend peak and the temporal occurrence of the related event are then compared and checked for accordance.

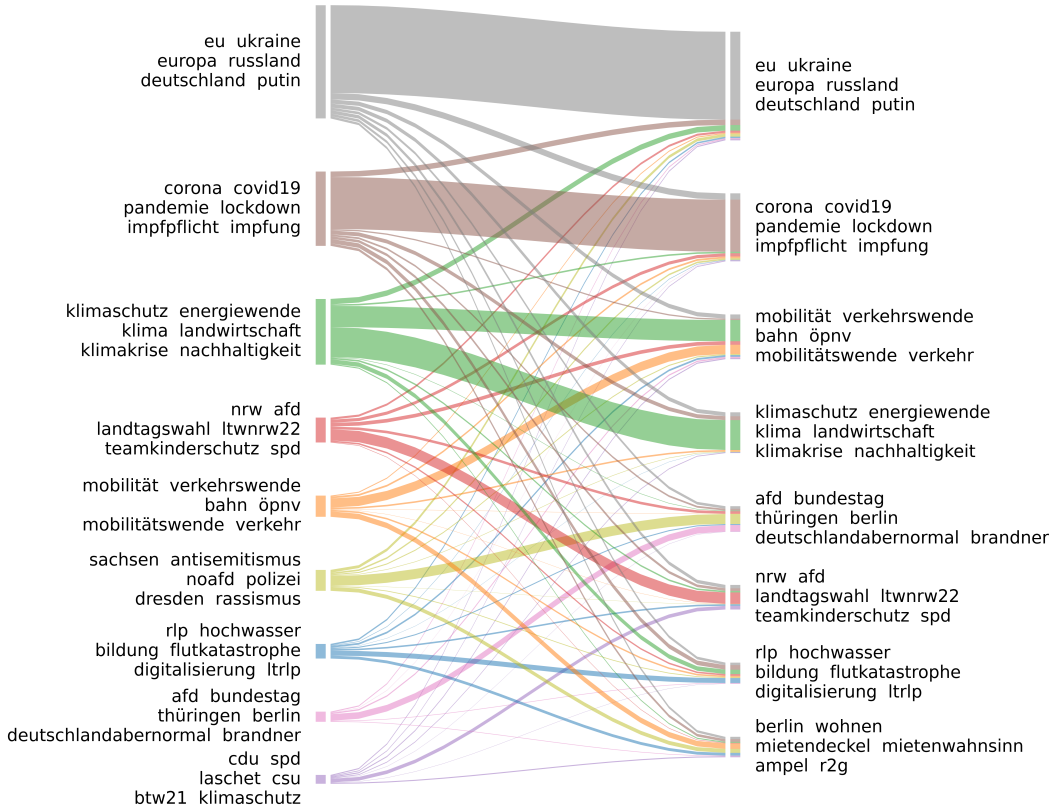


Figure 4.6: Alluvial diagram visualizing the temporal interrelation of trends

Table 4.2: Long-term trends and related events

	Hashtags	**Peak**	**Event**	**Reference** (accessed 2023-06-13)
1	corona, covid19, pandemie, lockdown, impfpflicht, impfung	January 2021	COVID-19 pandemic (17 November 2019 – present)	https://en.wikipedia.org/wiki/COVID-19_pandemic
2	eu, ukraine, europa, russland, deutschland, putin	February and March 2022	2022 Russian invasion of Ukraine (24 February 2022 – present)	https://en.wikipedia.org/wiki/2022_Russian_invasion_of_Ukraine
3	cdu, spd, laschet, csu, btw21, klimaschutz	September 2021	2021 German federal election (26 September 2021)	https://en.wikipedia.org/wiki/2021_German_federal_election

4	klimaschutz, energiewende, klima, landwirtschaft, klimakrise, nachhaltigkeit	March 2021	–	–
5	afd, bundestag, thüringen, berlin, deutschlandabernormal, brandner	May 2021	–	–
6	berlin, wohnen, mietendeckel, mietenwahnsinn, ampel, r2g	September 2021	–	–
7	nrw, afd, landtagswahl, ltwnrw22, teamkinderschutz, spd	May 2022	2022 North Rhine-Westphalia state election (15 May 2022)	https://en.wikipedia.org/wiki/2022_North_Rhine-Westphalia_state_election
8	sachsen, antisemitismus, noafd, polizei, dresden, rassismus	January 2022	–	–
9	rlp, hochwasser, bildung, flutkatastrophe, digitalisierung, ltrlp	July 2021	Flooding of Ahr and Eifel region in Germany (15 July 2021)	https://www.dw.com/en/flooding-in-germany-before-and-after-images-from-the-ahr-and-eifel-regions/a-58299008
10	mobilität, verkehrswende, bahn, öpnv, mobilitätswende, verkehr	March 2022	–	–

Table 4.2 shows that half of the top 10 long-term trends can be related to events like the COVID-19 pandemic or the Russian invasion of Ukraine. Popularity peaks of these trends are in close temporal proximity to the occurrence of the related events. We argue that for the other trends as well, meaningful descriptions can be found, like "Klimaschutz" (Engl. climate protection) for trend 4, "AfD" (German party) for trend 5, "Wohnungsmarkt" (Engl. housing market) for trend 6, "Diskriminierung" (Engl. discrimination) for trend 8 and "öffentliche Verkehrsmittel" (Engl. public transportation) for trend 10. Nevertheless, those trends are not directly linked to real-world events. Together, the event-referenced and manually labeled trends prove good functionality of our long-term trend detection method.

4.1.6 Conclusion

The work presented in this section tackles the issue of detecting long-term prevalent topics hidden in the large volume of short-lived news media. Based on methods known from the field

of temporal network analysis and community evolution, an approach to detect such long-term trends is presented. A case study based on German political Twitter data proves that meaningful trends are detected. Related real-world events can be identified for many top trends, as shown in Section 4.1.5. Future work might target more extensive evaluation procedures and additional quantitative metrics to describe the long-term evolution of trends. Also, a more sophisticated semantic topic model could extend the current trend detection approach. However, this would also require more expertise in the field of NLP, in particular, topic modeling. By now, this work is primarily built on temporal network analysis methods.

4.2 Actor-Networks behind Trends

Trends are a fundamental component of today's fast-evolving media landscape. Still, many questions about who participates in such trends remain unanswered. Do individual actors drive trends, or do interactions between actors reveal community structures? If so, do those structures change during the life cycle of a trend or between topically similar trends? In short: *Who is behind a trend?* Complementary to the methods and analyses examined in Section 4.1, this study aims at a better understanding of the *actor*-networks underlying trends. Also, it focuses on *short-lived* trends as opposed to the previously discussed long-term trends.

The network analytics model proposed in Chapter 3 is well suited for these analysis use cases. Trends are characterized to a large extent by their dynamics. The developed analytics model can also depict such developments over time. Further, since in this analysis, the main focus is placed on the actors involved in the trends and their social interactions, the network-based approach of the model is beneficial. Networks consisting of objects and their relationships are a natural representation of social interaction networks. Given a network-based representation of the trend-related actor-networks, methods known from the field of network science can be leveraged to study the structure and dynamics of these networks and to answer the question of "Who is behind a trend?".

Reference: This section is based mainly on the following peer-reviewed publication:

John Ziegler and Michael Gertz. Who Is behind a Trend? Temporal Analysis of Interactions among Trend Participants on Twitter. *Proceedings of the International AAAI Conference on Web and Social Media*, 17:960–969, 2023.

In the following, after covering the research questions and contributions of this study in Section 4.2.1 and a discussion of related work in Section 4.2.2, we present our approaches to trend de-

tection and the derivation of temporal properties of online social network trends (see Section 4.2.3). We then present various analytical methods that are aimed at a better understanding of the interactions among trend participants (see Section 4.2.4). Finally, we conclude with a summary and discussion of potential future extensions in Section 4.2.5.

4.2.1 Research questions and contributions

This work aims at a better understanding of the actor-networks behind trends on social media, expressed by the general question of "Who is behind a trend?". In more detail, we ask the following research questions:

RQ1 Are trends driven by individual actors, or do interactions between those actors reveal community structures?

RQ2 Do those structures change during the life cycle of a trend or between topically similar trends?

To answer these questions using appropriate computational methods, we conduct a case study based on a large Twitter dataset of more than 16 million tweets collected over the duration of the European soccer championship 2020 (EURO 2020). As an extension to the methodology proposed in Section 4.1, a Gaussian fitting method is developed to identify periods in which selected topics become trends. The detection process also determines the two stages of a trend, the up- (increasing prevalence) and down-trend (decreasing prevalence) stage, as well as the respective duration of a complete trend life cycle. To better understand the relationship among trend participants, we model the dataset as temporal snapshot networks with interactions representing Twitter @mentions. Changes in these networks during a trend (intra-trend) and between similar trends at different points in time (inter-trend) are analyzed. In summary, this study makes the following contributions:

1. To conduct a comprehensive analysis, we leverage a large-scale Twitter dataset that is known to contain topically similar trends over time. Known trends are leveraged for evaluation purposes.

2. By combining changepoint detection and Gaussian fitting, we present a novel method for trend detection. It allows us to distinguish between up- and down-trend, and it helps to determine the duration of detected trends.

3. We use identified trend durations as adaptive time windows and model the dataset as temporal snapshot-based networks accordingly. In contrast to a static window size used in similar

approaches, this adaptivity overcomes the difficulty of properly aggregating snapshots, a problem related approaches are struggling with.

4. An in-depth network analysis of the interactions among actors participating in trends is conducted. Changes within trends and between topically similar trends over time are considered.

Our findings confirm but also significantly extend previous results obtained by related work such as Budak et al. (2011), Asur et al. (2011), and Zhang et al. (2016) as these findings show new methods and results of an actor-centered analysis during and across temporal trends on social media. Regarding the first research question, they show that only a few Twitter accounts take up a large portion of the mentions in tweets, as observed by high domination-ratios, and therefore form the center of the respective trend. Additionally, only a few large communities of actors are present in respective interaction networks, and most users belong to a small group of actors. Regarding the second research question, trends are not centered around stable communities of actors, as shown by an intra-trend analysis that reveals great variability among community members across the life cycle of a trend. These communities are also not temporally stable, as shown in an inter-trend comparison. In contrast, the mentioned highly dominating actors are temporally stable, i.e., re-occurring across trends. Also, trends show a considerable overlap of participating users for topically similar trends at different points in time. Nevertheless, actors participating in a trend are strongly changing during the trend life cycle and vary between the up- and down-trend stages.

4.2.2 Related work

The methods described in the following lie at the intersection of trend detection and network analysis, with some focus on the study of communities. While those topics have already been studied in numerous ways individually, work that connects both aspects is rare. On the one hand, regarding trend analysis, we refer the interested reader to the survey by Sharma et al. (2016). Also, in a more general sense, Yang and Leskovec (2011) investigate temporal attention patterns of online media content. On the other hand, the books by Newman et al. (2006) and Latora et al. (2017) give an excellent overview of network science in general. More specifically, Javed et al. (2018) survey different community detection approaches. Similarly, the work by Rossetti and Cazabet (2018) provides an in-depth coverage of the field of community detection as applied to temporal networks.

The study of Budak et al. (2011) is probably most similar to the work presented in this section. By connecting topical trends with the social network structure of participating users, they are able to identify structurally different types of trends. According to their study, “coordinated” trends,

as opposed to "uncoordinated" ones, are mainly discussed among users that are also "friends" in the respective social network. In contrast, "uncoordinated" trends are not driven by clustered groups of actors but are instead driven by unrelated users. As validation, they use a Twitter dataset of trends connected with information about the Twitter social graph. Although their work also connects trend analysis with social interactions among participating actors, we specifically focus on ad-hoc interactions that are present during trends and further compare those interactions across similar trends at different points in time. With the trend detection method presented in this section, we can also differentiate between different phases of a trend and compare interactions during these accordingly. This differentiation is also an enhancement compared to the methodology presented in Section 4.1 that does not distinguish between the different stages of a trend.

Work that is related to socio-semantic networks also connects topical or, more broadly speaking, semantic networks with actor-networks, which is similar to our approach (Arroyo-Machado et al., 2021; Radicioni et al., 2021; Hellsten and Leydesdorff, 2020). Nevertheless, to the best of our knowledge, no work in the socio-semantic networks field focuses on actor-networks that underlie topical trends on social media.

Further, other additional work is related to ours, such as the one by Asur et al. (2011). Based on a Twitter dataset, they analyze the factors that influence the formation and persistence of trends. Even though they already consider factors influencing the impact of users regarding a trend, they do not investigate the network structures between those trend participants. In a similar direction, the work by Zhang et al. (2016) is concerned with the question of whether the so-called "crowd", meaning a large number of low-impact users or rather "opinion leaders" with a significant influence, contribute the most to trends on social media. They already highlight the importance of ordinary users as opposed to influencers and therefore underline the necessity of our work. Again their focus is not on the interactions among actors participating in a trend. They do not further analyze community structures of such "crowds", and their work lacks a temporal comparison of similar trends re-occurring over time.

Furthermore, recent work by Khan et al. (2021) focuses on detecting and ranking trends based on Twitter data. Unlike our work, they take an open-domain approach and, therefore, detect trends related to different genres but do not check for topically similar trends over time. Also, their method cannot determine the different phases of a trend and its duration. Marangoni-Simonsen and Xie (2015) take a different approach and use a changepoint methodology to detect community emergence in a sequence of networks. Their methodology applies to various kinds of communities and is not limited to the case of hashtag co-occurrence clusters. Finally, Huang et al. (2020) apply changepoint detection to a sequence of network snapshots to find temporal anomalies. In the

present work, we start with a changepoint detection step to find the appropriate window size for the built-upon network aggregations and subsequent analyses. Somehow similar is the work by Anghinoni et al. (2019), which proposes a novel trend detection method. Trends are represented as communities of complex networks that are extracted from time series data. In contrast to the work in this thesis, their work is more theoretical and does not deal with the use case of analyzing actor-networks that underly trends.

4.2.3 Trend detection

The first step of this analysis consists of detecting trends in a collection of social media posts, which represent the documents in this analytics scenario (see Section 3.2.1). Due to the lack of a benchmark dataset that fits the use case of examining actor-networks behind trends, we rely on a Twitter dataset related to the EURO 2020 soccer championship that we specifically collected for this purpose. We check whether our trend detection method delivers high-quality results by conducting an evaluation based on known real-world events. Detected trends and the time windows in which they are present provide the basis for subsequent analytical methods to study interactions among actors contributing to those trends over time. The perspective on the actor-networks underlying the analyzed trends complements the approach presented in Section 4.1, which focuses on the trends themselves along with their temporal evolution and not on the trend participants.

Dataset

The leveraged Twitter dataset consists of posts related to the EURO 2020 soccer competition. We rely on the Twitter search API v2 and gather tweets that either contain the official EURO 2020 account (*@EURO2020*) or the official hashtag (*#EURO2020*). To get a complete dataset, we use a time window that starts one week before (4 June 2021) and ends one week after (18 July 2021) the competition. Time-stamped information about mentions of users in Tweets and what hashtags are used is extracted from the raw tweets. Mentions as interactions among actors compared to retweets are used because they rather represent some social tie as opposed to actions of just sharing information. The statistics of the dataset can be found in Table 4.3.

As Figure 4.7 shows, the activity on the Twitter platform related to the EURO 2020 championship varies over time. A clear peak of attention can be observed during times of the soccer championship final (11 July 2021).

In line with the works of Asur et al. (2011) and Budak et al. (2011), we do not deal with the problem of topic extraction on its own but rather take hashtags as representatives of topics and,

Table 4.3: Rounded statistics of the collected Twitter dataset. For hashtag usages and user mentions, *multiple* occurrences in the same tweet are not considered.

Description	**Count** (*in million*)
tweets	16.163
users	3.802
hashtags	0.266
hashtag usages	27.594
user mentions	19.707

therefore, restrict ourselves to the detection of trends as determined by the temporal usage of hashtags.

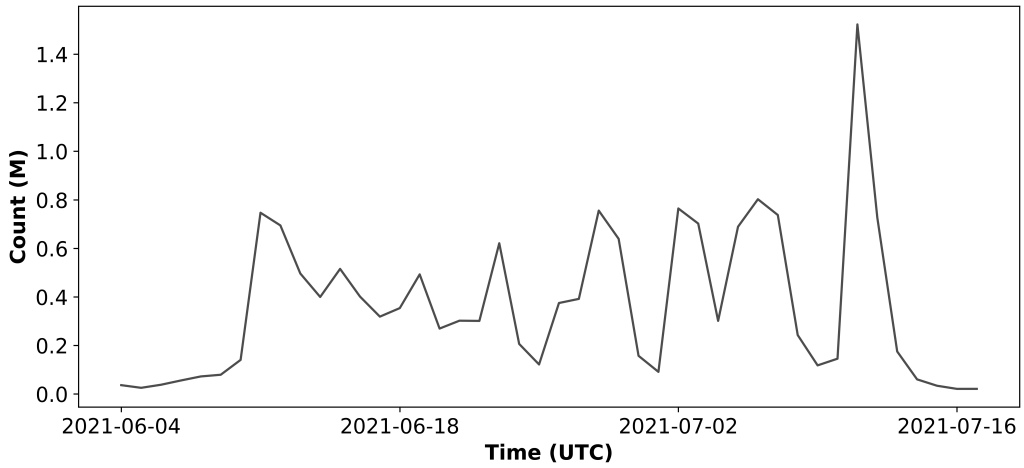


Figure 4.7: Number of tweets over time as part of the collected EURO 2020 dataset

Detection

Given the usage of a hashtag over time, the goal of the detection process is to find time windows in which the hashtag shows trending behavior. In contrast to the trends analyzed in Section 4.1, the trends we analyze in this study are short-lived and do not reveal long-term trending behavior. Based on the analysis model discussed in Chapter 3, an entity network (see Section 3.2.2) consisting of tweets as documents, hashtags as entities, and "usage" links between tweets and hashtags, can be leveraged for this analytics scenario. In this case, the mentioned hashtag usage, respectively occurrence, corresponds to the temporal in-degree of a hashtag node in the respective entity net-

work. As the hashtag usage is given on an hourly basis, the respective temporal network snapshots are also sampled based on a fixed time window of one hour.

Formally, let h be a hashtag, $\mathbb{T}$ the domain of time and the output of the detection process

$$\begin{aligned} &\xi(h) = ((t_{2n-1}, t_{2n}))_{n\in\mathbb{N}} \\ &\text{with} \quad \xi(h)_n \in \mathbb{T} \times \mathbb{T} \end{aligned} \tag{4.5}$$

the sequence of time windows defined as tuples of points in time that specify the duration of the found trends for that hashtag. Further, we refer to the usage of a hashtag h over time as $u_h(t) : \mathbb{T} \to \mathbb{N}$. In a time range $[t_{start}, t_{stop}]$ of a potential trend we model $u_h(t)$ as a Gaussian function with parameters s, c, and d:

$$\begin{aligned} &u_h(t) = s \cdot \exp\left(-\frac{(t-c)^2}{2d^2}\right) \\ &\text{with} \quad t_{start} \le t \le t_{stop}. \end{aligned} \tag{4.6}$$

Given that those parameters control the height, center, and standard deviation of the function, one can think of them as the strength (s), center (c) and duration (d) of the respective trend. In more detail, the characteristic full width at half maximum (FWHM) value of the model is used to determine the actual time range $[t_1, t_2]$ of the trend duration:

$$\begin{aligned} &\text{FWHM} = 2\sqrt{2\ln 2}\, d, \\ &t_1 = c - \frac{\text{FWHM}}{2}, \quad t_2 = c + \frac{\text{FWHM}}{2}. \end{aligned} \tag{4.7}$$

Consequently, time ranges of the up- and down-trend stages of the respective trend are derived as follows:

$$\begin{aligned} &\text{Up-Trend} : [t_1, c] = \{t \in \mathbb{T} | t_1 \le t \le c\}, \\ &\text{Down-Trend} : [c, t_2] = \{t \in \mathbb{T} | c \le t \le t_2\}. \end{aligned} \tag{4.8}$$

Figure 4.8 shows the temporal hashtag usage for *#GER*, which stands for the German national team. It further shows how a Gaussian function is fitted to the time window that potentially contains the trend. Parameters derived by the model are then used to define the characteristics of the present trend. In this case, the trend duration is determined to be about 3.4 hours.

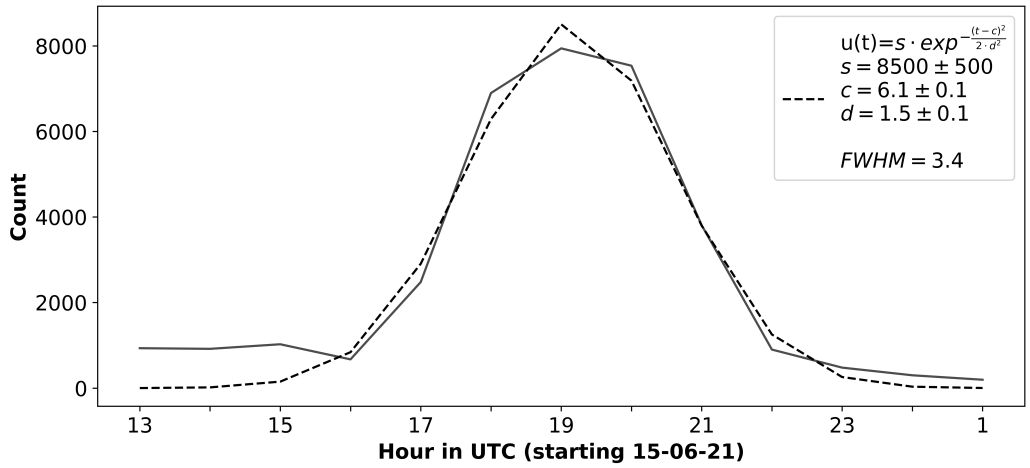


Figure 4.8: Example fit to identify the trend of the hashtag *GER* and its duration

Although Gaussian-like popularity progressions seem to fit our use case (as they are in line with typical temporal attention patterns on social media, e.g., see clusters T1 and T2 in Yang and Leskovec (2011)), it is not yet answered how the time ranges that potentially contain a trend are determined. Constantly checking a sliding time window for a successful model fit is inefficient. Therefore, we leverage a two-step process and first detect changepoints of the hashtag usage via Bayesian Online Changepoint Detection (Adams and MacKay, 2007). If a changepoint is detected, its point in time is taken as the center of a 12 h time window, and the trend detection is only then applied to this time range.

For example, Figure 4.9 shows detected changepoints for the temporal hashtag usage of *#GER*. Interestingly, changepoints are in close proximity to the soccer matches of the German national team. Later on, this pattern is exploited during the evaluation, as described in the next section.

For the actual implementation of the changepoint detection step, we rely on the `Kats`[4] Python package version 0.1.0 as a toolkit for time series analysis. To detect shifts in the average hashtag usage, we use the `NORMAL_KNOWN_MODEL` model parameter, together with the default threshold of 0.5 and a lag of 24. Given an hourly time resolution, the lag of 24 h is inspired by a media logic of daily new trends.

[4] Kats | Kats: `https://facebookresearch.github.io/Kats` (accessed 2023-05-11)

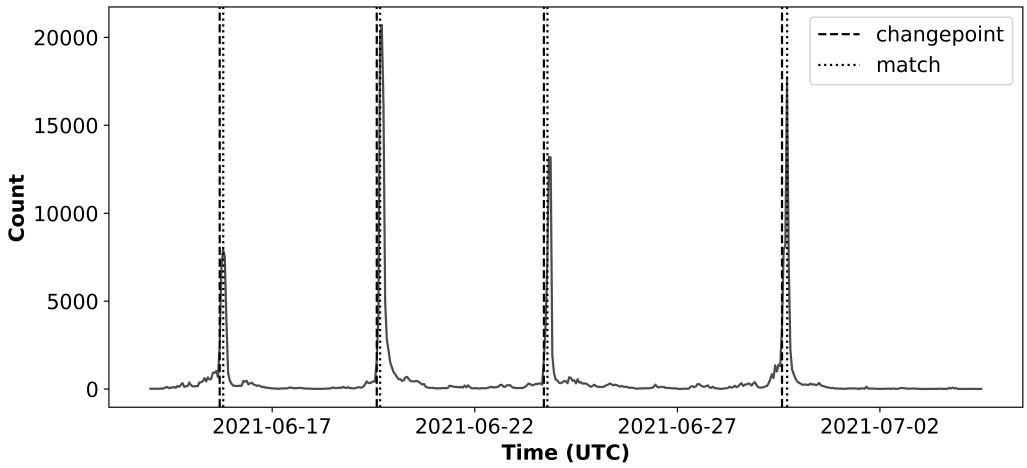


Figure 4.9: Detected changepoints for the hashtag *GER* and soccer matches of the German national team

Evaluation

To verify whether trends found in the present dataset do indeed reflect real-world trends, we follow an approach similar to the one used by Béres et al. (2018) and argue that real-world events characterized by their temporal limitation should also be reflected by a temporally limited shift of representative dataset statistics, in particular those of a trend.

As exemplified by Figure 4.9, detected changepoints and matches of soccer teams, as represented by related hashtags, are close in time. Therefore, we resort to the official schedule of the competition[5] as ground truth and argue that participating teams should be trending in the respective Twitter conversation during matches. This reasoning follows the rationale adopted in Béres et al. (2018). We rely on the official acronyms used in the competition schedule to find tweets that belong to soccer teams. Those are often leveraged as hashtags on Twitter, e.g., *GER* referring to the German national team would often be used as *#GER*.

We check whether trends are detected on the same day as the according team played a EURO 2020 soccer game. Given that for the group stage of the EURO 2020 competition, teams were arranged in groups of four, every team participated in at least three matches during the time window covered by the dataset. In total, 51 matches and, therefore, 51 expected trends are evaluated. Per team, an average of 98 % of the trends are correctly detected, and only 12 % of the detected trends are false positives under the assumption that only soccer matches caused a trend. Further

[5] EURO2021 Match Schedule: https://editorial.uefa.com/resources/026a-126a09addc81-6f092f1f9f89-1000/euro2021_match_schedule_-_english_-_310521_20210601103927.pdf (accessed 2023-05-11)

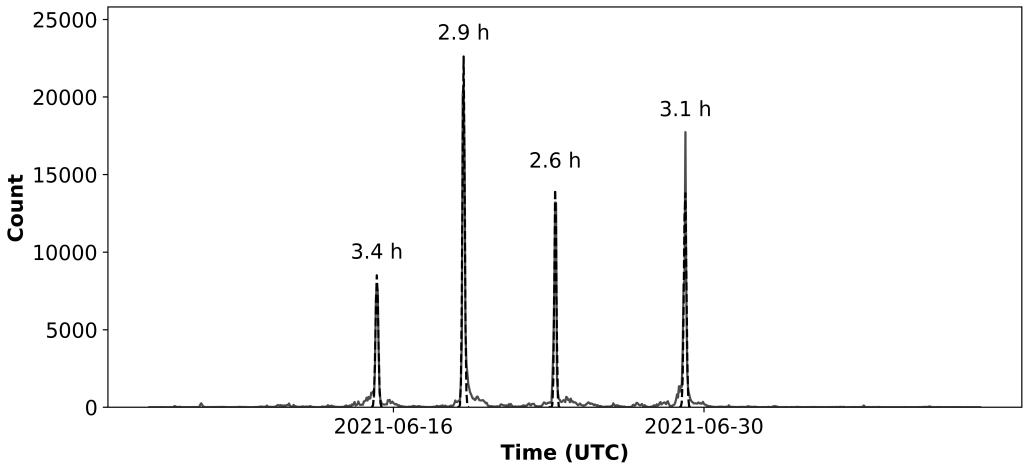


Figure 4.10: Trends and their durations for the hashtag *GER*

investigations reveal that false positive trends appear either the day before the tournament (53 %) or on the last day of the group stage (47 %). One can assume that the increased media coverage before the competition and the decision about which team makes it to the knockout stage, next to actual matches, also caused trends. Also, trend-related discourse on Twitter is not only limited to the actual soccer matches but covers other events as well. As an example, Table 4.4 shows the most liked English tweets published on 12 June 2021 mentioning *#DEN*. Even though the Danish team played against Finland and was trending this day, a large portion of the social media discourse is related to the cardiac arrest of the Danish soccer player Christian Eriksen (Norgaard and McSweeney, 2021). These findings strengthen the assumption that the proposed dataset and methodology are not limited to a soccer-specific social media analysis but apply to studying social media trends in general. Furthermore, the high accuracy gives reason to assume that our trend detection method is reliable and can be used to analyze interactions among actors participating in those trends during the following steps.

Further, a median trend duration of 2.6 ± 0.5 hours underlines the good performance of the approach. Given that a soccer match lasts about 1.5 hours plus a break of 0.25 hours (and often some extra time), this value seems reasonable. Especially one has to consider that news coverage typically starts earlier and stops later compared to the actual time range of the match.

Also related to the identification of trend durations, our proposed fitting procedure converges in all cases except for one (hashtag *SCO* (Scotland) on 14 June 2021, 6 am UTC), as shown in Figure 4.11, meaning that, in general, the Gaussian model can be seen as appropriate. Following a classical trend life cycle, it assumes a rise, peak, and downfall of the according trend. Nevertheless, looking at

Table 4.4: The discourse around trends on Twitter is not only related to soccer matches but also to other happenings, as in this case, the cardiac arrest of the Danish soccer player Christian Eriksen. Here, the most liked English tweets published on 12 June 2021 mentioning *#DEN* are shown.

Tweet ID	**Content**
1403766744624381955	YEEEES! The official word is here. #Erikson is alive! What a horrible scare. My thoughts are with his family, his friends, and all players on the pitch. #DENFIN #DEN #UEFA2020 #EURO2020 🙏 https://t.co/tuviGvtSVp
1403766315681259524	Unbelievable news 👏👏. Just unbelievable 👏👏. #DEN #FIN #EURO2020 https://t.co/tozmCoft1X

the trend for which duration identification fails reveals another possible trend development. In this case, the trend rises in a two- or three-step process and can be seen as multiple overlapping Gaussian functions (think of a Gaussian mixture model (Reynolds, 2009)). Compared to the findings of Yang and Leskovec (2011), the pattern might most accurately be described by their cluster T5, which also describes a two-step increase in attention. Given that this kind of up-trend pattern can be seen as relatively rare, it is not studied further in this work.

Overall, the above method allows us to identify trends and their durations robustly. Applied to our dataset, which also contains interactions among trend participants, it further allows us to analyze and temporally compare the social network structures underlying those trends.

4.2.4 Network analysis

The network analysis part aims at a better understanding of the interactions among trend participants. As already discussed, we resort to mentions in tweets for this purpose. Formally, $G(V, E)$ denotes the directed mention network of users extracted from our Twitter dataset. An edge $e = (v_1, v_2, t) \in E$ contained in the network is defined as a triple of two vertices $v_1, v_2 \in V$ and a timestamp t at which the respective link occurred (i.e., account v_1 mentions account v_2 in a tweet at time t). This approach aligns with the model proposed in Chapter 3. Specifically, the described mention network represents an entity network projection as examined in Section 3.4.1. The according network projection is based on the meta path $username \xrightarrow{belongs\ to} user\ profile \xrightarrow{posts}$

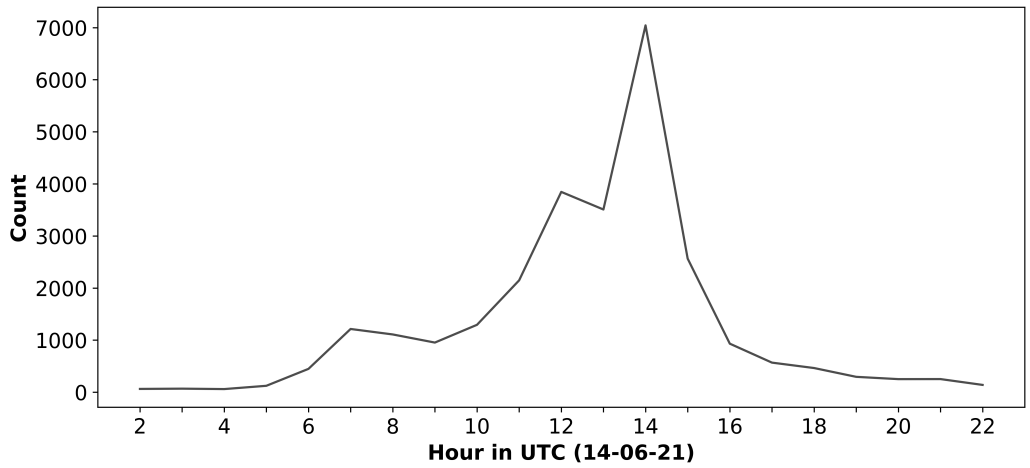


Figure 4.11: Example of a failing trend duration fit. In this case, the trend rises in a multi-step process instead of a single Gaussian-like increase.

$tweet \xrightarrow{mentions} username$. Further, the timestamp of the edge is the result of a network attribution as described in Section 3.4.2.

Given that a trend of a hashtag h is present during the time window lasting from t_1 until t_2, the set of users $V_h^{t_1,t_2}$ and the set of mentions $E_h^{t_1,t_2}$ define the interaction network $G_h^{t_1,t_2}$ among trend participants. Participants are those users that used the hashtag at least once during the time of the trend. In other words, a meta path of the structure $username \xrightarrow{belongs\ to} user\ profile \xrightarrow{posts} tweet \xrightarrow{uses} hashtag$ must exist for a *username* to be included in the participation network as user node. Formally, the set of interactions that have to be taken into account is defined as follows:

$$E \supseteq E_h^{t_1,t_2} = \{e = (v_1, v_2, t) \in E | (v_1 \in V_h^{t_1,t_2} \vee v_2 \in V_h^{t_1,t_2}) \wedge t_1 \leq t \leq t_2\}. \tag{4.9}$$

As a result, for a sequence of trends $\xi(h) = ((t_1, t_2), (t_3, t_4), \ldots, (t_{2n-1}, t_{2n}))$ detected for a given hashtag h, one obtains a series of snapshot networks $G_h = (G_h^{t_1,t_2}, G_h^{t_3,t_4}, \ldots, G_h^{t_{2n-1},t_{2n}})$ that contain the interactions among participants during those trends. Note that according to the respective trend durations determined with the approach described in Section "Detection", snapshots are adaptively sized instead of having a fixed window size.

Given the defined interaction networks, methods known from the field of community detection (see Javed et al. (2018)), e.g., Infomap, can be applied to gain a better understanding of the interaction patterns. Also, successive interaction networks can be compared based on their similarity.

We rely on the overlap coefficient as a similarity measure for that (Vijaymeena and Kavitha, 2016). Given two sets of vertices V_1 and V_2, the overlap coefficient α is defined as

$$\alpha(V_1, V_2) = \frac{|V_1 \cap V_2|}{min(|V_1|, |V_2|)}. \tag{4.10}$$

Further, as an extension to the overlap coefficient α as defined in Equation 4.10, we specify the relative overlap α_r as the intersecting fraction relative to the cardinality of the first set:

$$\alpha_r(V_1, V_2) = \frac{|V_1 \cap V_2|}{|V_1|}. \tag{4.11}$$

In the following, we will talk about "inter-trend" comparison when comparing interaction networks for the same hashtag at different points in time. As an example, comparing the four trends of the hashtag *GER* shown in Figure 4.10 would follow the "inter-trend" approach. In contrast, "intra-trend" analysis compares the up- and down-trend phases within a single trend. It focuses on only a single trend period.

Regarding the implementation of this framework, we rely on the `python-igraph` software package version 0.9.8 (Csárdi and Nepusz, 2006). For community detection, we use the Infomap community detection algorithm, which is already included in the `igraph` package. Centrality measures are derived via PageRank scores and the community detection is applied per snapshot. We argue that approaches from the field of evolutionary clustering (Chakrabarti et al., 2006) are not feasible for our community detection use case due to their inherent assumption that subsequent clusters should be similar, see the discussion about the method of "Temporal Smoothing" in Section 3.2 of Rossetti and Cazabet (2018). Our approach is less restrictive than those methods, given that we do not assume to find communities or stable clusters of communities at all.

As an example, Figure 4.12 shows the interactions among the five largest communities during the trend on Twitter accompanying the match of Germany vs. France. Communities are labeled according to the three most central nodes within the respective cluster.

Inter-Trend Results

Overall, the interaction networks derived from our dataset contain, on average, about 7 ± 4 k vertices and 20 ± 10 k edges, corresponding to a very low density of 0.0004 ± 0.0002. This result already shows that not a single highly connected group of nodes participates in the trends, but interactions are spread across a large user base. Furthermore, the high deviations show that trends

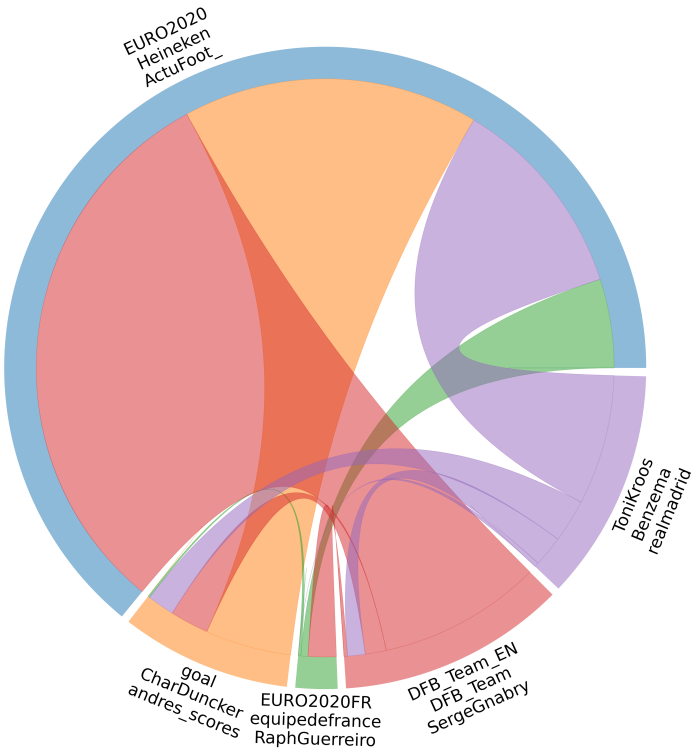


Figure 4.12: Exemplary mentions between the five largest communities in a trend, related to the soccer match of Germany vs. France, visualized as a chord diagram

significantly vary in their spreading across the social media platform. If one takes mentions as an indicator of the social network between actors, the terminology used in the study of Budak et al. (2011) can be leveraged. In that sense, analyzed trends can be classified as "uncoordinated", meaning that they are discussed among distributed as opposed to clustered users.

Interestingly, on average 21 ± 4 % of nodes overlap for successive trend networks shown by the median value $\alpha(V_b^{t_i,t_{i+1}}, V_b^{t_{i+2},t_{i+3}}) = 0.21 \pm 0.04$. Considering that real-world matches against different teams cause two comparable trends, one would expect a maximum overlap of about 50 %, assuming that the same amount of users represents supporters of one and the other team. Given this assumption, the fraction of overlapping users seems to be quite high, and one can argue that a similar user base is participating across topically similar trends. From that, one could conclude that the fan base supporting a team stays similar across matches. Alternatively, participants might as well be not supporters of individual teams but general soccer-related actors, e.g., sports media outlets or journalists.

Table 4.5: Domination-ratios for different trend phases, in- (I) and outgoing (O) mentions, as well as the top one (1) and ten (10) ranked users

	Complete	**Up-Trend**	**Down-Trend**
I1	0.16 ± 0.03	0.17 ± 0.04	0.14 ± 0.03
I10	**0.46 ± 0.04**	**0.48 ± 0.04**	**0.48 ± 0.06**
O1	0.008 ± 0.003	0.02 ± 0.01	0.009 ± 0.004
O10	0.05 ± 0.02	0.09 ± 0.05	0.05 ± 0.02

To determine whether single users play an outstanding role in the network, similar to Asur et al. (2011), we calculate the domination-ratio as the proportion of mentions that come from or go to the most mentioning/mentioned user. For an even more meaningful quantity, we also sum up the domination-ratios for the ten most active participants (see Table 4.5 "I10" and "O10"). Because mentions imply a direction, we differentiate between out- and ingoing mentions. As shown in Table 4.5, especially for ingoing mentions, the domination-ratio is relatively high, which means that only a handful of user accounts take up a large portion of the overall mentions and thereby form the core of actors that the trend is centered around. One might call those actors "trend-hubs" or "trend-influencer". Interestingly, the top ten most dominating users reach a median overlap coefficient of 0.46 ± 0.04. This value is close to the expected maximum value of 0.5, as explained above.

Contrarily, no accounts dominate the actual mentioning activity, as the domination-ratios for outgoing mentions are low. This finding aligns with the insights of Asur et al. (2011), who highlight the link between low domination-ratios and longer trend durations. Zhang et al. (2016) as well outline the importance of the "crowd" participating in a trend for it to gain considerable popularity.

The most dominant user accounts regarding ingoing mentions during matches of the German national team are shown in Table 4.6. Most actors are related to soccer players and teams, national soccer associations, or the EURO 2020 championship. Obviously, those users have a large relevance for soccer-related trends. Accounts closely linked to the German national team, in this case *DFB_Team_EN* and *DFB_Team*, can be found across all trends. The *EURO2020* account can be seen as an artefact of the dataset collection process. On the other hand, one can also observe large variations between the different matches/trends. Depending on the opposing team accounts related to this one also become relevant, e.g., *England* or *sterling7*, for the match against the English team.

Table 4.6: Top ten most dominating users for trends during matches of the German national team

GER vs. FRA	**GER vs. POR**	**GER vs. HUN**	**GER vs. ENG**
EURO2020	EURO2020	EURO2020	EURO2020
goal	DFB_Team	goal	goal
InvictosSomos	goal	Football__Tweet	DFB_Team
Football__Tweet	DFB_Team_EN	Footballogue	England
DFB_Team_EN	ToniKroos	SquawkaNews	DFB_Team_EN
DFB_Team	Cristiano	DFB_Team	sterling7
Cristiano	realmadrid	DFB_Team_EN	BBCSport
EURO2020FR	InvictosSomos	InvictosSomos	ChelseaFC
realmadrid	2010MisterChip	brfootball	ManCity
ToniKroos	selecaoportugal	2010MisterChip	Football__Tweet

Besides individual actors, when it comes to community detection, on average, 700 ± 300 communities per network are found. Of these communities, most consist of only a small number of users, as shown in Figure 4.13. The distribution of community sizes approximately follows a power law decay. The largest communities, on average, consist of 14 ± 4 % of nodes of the complete network. Again, strongly deviating values are observed, which suggests large differences in the trend dominance of single communities.

Interestingly, a pairwise comparison of the top ten largest communities between trends reveals large fluctuations. On average, even the maximum overlap coefficient with a value of 0.21 ± 0.04 does not surpass the similarity score between complete networks. Therefore, communities do not seem to be temporally stable across trends. This finding is underlined by a very low overlap between the largest communities of subsequent trends (0.04 ± 0.03). Also, only a small fraction of nodes of the largest community during a trend can be found during the next trend ($\alpha = 0.13 \pm 0.03$).

To verify whether specific communities of actors continue within other communities in succeeding trends, we check for the maximum fraction of users that stay together between any pair of the top ten communities in successive trends. We use the relative overlap coefficient α_r as defined in Equation 4.11 for that. Figure 4.14 gives a visual example of such a community flow in the form of an alluvial diagram. It should be noted that the sizes of the community blocks are not absolute but relative to the in- respective outgoing overlap. The top three accounts with the highest centrality values are used as community labels. One can see that the community containing the *DFB_Team_EN* Twitter account continues partially across trends. Nevertheless, for the most

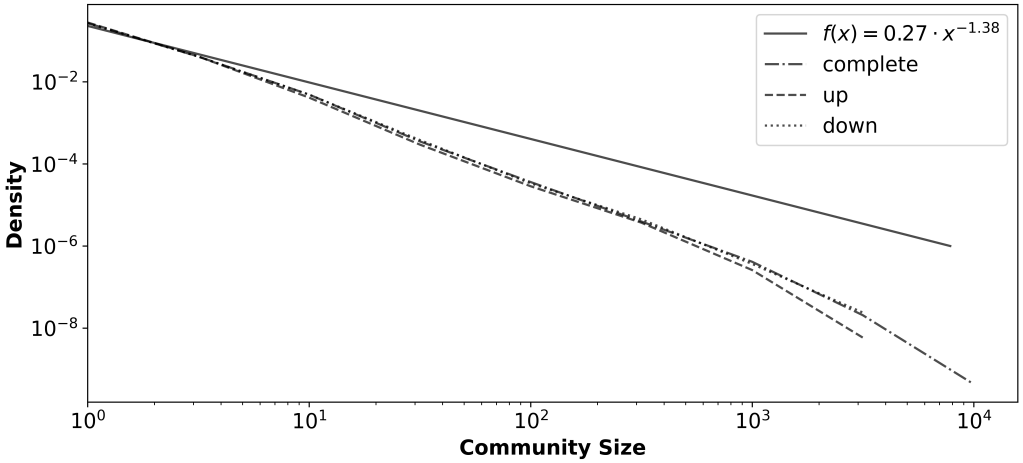


Figure 4.13: Community sizes differentiated by trend stages. Densities approximately follow power law distributions.

part, communities get mixed quite strongly as a low maximum value of $\alpha_r = 0.10 \pm 0.03$ between successive top ten communities indicates.

Intra-Trend Results

In contrast to the inter-trend setting, the intra-trend analysis compares interaction networks between different trend stages (up- vs. down-trend). The mention networks during those stages contain, on average, about 3 ± 2 k respectively 5 ± 3 k vertices, as well as 6 ± 4 k respectively 11 ± 6 k edges, meaning that the mention networks reveal very low densities of 0.0008 ± 0.0005 respectively 0.0005 ± 0.0003. Similar to the findings for the inter-trend analysis, a group of loosely interacting actors seems to participate across the stages of a trend. In this case, one could as well talk about "uncoordinated" trends (Budak et al., 2011). Furthermore, it must be noted that, on average, more distinct users participate in the later phase of a trend. This disparity can probably be explained by an already larger popularity of the trend at that point in time.

Usually, one would not expect entirely differing networks during up- and down-trend phases. However, this is what our analysis reveals. Within a single trend, only a median of 28 ± 4 % of nodes is overlapping, showing that only about a quarter of the users participates across the complete trend life cycle. Also, regarding community analysis, results are similar to the ones found for the inter-trend setting. On average, 400 ± 200 respectively 500 ± 200 communities per network are found. Figure 4.13 shows the distribution of community sizes over all actor-networks. As can be seen, communities primarily consist of only a few users, a property that holds for all trend

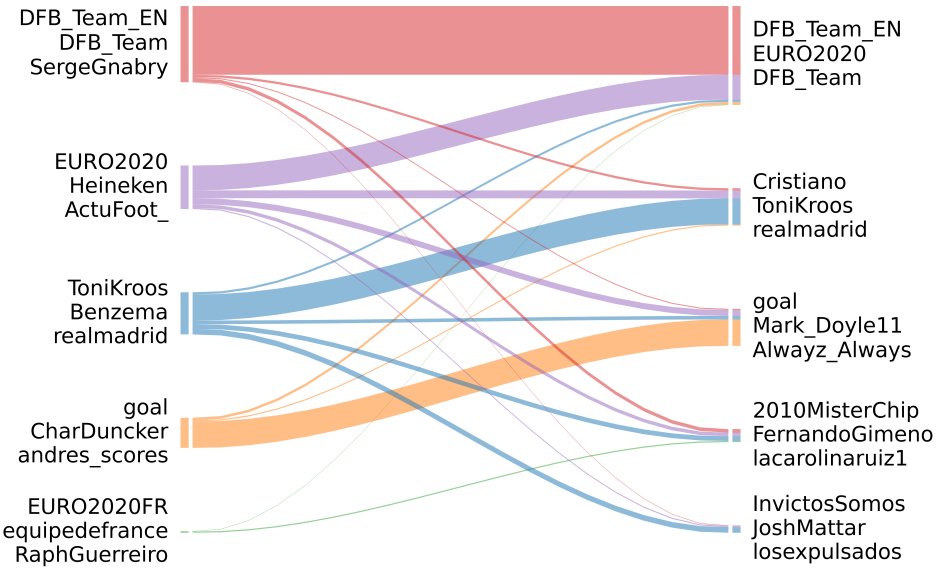


Figure 4.14: Exemplary inter-trend overlap of the five largest communities visualized as an alluvial diagram – GER vs. FRA (left) GER vs. POR (right)

stages. The distributions of community sizes approximately follow power laws with even less large communities as expected by named distribution. The largest communities cover on average 16±6 % respectively 14 ± 5 % of actors of the complete ad-hoc network and are thereby quite dominant among respective interactions. On average, the maximum similarity in terms of overlap coefficient between the top ten largest communities during the up- and down-trend is only 0.28 ± 0.08. As the two stages are part of the same trend, one would expect a more stable base of actor communities contributing to the trend across its life cycle. Similar results are obtained by simply comparing the largest communities between the two trend stages. On average, only a fraction of 6 ± 6 % of actors overlaps. Also, again a low maximum value of $\alpha_r = 0.10 \pm 0.04$ is observed by comparing the top ten communities of successive trend stages. Nevertheless, in all cases, actors of the largest community found during the up-trend also participate in the down-trend, meaning that at least those actors are stable across the entire trend life cycle.

Furthermore, findings related to the domination of users, as shown in Table 4.5 and already discussed for the inter-trend analysis, also hold for the intra-trend setting. Domination-ratios of 0.48 regarding the top ten ranked users can be observed for both trend stages. By comparing a maximum of ten most dominant accounts between trend stages, an overlap of 60 ± 10 % is observed on average. Dominant accounts seem to be relatively stable within a trend. For example, the top five dominant users during matches of the German national team regarding ingoing mentions and separated by trend stages are shown in Table 4.7. One might expect that the changes in these

Table 4.7: Top five most dominating users for different trend stages during matches of the German national team

Stage	GER vs. FRA	GER vs. POR	GER vs. HUN	GER vs. ENG
Up	DFB_Team	EURO2020	goal	EURO2020
	Cristiano	InvictosSomos	DFB_Team_EN	DFB_Team_EN
	EURO2020	DFB_Team_EN	EURO2020	goal
	goal	DFB_Team	DFB_Team	DFB_Team
	DFB_Team_EN	goal	InvictosSomos	England
Down	EURO2020	EURO2020	EURO2020	EURO2020
	goal	DFB_Team	goal	goal
	Football__Tweet	DFB_Team_EN	DFB_Team	England
	DFB_Team_EN	ToniKroos	DFB_Team_EN	DFB_Team_EN
	DFB_Team	Cristiano	InvictosSomos	BBCSport

dominant accounts between the two trend stages are caused by different foci in media attention, e.g., pre-match expectations vs. moderating the soccer match. Unfortunately, analysis of the actor-networks does not tell much about the topical variations within a trend life cycle. More suitable semantic analyses are needed to better understand how media coverage (e.g., discussed topics) changes over the lifespan of a trend.

4.2.5 Summary and discussion

Given the important role of social media platforms and their influence on society, obtaining insights into who contributes to trends and how underlying actor-networks are structured will continue to play an important role. In the present work, we tackle the problem of combining two core methods for such studies, trend detection and network analysis, to get a more fine-grained view of social interactions among actors during trends on social media. In the context of a large-scale Twitter dataset related to the 2020 UEFA European Football Championship, we developed several novel methods to analyze and explore trends. Our novel Gaussian-based trend detection method allows us to differentiate between up- and down-trend as well as to determine the duration of a trend. An event-based evaluation proves good performance.

Furthermore, with the dataset and detected trends on hand, we are able to analyze topically similar trends across time (inter-trend) and within the trend life cycle (intra-trend). Our analysis focuses, in particular, on the actors behind those trends, modeled in the form of temporal men-

tion networks extracted from the dataset. Trend-dependent time windows allow for an adaptive snapshot-based network aggregation. In this regard, the study also serves as a demonstration of the analytics model presented in Chapter 3. Its temporal, network-based approach is well suited for the present use case of studying interaction networks among trend participants.

Among other results, our findings show a considerable overlap of the user base in an inter-trend setting but not within a trend during its different stages. Users participating in a trend seem to vary a lot during the life cycle of a trend but not so between similar trends across time. Furthermore, trends are centered around a small set of highly influential users, as indicated by high domination-ratios. This core of actors is also stable across time. In contrast, even though large communities of actors are present, these are neither stable within nor across trends.

In general, the methods and techniques described in this section provide a solid basis for studying actor-networks underlying trends on social media. Also, this study's methodology and analysis results complement the insights examined in Section 4.1. While the previous work presented in Section 4.1 focuses on the detection and analysis of long-term trends, the present study focuses on investigating the *actor*-networks underlying *short-lived* trends.

Outlook

Even though these insights are primarily based on the analyzed Twitter EURO 2020 dataset, similar results can be obtained for other types of social media datasets as analyzed content is not only centered around soccer matches but covers the discourse around general events as well (see Section "Evaluation"). Also, the trend detection method captures common media attention patterns, meaning that it applies to other trend analysis scenarios as well. In this sense, the present work could be extended in several ways. For example, the trend detection method elaborated in Section "Detection" could be enhanced by techniques that can deal with different trend progressions, such as the one shown in Figure 4.11. Also, incorporating additional information like terms and named entities into the proposed network model might complement the analysis with a better semantic understanding of given trends. This way, topical shifts within and across trends might also be recognized. Furthermore, it might be interesting to integrate additional data sources to gather a more complete picture of who is participating in a trend on which platform. Are there cross-platform patterns that emerge synchronously, e.g., in a coordinated fashion by a small group of users? Additionally, from a methodological point of view, evolutionary clustering might lead to different results when it comes to detecting temporally stable communities in the actor-networks behind analyzed trends. Also, in combination with the methodology developed in Section 4.1, it might be of interest to analyze the actor-networks underlying long-term trends. Can one find

similar network structures as in the case of short-term trends? Do the actor-networks change in a similar way compared to short-lived trends? How do the actor-networks differ structurally between multiple long-term trends?

5 Conversations

Conversations are an integral part of online social media, for example, in the form of Twitter replies or YouTube comments. Gaining insights into these conversations is of significant value for many commercial as well as academic use cases as it contributes towards a better understanding of the investigated community and the discussed topics. From a computational perspective, however, analyzing conversation data is complex, and numerous aspects must be considered. Next to the structure of conversations, the discussed content – as well as their dynamics – have to be taken into account. Still, most existing modeling and analysis approaches focus only on one of these aspects and, in particular, lack the capability to investigate the temporal evolution of a conversation. To address these shortcomings, in this chapter, we present CODY, a content-aware, graph-based framework to study the dynamics of online conversations along multiple dimensions and, thereby, follow the approach of the model presented in Chapter 3. Its capabilities are extensively demonstrated by conducting three experiments based on a large conversation dataset from the German political Twittersphere. First, the posting activity across the lifetime of conversations is examined. We find that posting activity follows an exponential saturation pattern. Based on this activity model, we develop a volume-based sampling method to study conversation dynamics using temporal network snapshots. In a second experiment, we focus on the evolution of a conversation's structure and leverage a novel metric, the temporal Wiener index, for that. Results indicate that as conversations progress, a conversation's structure tends to be less sprawling and more centered around the original seed post. Furthermore, focusing on the dynamics of content in conversations, the evolution of hashtag usage within conversations is studied. Initially used hashtags do not necessarily keep their dominant prevalence throughout the lifetime of a conversation. Instead, various "hashtag hijacking" scenarios are found. Overall, the work on the evolution of online communities presented in this chapter serves as another use case example of the network analytics model proposed in Chapter 3.

Reference: This chapter is based mainly on the following preprint publication. Its content results from the collaboration with Fabian Kneissl, whom the author supervised during his master thesis, "Time-Dependent Graph Modeling of Twitter Conversations":

John Ziegler, Fabian Kneissl, and Michael Gertz. CODY: A graph-based framework for the analysis of COnversation DYnamics in online social networks. *arXiv preprint arXiv:2310.08140*, 2023.

The remaining part of this chapter is structured as follows: After an introduction in Section 5.1 which covers the study's research questions and contributions, in Section 5.2, related work is outlined and discussed. Next, the CODY model is introduced in Section 5.3. Its capabilities are experimentally examined in Section 5.4. Finally, Section 5.5 summarizes this chapter.

5.1 Research Questions and Contributions

In a recent work by Brambilla et al. (2022), the authors state that "[m]ost studies on social networks have focused only on user relationships *or* [emphasis added] on the shared content, while ignoring the valuable information hidden in the digital conversations, in terms of structure of the discussion and relation between contents, which is essential for understanding online communication behavior." In summary, they highlight the need for an analysis model to incorporate a conversation's content *and* structure. While this insight already leads towards a more holistic conversation analysis model, it still lacks an essential aspect. As conversations are fundamentally characterized by their temporal evolution, a respective analysis model should also consider a conversation's dynamics. To the best of our knowledge, existing conversation analysis approaches are still missing this aspect (Cogan et al., 2012; Saveski et al., 2021; Brambilla et al., 2022). In line with the model presented in Chapter 3, we overcome these shortcomings by proposing a graph-based conversation analysis model, named CODY, which incorporates the conversation's *structure*, *content*, and *dynamics*. Its capabilities are demonstrated by focusing on three main research questions:

RQ1 How does the posting activity change over the lifetime of a conversation? Does a generalizing model describe the observed activity pattern?

RQ2 How do conversations structurally evolve? Is there a metric to quantitatively describe the structural evolution?

RQ3 How does the focus of discussed topics change over the lifetime of a conversation? Do initially used hashtags keep their dominant prevalence?

To answer these research questions, an analysis model needs to be developed that is versatile enough to cope with the three dimensions of a conversation – structure, content, and dynamics – as no current approach exists for that. The proposed CODY model is based on the concept of temporal heterogeneous information networks (THINs) (see Sun et al. (2010), Li et al. (2018),

and Milani Fard et al. (2019)) and therefore comes with proper flexibility to model a conversation's evolution, along with its structure and content. By leveraging a large conversation dataset from the German political Twittersphere, the model's applicability is shown and the above research questions are investigated. The developed methodology and the results of our experiments make up several contributions:

- The graph-based CODY model constitutes a versatile framework to study the structure, content, and dynamics of conversations. Its flexibility in terms of integrated node/edge types allows to model various (content) objects and their interplay within a conversation.
- RQ1: Based on empirical results, the posting activity of a conversation follows an exponential saturation pattern. Contributions occur mainly at the early stage of a conversation, and participation becomes less towards the end.
- The developed activity model lays the theoretical foundation for a volume-based sampling approach used to model a conversation's evolution based on temporal snapshots. The common issue of finding an appropriate sampling technique is overcome with that.
- RQ2: The newly proposed temporal Wiener index metric allows to quantitatively measure a conversation's structural evolution. Experimental results show that as conversations progress, their structure tends to be less sprawling and more centered around the original post.
- RQ3: Focused on the content of conversations, investigations regarding the temporal prevalence of hashtags show that initially used hashtags do not necessarily keep their dominant role across a conversation's lifetime. Instead, various "hashtag hijacking" scenarios can be observed.

Terminology

Regarding the used terminology, in line with previous work (Cogan et al., 2012), we focus on conversations that start from an initial social media post and that are made up of direct user interactions related to this post rather than "conversations" that are centered around an event or other social media entities, such as @mentions. In our understanding, conversations must be based on direct interpersonal communication instead of latent social interactions. In the context of the Twitter platform, replies are considered "[...] the most direct communication sign between two users." (Cogan et al., 2012) Therefore, Twitter reply threads make up conversations.

5.2 Related Work

The study presented in this chapter touches on different research topics. Even though differentiation is not always definite, this section aims to clarify the study's similarities and differences compared to existing work. For that, related work from online conversation analysis is covered first. Secondly, the difference between this study and information diffusion is explained. Finally, methodological similarities are described by summarizing work that also leverages THINs.

Online conversation analysis: Most closely related to our work are studies about the modeling and analysis of online conversations. As such, Brambilla et al. (2022) conduct an in-depth study of various online conversations. For their analysis, they propose a graph-based framework and mainly focus on intention analysis as well as network construction. In contrast to us, they do not consider the dynamics of the conversation graph itself. Still, they consider temporal aspects of the analyzed conversations, such as their duration and typical reply times. Similar to our approach, they also view the network as heterogeneous by including different node and edge types. Recently, Botzer and Weninger (2023) leverage entity graphs extracted from online conversation data, in their case Reddit, to study whether online discourse is predictable, how the online conversations are structured and whether theories on spreading activation can be successfully applied in the context of online discourse analysis. Compared to our work, they do not investigate the conversation graphs' dynamics or consider the heterogeneity of the conversations' content. Further, in one of the early works on conversation graph analysis, Cogan et al. (2012) investigate the characteristics of conversation networks emerging from an initial social media seed post. This understanding of a conversation graph is in line with our definition (see Section 5.1). In their work, they propose a method to gather conversation data as completely as possible and conduct an experimental analysis based on Twitter data. Their outlook highlights the importance of future work considering a conversation graph's temporal evolution.

Focused on the aspect of toxicity, Saveski et al. (2021) also study the structure of Twitter conversations. They conduct their analysis on different levels, from individual users to the entire conversation group. Even though applied in a static setting, they also leverage the Wiener index to investigate the topology of analyzed reply trees. In two prediction tasks, they aim at forecasting future toxic conversation characteristics based on previous conversation data. In another application-focused work, Mathew et al. (2016) analyze Twitter conversation data related to e-commerce promotional events. They model the data as a user-centric network by considering replies, mentions, and retweets for the weighted edges and Twitter users as nodes. Regarding the network's temporal evolution, they determine the users' flow between different network components across time. Nevertheless, they do not elaborate on their temporal graph model but only

propose a sampling approach using fixed time windows. Also, they do not investigate the dynamics of any network metric and solely focus on users, not the content, as part of a conversation.

Temporal heterogeneous information networks: As part of our methodology, we leverage THINs to model the evolution of online conversations. Even though, to the best of our knowledge, none of the existing work applied THINs in the context of conversation analysis, they are extensively used for other use cases. For example, Sun et al. (2010) propose a method to detect communities in THINs. They understand THINs as a sequence of temporal network snapshots, which aligns with our approach. Similarly, Cuzzocrea and Folino (2013) deal with community detection in temporal information networks. They specifically focus on the temporal tracking of the detected communities and propose a detection method to analyze structural changes regarding the community structure of the evolving network. Further, again in the context of THINs, Milani Fard et al. (2019) tackle the task of meta path prediction. They leverage temporal snapshots to model a network's evolution. Sajadmanesh et al. (2019) also predict relationships in THINs. Additionally, they predict the time when these relationships will occur. Further, as described in Section 3.3.3, multiple other use cases are elaborated on in the context of THINs.

Information diffusion: Information diffusion is a topic that is also extensively studied in the context of online social networks. The survey by Guille et al. (2013) and the more recent one by Yujie (2020) give a good overview of this research field. Commonly used information diffusion models also leverage topological information of the underlying social network. The spreading process is frequently modeled as a network, referred to as diffusion graph. In contrast to our work, information diffusion mainly investigates how information spreads across the entire network. We, however, are less interested in the information diffusion process but instead focus on the graph-centric evolution of individual online conversations, which includes not only the discussed information but also the participating actors and the structure of a conversation. Still, based on our model, temporal meta paths related to the concept of THINs could be leveraged to study information diffusion processes within individual conversations.

5.3 CODY Model

As already described, a conversation's structure, content, and dynamics must be considered for a conversation analysis model to be leverageable in different analysis scenarios. Following the same pattern, this section introduces the CODY model in three steps. First, Section 5.3.1 explains how the CODY model considers the structure of conversations. Secondly, the temporal evolution of a conversation is integrated into the model in Section 5.3.2. Finally, Section 5.3.3 complements the model by additionally considering the content of conversations. Along with developing the

model, the reference to the network analytics model proposed in Chapter 3 is always made. Also, apart from the conversation model, metrics to investigate a conversation's structural evolution are needed. Therefore, Section 5.3.4 introduces a novel metric, called *temporal* Wiener index, to quantitatively study how a conversation's structure evolves over time.

For an illustration of a conversation represented using the CODY model, see Figure 5.1. Different node and edge types are indicated by different symbols and line styles, respectively. Further, temporal network snapshots are used to show the conversation's dynamics.

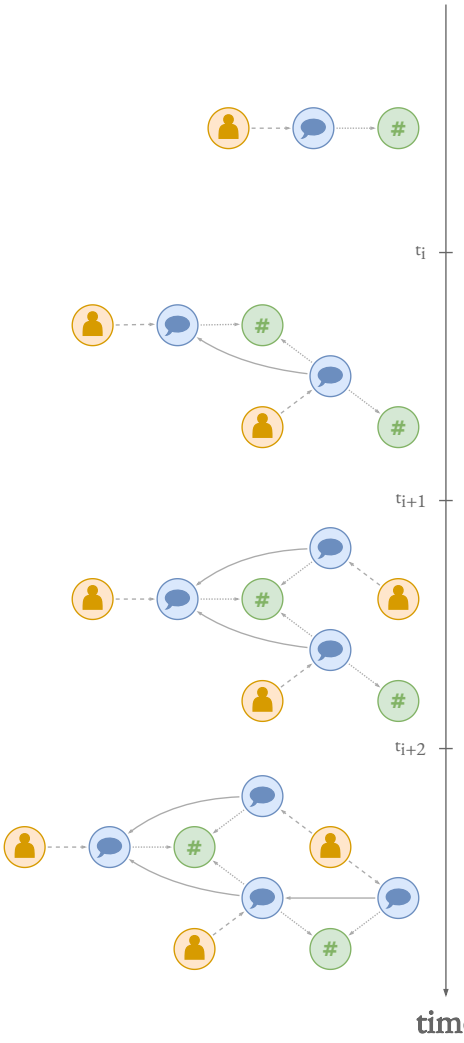


Figure 5.1: The graph-based CODY model incorporates three central dimensions of a conversation: structure, content, and dynamics. Different node and edge types are represented by different symbols and line styles, respectively. Temporal snapshots enable the dynamics of the conversation to be modeled.

5.3.1 Structural conversation model

Considering the structure of a conversation is an essential requirement for a holistic analysis model. On an abstract level, an online conversation consists of posts (e.g., tweets or comments) and links between these posts that represent semantic relationships (e.g., replying or commenting). The concept of graphs is particularly suitable for modeling objects, in our case posts, and their relationships. We resort to these graphs as the primary modeling approach for the CODY framework. Posts contained in the conversation are represented as nodes and are linked via semantic relationships represented as edges. For a conversation graph with nodes V and edges L to take the different semantic types into account, the network definition comes with a node type mapping $\varphi : V \rightarrow A$ and a link type mapping $\psi : L \rightarrow R$ with A and R denoting the network's node and edge types respectively. This network typing aligns with the concept of HINs.

Definition 5.1 (Structural conversation network)**.** The structural network $G_{struc} = (V, L)$ of a conversation consists of a set of posts modeled as nodes V ($\varphi(v) =$ post with $v \in V$) and a set of links $L \subset V \times V$ among these posts. Each link $l_{ij} = (v_i, v_j)$ is a tuple of the two related posts v_i and v_j.

A single conversation is treated as an individual network based on Definition 5.1. Generally, interpersonal communication modeled by the structural conversation network implies directionality and a post is usually meant as a response to another, e.g., reply → seed post. Therefore, the structural conversation network is a *directed* network. Also, the outgoing degree of a node is one except for the root node, which is not a response to any other post. Therefore, if one would not consider the edges' directionality, two posts would be connected by exactly one path, and the structural conversation network could be described as a *tree* (see Cogan et al. (2012)).

Model reference. The structural conversation network proposed in this section is in line with the network analytics model of Chapter 3. Specifically, if one understands the posts contained in the conversation network as document nodes, the proposed conversation network corresponds to a document network as described in Section 3.2.1 with the post nodes being linked by conversation interactions, e.g., replies.

5.3.2 Temporal conversation model

Conversations are inherently dynamic and evolve over time. It is crucial to consider these dynamics for modeling a conversation. Typically the posts contained in a conversation come with a timestamp, such as the publishing date. Formally, the assignment of timestamps is described by a node time mapping $\pi_t : V \rightarrow \mathbb{T}$ with $\mathbb{T}$ representing the domain of time. Given that the posts' times-

tamps impose a natural order on a conversation's elements, the respective structural conversation network can be split into a temporal sequence of snapshots.

Definition 5.2 (Temporal conversation model)**.** Each post in a structural conversation network comes with a timestamp: $\pi_t(v) = t$. Given a sequence of n timestamps $(t_i)_{i=0}^{n}$ with $t_i < t_{i+1}$, the structural network G_{struc} can be split into n temporal snapshots: $G_{temp} = (G_i)_{i=1}^{n}$. For each snapshot $G_i = (V_i, L_i) \in G_{temp}$ the contained nodes and edges are defined as follows: $V_i = \{v \in V \mid t_0 \leq \pi_t(v) = t < t_i\}$ and $L_i = \{l_{ij} = (v_i, v_j) \in L \mid v_i, v_j \in V_i\}$.

Notably, the model does not require equally sized snapshots, meaning that the condition $|t_{i+1} - t_i| = |t_{i+3} - t_{i+2}|$ does not necessarily hold. Instead, the network G_{struc} might also be sampled according to the volume of conversation posts as an alternative to equally sized time windows. In such a case, the condition $|V_i'| = |\{v \in V \mid t_{i-1} \leq \pi_t(v) = t < t_i\}| = |V_{i+1}'| = |\{v \in V \mid t_i \leq \pi_t(v) = t < t_{i+1}\}|$ holds, meaning that the amount of newly added posts is constant across the network snapshots. This *volume-based sampling* approach is especially suitable in cases where conversation posts are not equally distributed over time. Therefore, a time-based sampling, as suggested by Cogan et al. (2012) and used by Mathew et al. (2016), would lead to a highly skewed distribution of newly added posts across the network snapshots.

Model reference. The temporal conversation model, as proposed in this section, is again coherent with the analytics model leveraged throughout this thesis. First of all, the timestamps of the posts are an example of document attributes as described in Section 3.2.3. Based on these, the temporal conversation network is created in line with the approach taken for the document network snapshots (see Section 3.3.1). The named section also explains the volume-based sampling as applicable in an accumulative snapshot creation setting. The same sampling strategy is leveraged for the CODY model.

5.3.3 Content-aware conversation model

Next to structure and dynamics, the actual content of a conversation must be considered by a conversation model. Various types of semantically meaningful content can be part of a conversation, such as the written text, shared URLs, posted images, or involved users. These content objects are extracted from the analyzed posts to be incorporated into the conversation network. Figure 5.2 illustrates this extraction process in line with Figure 3.2, which visualizes the schema of a document entity network as part of the analytics model proposed in Chapter 3. In the content-aware conversation network, the extracted content objects are linked to their belonging post node.

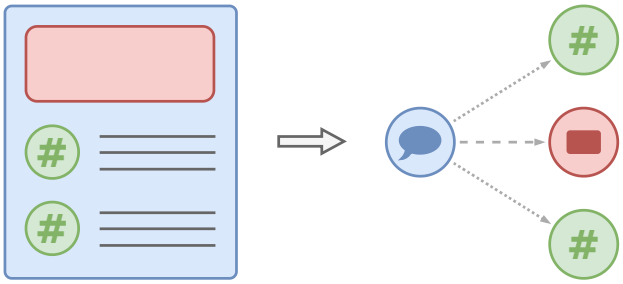


Figure 5.2: Modeled content objects are extracted from the analyzed posts. Extracted content, the two hashtags and the image, is linked to the respective post node in the content-aware conversation network.

Definition 5.3 (Content-aware conversation network)**.** Given G_{struc}, the content-aware conversation model G_{cont} contains additional content objects modeled as nodes of additional types $a \in A$. The actual content objects are extracted from the respective conversation posts: $\epsilon : V_P \rightarrow 2^{V_C}$ with $V_P = \{v \in V \mid \varphi(v) = \text{post}\}$ and $V_C = \{v \in V \mid \varphi(v) \in A \backslash \{\text{post}\}\}$. To indicate the extraction process, in G_{cont} the content objects are linked to the original post node(s): $v_p \rightarrow v_c$ with $v_c \in V_C$ and $v_c \in \epsilon(v_p)$. These links might be typed. Respective types are modeled as additional edge types $r \in R$.

The same content object might belong to different posts in the content-aware conversation network, e.g., the same URL might be shared in multiple posts. Therefore, two formerly disconnected post nodes might become indirectly linked via shared content nodes (e.g., URLs, hashtags, usernames).

Model reference. The definition of the content-aware conversation network follows the definition of an entity network as described in Section 3.2.2. The content objects are linked to their original post node in the content-aware conversation network as entities are linked to their original document node in the entity network.

The content enrichment of the structural conversation network also applies to the individual snapshots of the temporal conversation network G_{temp}. Each network snapshot then contains the content objects belonging to the contained posts. Together, the temporal and content-aware conversation model incorporates a conversation's structure, dynamics, and content. Formally, the CODY model can be seen as a THIN.

5.3.4 Temporal Wiener index

To quantitatively describe the structure of conversation networks, tailored metrics are needed. In a recent work, the Wiener index, known from the field of chemistry (Wiener, 1947), is leveraged to measure the structure of reply trees (Saveski et al., 2021). According to Saveski et al. (2021), the Wiener index describes two characteristic topologies of a tree-like graph in its extreme cases. On the one hand, a low Wiener index indicates a conversation in which all non-seed posts respond to the original seed post, and these non-seed posts do not show any interactions among each other. On the other hand, a high Wiener index indicates a conversation tree with a single dominant interaction branch. To the best of our knowledge, no work extends the Wiener index to fit temporally evolving conversation trees. Based on the definition of the static Wiener index employed by Saveski et al. (2021), we define the *temporal* Wiener index as formalized in the following:

Definition 5.4 (Temporal Wiener index)**.** Given a temporal sequence $G_{temp} = (G_i)_{i=1}^{n}$ of tree-like structured conversation networks, its temporal Wiener index $\mathbb{w}(G_{temp})$ is determined by the average distance between post nodes in each network snapshot:

$$\mathbb{w}(G_{temp}) = \left(\frac{1}{|V_i| \cdot (|V_i| - 1)} \sum_{m \in V_i} \sum_{n \in V_i} \delta_i(m, n) \right)_{i=1}^{n} \tag{5.1}$$

with $\delta_i(m, n)$ denoting the shortest path length between the nodes m and n in the temporal snapshot G_i.

Given a temporal sequence of conversation networks, the temporal Wiener index thus itself is given as a temporal sequence of static Wiener indices. Therefore, the time series of Wiener indices allows for studying the temporal changes of the Wiener index across the temporal evolution of a conversation.

5.4 Experiments

This section discusses various experiments and their results to demonstrate the capabilities of the CODY model. A Twitter conversation dataset that we collected is leveraged for the conducted experiments. It is described in detail in Section 5.4.1. Subsequently, in Section 5.4.2, the posting activity across the lifetime of conversations is analyzed and an empirical model is derived. Further, Section 5.4.3 focuses on investigating the structural evolution of conversations. For this, the temporal Wiener index is leveraged, and its dynamics are studied. Finally, in Section 5.4.4, the

dynamics of the content discussed in the conversations are analyzed. Specifically, the evolution of hashtag prevalence is studied, and several hashtag hijacking scenarios are examined.

5.4.1 Twitter conversation dataset

The dataset used for the subsequent experiments is based on a collection of German political Twitter accounts published as EPINetz Twitter dataset (König et al., 2022). We collected conversations that started from a seed post of one of these accounts by leveraging the official Twitter search API v2. According to Maireder and Ausserhofer, the political domain is particularly suitable for examining the structure of conversations (Maireder and Ausserhofer (2014), as cited in Koylu (2019)): "Retweets and mention/reply functions have been found to influence the structure of conversational discourse if it is especially related to politics." In total, the dataset consists of more than 2 M tweets as part of 5 k conversations, with each conversation including at least 50 posts. Figure 5.3 shows the distribution of conversation sizes. The tweets were published between December 2020 and January 2023, and the median conversation duration in the dataset is about seven days.

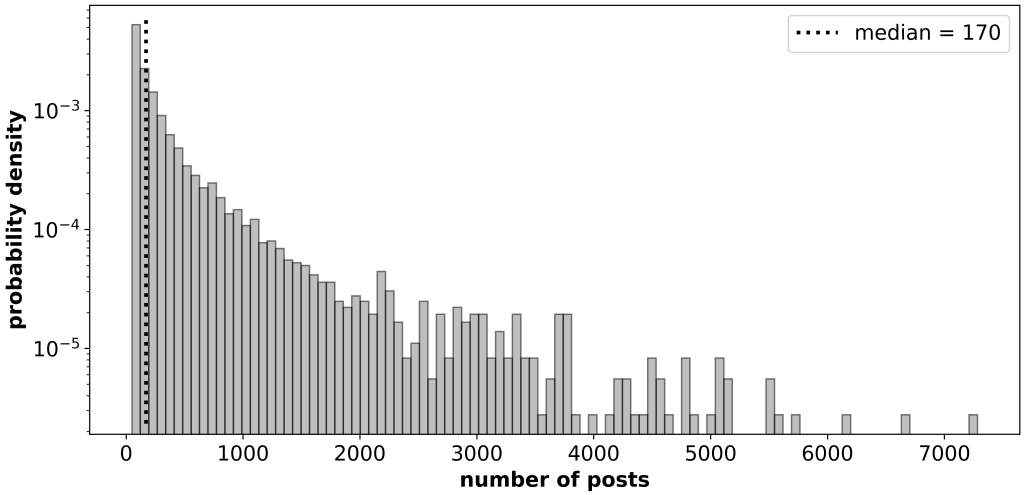


Figure 5.3: Distribution of conversation sizes as contained in the leveraged dataset of the German political Twittersphere

According to Sun and Han (2013), the Twitter information network is well suited to be modeled as HIN. In line with the CODY model, we view each conversation as a temporal and content-aware conversation network with the contained tweets as posts and hashtags as content objects. Figure 5.4 illustrates the network schema of the conversation model used for the subsequent experiments. In the analyzed networks, links between tweets indicate replies which are seen as "[...] the most

direct communication sign between two users" (Cogan et al., 2012) and therefore, make up the structure of a conversation.

Regarding data quality, there is no guarantee that the complete lifetime of a conversation is captured. This would require a continued re-crawl of the conversation data, which is unfeasible due to API usage restrictions. Nevertheless, the reported high average conversation duration suggests that most of the conversation's evolution is captured. This assumption is also underlined by the conducted experiment, which examines the posting activity of the collected conversations. Most activities seem to occur within the first 10 % of a conversation's lifetime. Therefore, sufficient posts should be contained in the dataset even for shorter conversations, for which data of less than a day is captured. In their analysis, Cogan et al. (2012) state that conversation activity usually stops after 6 hours. Further, additional data quality issues related to online conversations have been reported in the past, such as replies to private profiles or the deletion of posts (Cogan et al., 2012). However, less than 5 % of conversations seem to be affected by that. Missing posts lead to a split of the conversation graph into multiple components. Therefore, we take the weakly connected, giant component for creating the reply trees to handle the cases in which already deleted posts or those that link to private profiles are part of a conversation. Also, in a prior step to generate these reply trees, tweets are grouped into conversations based on the `conversation_id` field contained in the raw API payload. Among the tweets of a conversation, reply links are determined by the additional `referenced_tweets` field and the contained entries of type `replied_to`.

Even though the following experiments are conducted based on the described dataset, we argue that the CODY framework is applicable to a wide range of use cases and conversation data of platforms different from Twitter could be used as well. The primary assumption of the model, which is to have interrelated and timestamped conversation posts, is fulfilled by a large variety of online conversation data sources.

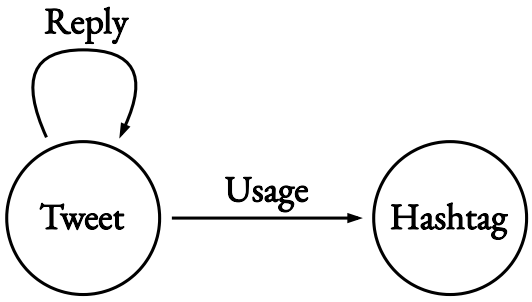


Figure 5.4: Network schema used for the conducted experiments. A conversation's reply tree consist of tweet nodes connected by replies. Further, the "usage" relationship indicates that a hashtag is used within a tweet.

5.4.2 Posting activity

Analyzing the posting activity across the lifetime of a conversation is crucial for a more general understanding of conversation dynamics (see RQ1). It lays the foundation for deciding which temporal sampling approach must be leveraged to model the conversation dynamics correctly. Figure 5.5 shows the volume completion rate (relative number of posts) of the conversations in our dataset plotted against the conversations' lifetime (time completion rate). The dark and lighter value bands indicate mean values and their standard deviations. Values are calculated using a 10^{-5} resolution of the completion rate. The large fluctuations show that temporal posting activity varies a lot between conversations. Still, a general activity pattern across the lifetime of a conversation can be observed. If λ_{volume} denotes the volume completion rate of a conversation, which can also be described as the fraction of the total amount of conversation posts, and λ_{time} denotes the time completion rate (lifetime) of a conversation, a conversation's posting activity can be modeled as

$$\lambda_{volume} = 1 - e^{-\alpha\lambda_{time}} \tag{5.2}$$

with α as the saturation rate. As shown by the fitted function in Figure 5.5 and indicated by a reduced chi-square value, χ^2_ν, of 0.4, this model, despite its simplicity, describes the process in a precise manner. For the curve fitting, the `lmfit` Python package is used (Newville et al., 2016). Most of the posting activity happens during the early stage of a conversation. After an exponential increase in conversation posts, the posting activity slows down and converges towards its final saturation level. This saturating process is also characterized by the saturation time constant Γ, defined as $\Gamma = \frac{1}{\alpha}$. For our dataset, the saturation time is about 0.03, meaning that on average, after 3 % of the conversation lifetime, about $1 - \frac{1}{e} \approx 0.63 \mathrel{\hat{=}} 63\,\%$ of the posts are published.

The non-linear activity pattern must be considered when selecting the sampling method for the temporal modeling of conversations. Given the exponential increase in posts during the early stage of a conversation, a sampling approach using fixed time windows would lead to a highly skewed distribution of newly added posts across the network snapshots of a conversation. In the beginning, the network snapshots would grow significantly, whereas in later stages, only minor changes would be captured. Quantitatively the change rate of sampling snapshots can be described by the derivative of the posting activity:

$$\lambda'_{volume} = \frac{d}{d\lambda_{time}}\lambda_{volume} = \alpha e^{-\alpha\lambda_{time}}. \tag{5.3}$$

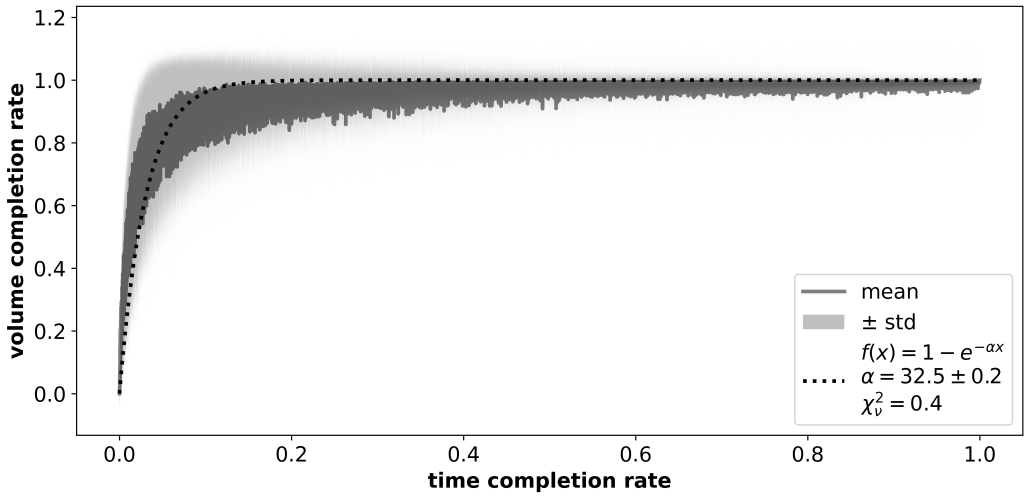


Figure 5.5: The evolution of a conversation regarding the number of posts is characterized by an exponential saturation function.

Therefore, the ratio of snapshot change rates during two points in time λ_{time_1} and λ_{time_2} of a conversation's lifetime is given as:

$$\frac{\lambda'_{volume}(\lambda_{time_1})}{\lambda'_{volume}(\lambda_{time_2})} = e^{\alpha(\lambda_{time_2} - \lambda_{time_1})}. \tag{5.4}$$

For our conversation activity model ($\alpha = 32.5$), about 131 times more posts would be added to a snapshot right at the beginning of a conversation ($\lambda_{time_1} = 0.05$) compared to a later stage ($\lambda_{time_2} = 0.2$) by using fixed time windows. Therefore, to correctly capture changes across the entire lifetime of a conversation, the exponential saturation regarding the posting activity needs to be considered. This can easily be done using fixed volume sampling instead of fixed time windows. In line with the accumulative snapshot creation proposed for the network analytics model in Section 3.3.1, the conversation size difference between two successive snapshots is kept constant for this. In our case, we determine a sampling rate of five new posts per snapshot. This sampling approach is also used for the subsequent experiments. Given that the relative volume of posted tweets can be seen as *normalized* conversation time, we use a conversation's completion rate as the unit for the temporal dimension whenever appropriate.

5.4.3 Temporal Wiener index

Modeling conversations as networks allows us to analyze the conversations' topological structure and by leveraging the *temporal* Wiener index, this can also be done by considering the conversation's dynamics (see RQ2). For the following analysis, we calculate the temporal Wiener indices for the conversations contained in our dataset. Only their reply trees are considered without additional content objects (i.e., hashtags) to comply with the definition of the Wiener index metric. For the software implementation, the `igraph` Python package is used (Csárdi and Nepusz, 2006). To handle the edge case of a single-node reply tree, we define the Wiener index as zero at this conversation state.

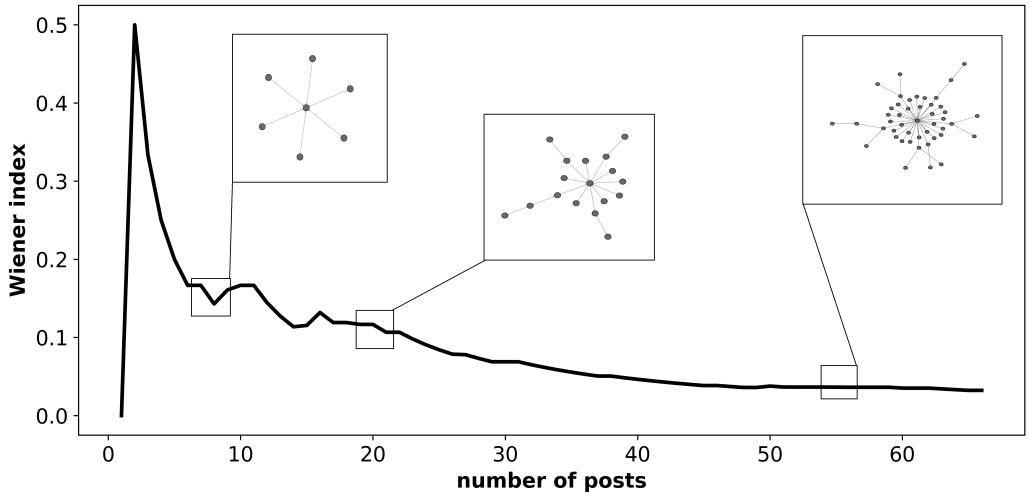


Figure 5.6: Exemplary temporal Wiener index. After a steep increase at the beginning of the conversation, the value rapidly decreases. Structural changes in the conversation also reflect this evolution. Over time most posts directly link to the original seed post, and offshoots are rarely observed.

Figure 5.6 shows a temporal Wiener index example. As one can observe, the structural evolution of the network is also reflected by the temporal change of the Wiener index. The initial rise of the Wiener index value is caused by the first reply, which directly links to the original seed post. Given that only one shortest path of length one is contained in the network at this stage, namely from the first reply post to the original post, a maximum Wiener index value of 0.5 is derived for this network snapshot. From there on, most posts are directly linked to the seed post, reflected by the rapid decrease of the temporal Wiener index. Overall, offshooting branches are rarely observed in the conversation network, which explains its low final value of about 0.03.

To not just focus on a single conversation example but to analyze the structural evolution of conversations in a more general way, the average temporal Wiener index across the lifetime of a

conversation is shown in Figure 5.7 in the form of a box plot. First, it should be noted that temporal Wiener index values vary a lot between the different conversations, as shown by the large quartile ranges, the high variability, and the outliers. Still, a general evolution pattern can be observed. Over time, the Wiener index decreases and converges towards a final median value of about 0.01. Structurally, this means that over time, most replies directly link to the original seed post, and sprawling conversation branches do not seem to play a significant role in the conversation network. Nevertheless, to return to the high data variability, in rare cases, offshoots might be observed in the conversation structure, reflected by the higher outlier values as shown in the box plot. Interestingly, Cogan et al. (2012) report that more than 60 % of their analyzed Twitter conversation graphs follow the structure of paths. Apart from the described outliers, this structure is less prevalent in our case as it would lead to significantly higher final Wiener index values. Many reasons are conceivable for this structural difference. One might be that, in our case, most conversation participants wanted to interact directly with the political actor who posted the original seed post. Thus, more star-like conversation structures are observed.

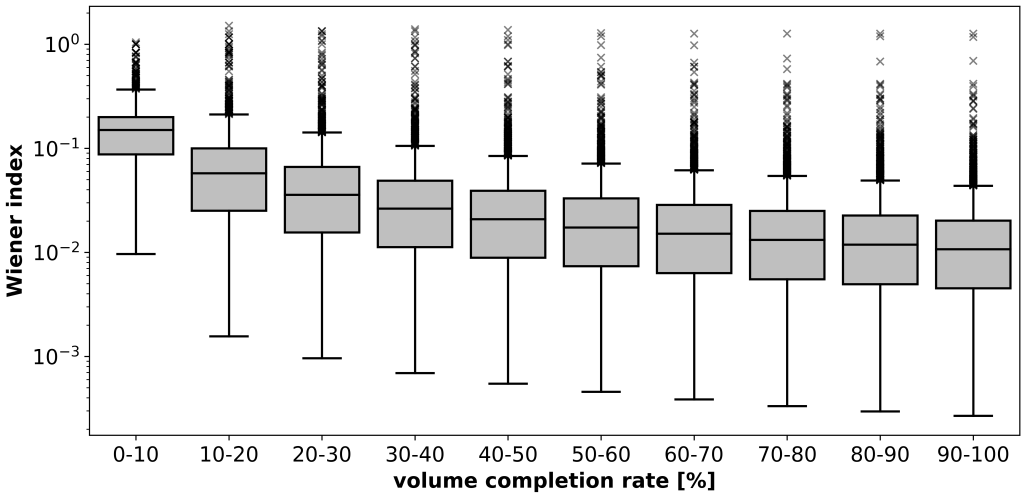


Figure 5.7: Box plot showing the averaged temporal evolution of Wiener index values across the lifetime of the analyzed conversations. Decreasing values indicate the tendency of conversations to become less sprawling.

5.4.4 Hashtag hijacking

In a holistic conversation analysis framework, the discussed content and its dynamics should also be considered next to the structure of the conversation. On the Twitter platform, used hashtags are an indicator of the discussed topics and the semantics of posted content. In that sense, hashtags can be leveraged to analyze how the topical focus of a conversation changes over time. More specifically,

one can investigate whether initially used hashtags keep their dominant prevalence over the lifetime of a conversation (see RQ3). For this, we leverage the proposed Twitter conversation dataset and, given that hashtags are represented as nodes in the content-aware conversation network, we temporally track their degree centrality as an indicator of the importance they play in the conversation.

In the second step, the temporal degree evolution of the hashtag(s) used in the initial seed post is compared to the evolution of the other hashtags used later in the conversation. Suppose a non-initial hashtag reaches a higher degree centrality value than the initially used hashtag(s) at the end of a conversation we denote the scenario as *hashtag hijacking*. In case of an overtake that is followed by a "retake" of an initial hashtag so that it reclaims the highest prevalence with regards to its degree centrality in the conversation network at the conversation's ending, the scenario is called *failed hijacking*. It is not a hijacking scenario if no non-initial hashtags are used in a conversation, or no overtake happens. If multiple hashtags take over the initial hashtag(s), the scenario is only counted as a single hijacking. Semantically, a hashtag hijacking scenario can be seen as an indicator of a *topical shift* within a conversation. Nevertheless, it has to be noted that at this point, we do not employ any NLP techniques to analyze the semantic similarity of the prevalent hashtags in a conversation and whether these are used synonymously. Takeovers might as well be performed by semantically similar hashtags, and therefore, only a slight semantic shift would be caused. An extension to overcome this shortcoming would be a valuable future contribution.

To apply the methodology, we focus on conversations where the initial seed post contains at least one hashtag. Also, conversations should have at least one hashtag that is used a sufficient number of times. In our case, we set this threshold to five usages in a conversation. About 1.6 k of the 5 k conversations in the analyzed Twitter corpus fulfil the outlined requirements. In total, about 1 k hijacking scenarios (~ 67 %) are detected in these conversations. Further, 111 failed hijacking scenarios (~ 7 %) and 427 scenarios without hijacking (~ 26 %) are found. Compared to the total number of conversations in the dataset, successful hashtag hijackings comprise a considerable portion of about 22 %.

As an example of a hijacking scenario, Figure 5.8 shows the temporal degree centrality of three hashtags used in the related Twitter conversation[1]. The original tweet of the conversation criticizes a climate protest by the "Letzte Generation" (Engl. "Last Generation") movement, in which activists glued themselves to the tarmac of an airport in order to interrupt flight operations. The *Klimaschutz* (Engl. climate protection) hashtag is used in this tweet. Interestingly, the initial thematic positioning of the tweet is overtaken. This takeover is reflected by the hashtag hijacking

[1]Volker Wissing on Twitter: [...] / Twitter: https://twitter.com/Wissing/status/1596101821994606592 (accessed 2023-07-10)

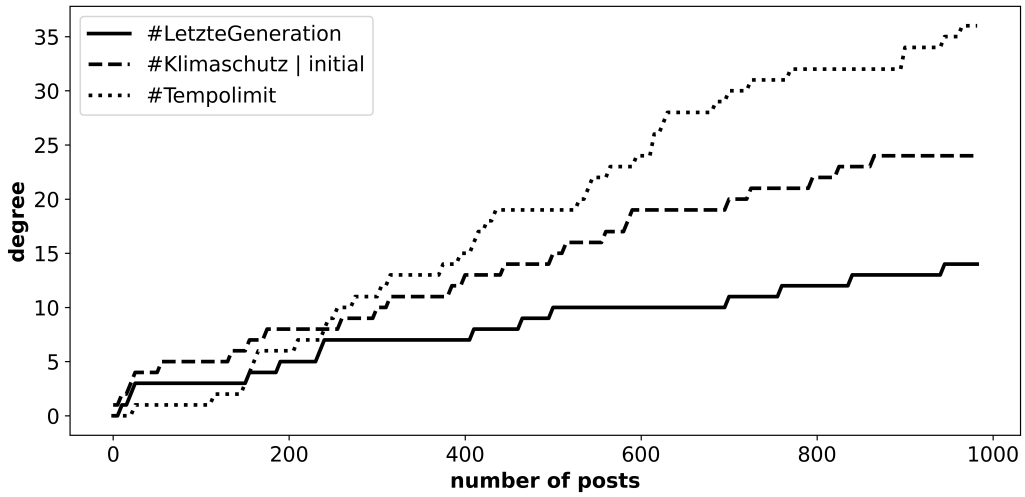


Figure 5.8: Example of a hashtag hijacking scenario. The initial hashtag *Klimaschutz* (Engl. climate protection) is overtaken by the hashtag *Tempolimit* (Engl. speed limit) in terms of degree centrality.

of the *Tempolimit* (Engl. speed limit) hashtag, which represents the discussion about a speed limit on German autobahns. In this case, the hijacking can be understood as a counter-positioning of many conversation participants. Alternatively, their intention might have been to relate this conversation to another climate-related conversation.

Apart from a hashtag hijacking scenario occurring within a conversation, some use cases might also benefit from an analysis of *when* the takeovers happen. Figure 5.9 shows a heatmap of the occurrence probability of the takeovers related to the (failed) hijacking scenarios across the lifetime of a conversation. Dark patches indicate a high takeover occurrence probability, whereas bright ones indicate the opposite. All heatmap rows are normalized based on the total count of occurrences of the respective takeover. As timestamps of the occurrences, we take, on the one hand, the first point at which a takeover is happening by the finally dominating hashtag. For both scenarios, this is when a non-initial hashtag takes over the initial hashtag(s). On the other hand, for the failed hijacking scenario, we also keep track of the last time a "retake" of one of the initial hashtags occurs. At this point, one of the initial hashtags is reclaiming its position as the most prevalent hashtag in the conversation. As shown by Figure 5.9, many initial takeovers happen early in the conversations. A slight shift towards the end of the conversation can be observed for the failed hijacking scenario. Nevertheless, the "competition" for the most prevalent position is not finished at this point. Towards the end of the conversations, most retakes occur.

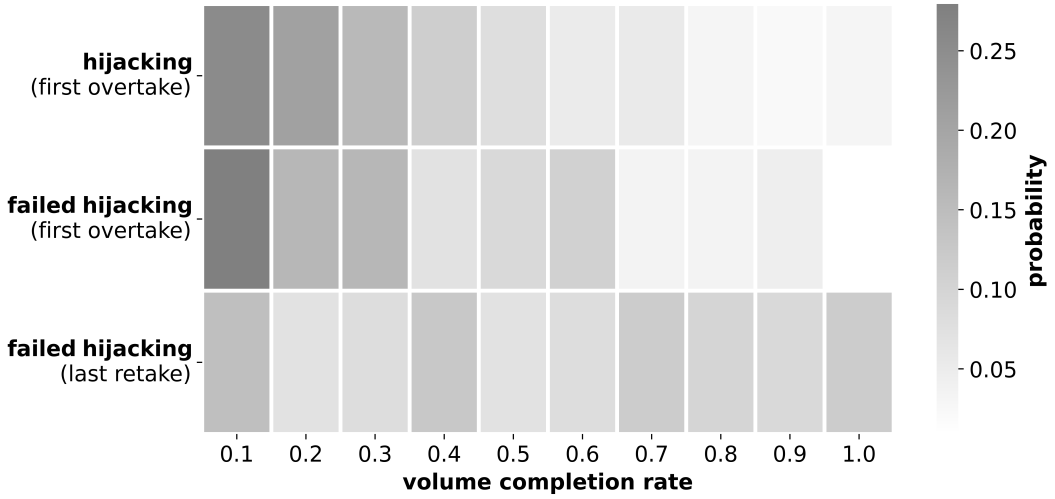


Figure 5.9: Initial hashtag overtakes in hijacking scenarios mainly happen at the beginning of a conversation. Unlike in successful scenarios, in failed ones and for retakes, the overtake tends to occur slightly later in the conversation.

5.5 Conclusion and Future Work

Existing work that focuses on analyzing conversations in online social networks only partially considers the three dimensions of a conversation: structure, content, and dynamics (see Section 5.2). To address this shortcoming, the CODY model is presented in this chapter. Based on the scheme of THINs and in line with the concepts proposed in Chapter 3, it allows one to model a conversation's topological structure, incorporate various content objects, and consider its temporal evolution. The results of several experiments are presented to demonstrate the model's capabilities. A conversation dataset of the German political Twittersphere is leveraged for these experiments (see Section 5.4.1). Still, the conducted experiments are not tight to this dataset but apply to an extensive range of data sources. The proposed CODY framework can be used for various use cases and conversation data sets. Only the primary assumption of the model, which is to have interrelated and timestamped conversation posts, needs to be fulfilled for the model to be applicable.

Regarding general applicability, the CODY model also serves as an exemplary use case of the network-based analytics model proposed in Chapter 3, which highlights the latter model's versatile usage possibilities and broad applicability. In summary, the CODY model follows the concept of temporal entity snapshot networks by referring to the terminology of the proposed analytics model.

Furthermore, in the first experiment, the posting activity across the lifetime of a conversation is examined. Modeling attempts show that an exponential saturation pattern characterizes the ac-

tivity (see Section 5.4.2). This empirical model is leveraged to develop an appropriate sampling technique for modeling a conversation's dynamics based on temporal network snapshots. Instead of commonly used fixed time windows, the sampling is conducted based on a fixed growth of the snapshots to avoid a highly skewed distribution of changes in network size. Further, a novel metric, the temporal Wiener index, is introduced to characterize the structural evolution of conversations. Analyses based on the leveraged Twitter dataset show that with the progression of a conversation, its structure tends to be less sprawling and more centered around the original seed post (see Section 5.4.3). Finally, the content of conversations is analyzed by investigating the prevalence of hashtags over the lifetime of a conversation. In particular, we investigate whether initially used hashtags kept their dominance over time. The results show that this is not necessarily the case. Instead, what we call "hashtag hijacking" scenarios are observed in which non-initial hashtags gain more importance in a conversation than initially used ones (see Section 5.4.4).

Of course, the conducted experiments are not exhaustive but may be extended in several directions. In this regard, future work might consider additional content objects, such as shared images or videos. Analyzing these multimedia objects might lead to an even better semantic understanding of the analyzed conversations. Along with that, more advanced NLP methods might also be employed, for example, to analyze the sentiment within a conversation or to investigate the hashtag hijacking scenarios from a semantic perspective. Also, one might frame the hashtag hijacking phenomenon as a prediction task and develop a method to predict hashtag hijacking scenarios or, in the case of the temporal Wiener index, how a conversation structurally evolves. In this sense, it is also of interest whether a conversation's structural and content-wise evolution depend on each other and if they are correlated in some way.

6 Realization and Application

In contrast to the other chapters in this work, this one aims to provide insights into the technical applications needed to put discussed theoretical approaches into practice. For that, the following chapter consists of four main parts. First, a temporal network media dataset is presented in detail in Section 6.1. Thereby, a large emphasis is placed on the dataset's statistics and covered dynamics. Secondly, the EPINetz platform is presented in Section 6.2. The platform aims to provide access to the mentioned media dataset. For that, the EPINetz platform is implemented as a web application, allowing interested users to explore the collected media dataset interactively. Based on the modeling approach presented in Chapter 3, these exploration capabilities also include graph analytics methods. To provide such capabilities, it is necessary to manage the underlying data appropriately and to provide performant query interfaces, requiring an efficient (temporal) graph data management and analytics system. Finding such an appropriate system is not straightforward, and given the use case-specific requirements, no publicly available benchmark can be leveraged. Therefore, our own use case-specific benchmark is conducted, and its results are presented in Section 6.3. A system that best fits the technical requirements is found by testing various data management systems and different configuration setups for multiple dataset sizes and graph queries. Finally, the TrendTracker web application for the interactive exploration of social media trends is presented in Section 6.4. It serves as an example of how temporal and network-based data exploration capabilities might be leveraged to facilitate the analysis of temporally evolving and highly interrelated data.

6.1 Dataset

This section examines the dataset that lays the foundation for various analyses conducted throughout this work. For example, the study on long-term trends, as presented in Section 4.1, is based on this dataset. Also, the EPINetz platform outlined in Section 6.2 is built on top of this data. Thereby, it not only enables showcasing the capabilities of the temporal, network-based model of Chapter 3 but also contributes towards the development of a comprehensive media analytics system. The longitudinal and broad media coverage of the German political domain makes it a

unique dataset. In general, it consists of data from two sources: Twitter and news articles. The part of the dataset that comes from the Twitter platform is examined in detail in Section 6.1.1, whereas the news article subset is discussed in Section 6.1.2. The textual data from these two sources is also analyzed using natural language processing methods. Section 6.1.3 describes the relevant parts of this processing step and gives statistics on the extracted data. For the EPINetz platform, a subset of both data sources is used. This EPINetz dataset is presented in Section 6.1.5. Further, the collected data is modeled as a temporal heterogeneous information network in line with the analysis model proposed in Chapter 3. Therefore, specific network-related information about the datasets is given in Section 6.1.4 as well as for the EPINetz subset in the according Section 6.1.5. The statistics presented in this section are based on the state of the dataset from 11 July 2023.

6.1.1 Twitter data

A large portion of the dataset is made up of tweets that are collected from the Twitter platform via the official Twitter API v2. An example of how the raw tweet data, as returned by the API, looks is given in Section 3.2.4. In total, the dataset contains about 82 M tweets and profile information of about 1.1 M Twitter users. About 1.7 M individual Twitter users posted these tweets. Therefore, not for every tweet author, the respective profile information is also contained in the dataset. To a large degree, the tweets are related to user accounts owned by (German) politicians, lobbyists, news outlets, broadcasting companies, or journalists. To show some exemplary tweets, Table 6.1 ranks the top 3 most liked tweets as contained in the dataset. With the third place being held by the tweet of Barack Obama[1], one politician is also represented in this list.

Table 6.1: Top 3 most liked tweets. The publishing date is given in UTC.

Username	**Text**	**Publishing date**	**Likes**
chadwickboseman	https://t.co/aZ2JzDf5ai	2020-08-29 02:11:50	7,345,887
elonmusk	Next I'm buying Coca-Cola to put the cocaine back in	2022-04-28 00:56:58	4,666,655
BarackObama	"No one is born hating another person because of the color of his skin or his background or his religion..." https://t.co/InZ58zkoAm	2017-08-13 00:06:09	3,953,368

[1] Barack Obama – Wikipedia: https://de.wikipedia.org/wiki/Barack_Obama (accessed: 2023-07-13)

Furthermore, Figure 6.1 shows the number of published tweets over the entire time span covered by the dataset. The oldest tweets contained in the dataset were published at the beginning of 2006. From there on, newly published tweets were added to the dataset until the beginning of 2023, when Twitter severely restricted the free access to its API (Dotson, 2023). Most of the tweets were published between 2020 and 2023. The peak in tweet volume at the beginning of 2020 is most likely related to the outbreak of the COVID-19 pandemic[2] when many more tweets were collected.

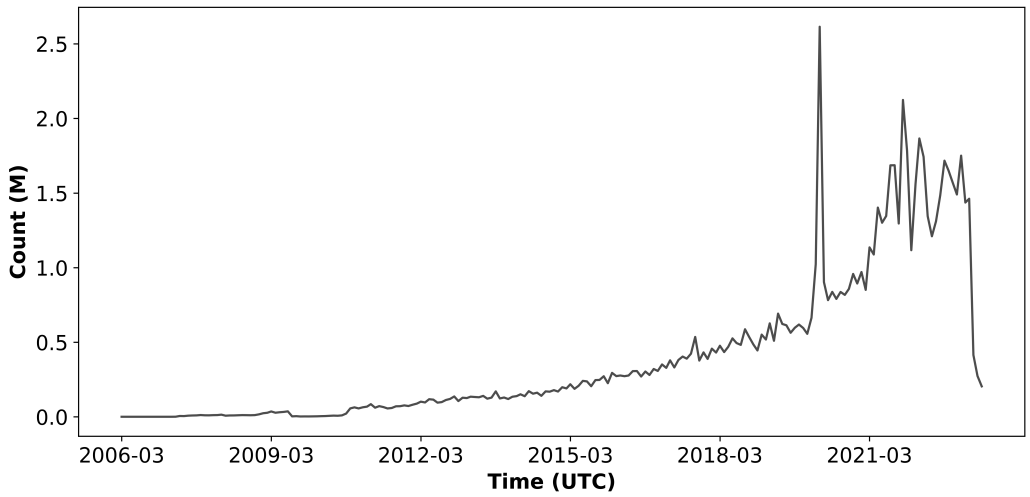


Figure 6.1: Amount of tweets over time. Tweets in the dataset were published between the beginning of 2006 and 2023, with most tweets falling into the time range of 2020 until 2023.

Not only direct tweets are contained in the dataset but also retweets. As mentioned, some tweets are not directly linked to any user for which the respective profile information is collected. These tweets have mainly been collected to cover a broader conversation (i.e., including the retweets) or to cover specific topics (e.g., the COVID-19 pandemic). Tweets related to a topic are selected based on the used hashtags, which must be relevant to the topic (e.g., *#covid19*). In all the collected tweets, about 1.4 M individual hashtags are occurring. Table 6.2 shows the dataset's top 10 most occurring hashtags. One can see that a lot of the top hashtags are related to the COVID-19 pandemic: *corona*, *covid19*, *coronavirus*, *covid*, *maskenpflicht* (Engl. mask duty) and *lockdown*. Also, the remaining "non-covid" hashtags are related to German politics: *afd* (German party), *berlin* (capital of Germany and the place of the German parliament), *spd* (German party) and *cdu* (German party).

[2]COVID-19 pandemic – Wikipedia: https://en.wikipedia.org/wiki/COVID-19_pandemic (accessed 2023-07-12)

Table 6.2: Top 10 most occurring hashtags

Hashtag	**Occurrence count**
corona	3,151,309
covid19	1,572,382
coronavirus	1,268,890
afd	739,344
berlin	441,468
spd	379,993
covid	367,262
maskenpflicht	333,193
cdu	326,108
lockdown	314,056

Next to topic-related statistics, like the top most used hashtags, it is also interesting to examine the dataset from a more actor-focused perspective. Following this direction, Table 6.3 shows the top 10 Twitter users that published the most tweets as contained in the dataset. These accounts again indicate the dataset's centering around the German (political) media sphere. Of the 10 Twitter accounts shown in Table 6.3, seven represent German news organizations: *na_presseportal*, *welt*, *FAZ_NET*, *BILD*, *Tagesspiegel*, *dwnews* and *MDRAktuell*.

Table 6.3: Top 10 most active Twitter accounts according to posted tweets

Username	**Number of tweets**
na_presseportal	654,285
welt	577,142
FAZ_NET	495,955
BILD	359,883
Tagesspiegel	324,767
citoyenneFrance	324,189
dwnews	318,202
HolgerEwald1	300,502
BMG_Bund	288,649
MDRAktuell	281,079

In addition to the actively posting actors, one might be interested in which accounts get mentioned the most. Table 6.4 shows the 10 most mentioned Twitter accounts as contained in our dataset. All of the presented accounts are related to the political domain. They either represent German politicians or German parties (*CDU*, *spdde* and *Die_Gruenen*).

Table 6.4: Top 10 most often mentioned Twitter accounts

Username	**Mention count**
Karl_Lauterbach	5,590,294
c_lindner	1,599,117
Markus_Soeder	1,358,947
CDU	991421
SWagenknecht	894,001
Ralf_Stegner	872,628
MarcoBuschmann	816,720
StBrandner	804,947
spdde	789,646
Die_Gruenen	782,977

6.1.2 News data

Next to the Twitter subset, a large amount of news article data is also included in the dataset. These news articles are crawled from publicly accessible news websites. Primarily, news articles from the fields of politics, business, finance, and society are collected. Apart from the text of the article, metadata is also included. This metadata includes the URL of the published article, the language, the title, the authors, when the article was published, and when it was crawled. In total, about 1.8 M news articles are contained in the dataset. Figure 6.2 shows the distribution of crawled news articles over time. The according crawling time is used for the time axis because the publishing date is not always available from the crawled HTML data.

The dataset's news articles were crawled from 12 different news outlet websites, with most of them being mainly prominent in the German news landscape. Also, the German news outlets, such as the Frankfurter Allgemeine Zeitung (*faz.net*) or the Bild (*bild.de*), contribute the most to the number of news articles included in the dataset as shown by Figure 6.3. All of the top seven news outlets are specifically targeted at the German news market.

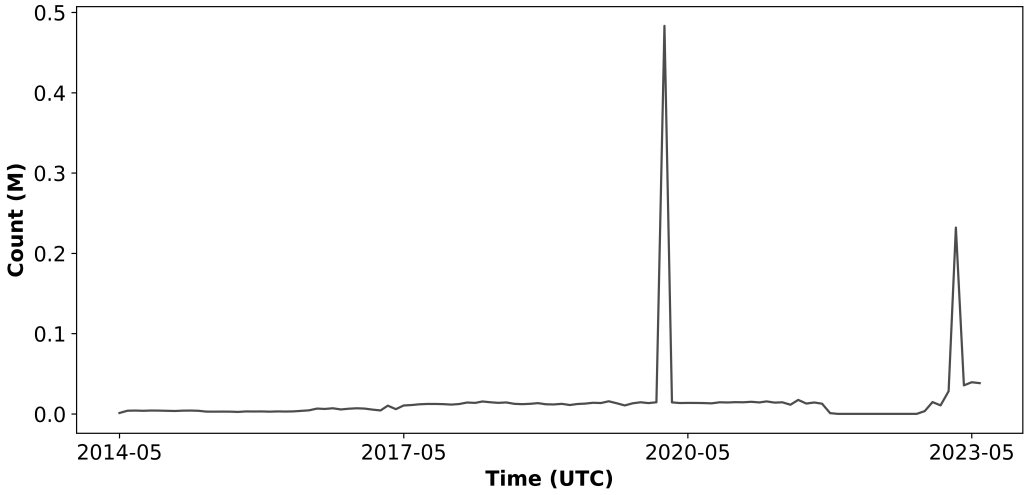


Figure 6.2: Amount of news articles over time. News articles in the dataset were crawled starting from the spring of 2014. The two periods at the beginning of 2020 and 2023 show large peaks in the volume of crawled articles.

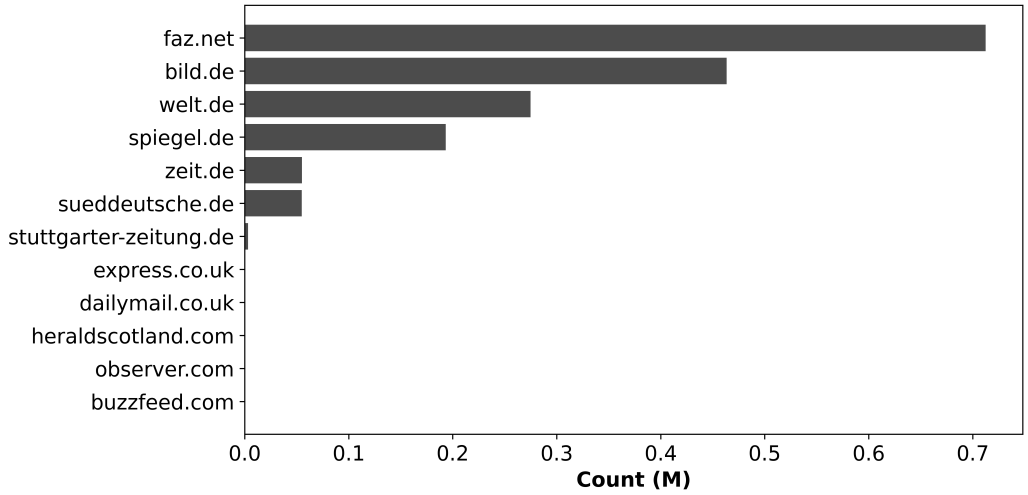


Figure 6.3: Amount of news articles grouped by outlet. German news outlets hold all of the top seven positions.

6.1.3 Natural language processing

The data retrieved from the Twitter API and from crawling the news websites is not kept in its raw form. Using methods known from natural language processing (NLP), the textual data of the collected tweets and the news articles is processed. First of all, individual terms are extracted from the contained text (e.g., *again*, *you* or *project*). In total, about 16 M terms are part of the

extracted vocabulary, meaning that about 16 M individual terms occurred in the dataset's textual data. Secondly, named entities, such as locations, people, or organizations, are extracted. For the extraction process the `spaCy`[3] Python package is used. In sum, more than 6 M named entities are found in the dataset. To improve data quality, the found named entities are also checked to have a corresponding entry in the Wikidata[4] collection. Only these named entities with such an equivalent are considered during the network creation step.

Table 6.5: Top 10 most occurring named entities. Named entities have to be part of Wikidata.

Named entity	**Occurrence count**
Deutschland	2,239,608
SPD	1,127,896
CDU	972,540
EU	884,073
Corona	867,342
Berlin	832,299
FDP	677,822
Ukraine	619,792
Europa	544,532
Russland	515,234

Table 6.5 shows the top 10 named entities that occur the most in the dataset and are also part of the Wikidata collection. Again, one can see the dataset's focus on German politics. Three German parties are contained in the list of the most occurring named entities: *SPD*, *CDU* and *FDP*. Further, the Russian invasion of Ukraine[5] is reflected (*Ukraine* and *Russland*), as well as the COVID-19 pandemic (*Corona*).

6.1.4 Network data

In line with the network-based analytics model examined in Chapter 3, the dataset presented in this section is modeled as a temporal heterogeneous information network. For that, the tweets and the news articles are treated as documents and are used as the basis for the document network described in Section 3.2.1. Various objects are extracted from these documents, such as hashtags

[3] spaCy · Industrial-strength Natural Language Processing in Python: `https://spacy.io` (accessed 2023-07-18)
[4] Wikidata: `https://www.wikidata.org/wiki/Wikidata:Main_Page` (accessed 2023-07-14)
[5] Russian invasion of Ukraine – Wikipedia: `https://en.wikipedia.org/wiki/Russian_invasion_of_Ukraine` (accessed 2023-07-12)

and Twitter usernames for tweets and named entities for news articles. These extracted entities enrich the document network and turn it into an entity network.

Further, to take the dynamics of the dataset into account, the documents' timestamps, meaning the publication date for the tweets and the crawling time for the news articles, are incorporated into the network in the form of document attributes. According to the analytics model, they can also be used within entity projections this way. Adding up the number of tweets and news articles, about 84 M documents are part of the analytics network derived this way. Apart from these document nodes contained in the network, it consists of more than 12 M entity nodes. The entity nodes are of type named entity, term, hashtag, or Twitter user.

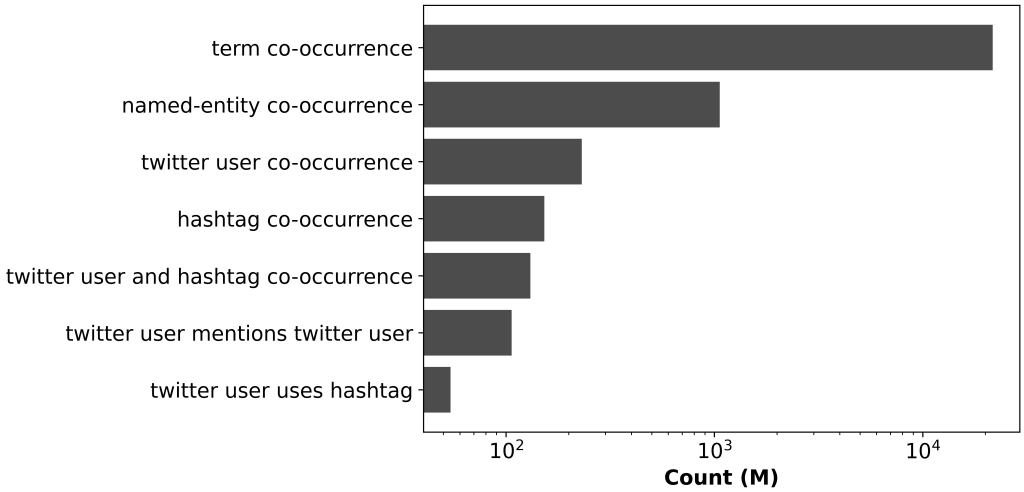


Figure 6.4: Number of edges grouped by their type. Term co-occurrences contribute the most to the number of edges.

Further, the network contains edges among these named entities. In total, more than 23 B of these edges are part of the network. They also have various types, denoted as: "Twitter user mentions Twitter user", "Twitter user co-occurrence", "named-entity co-occurrence", "term co-occurrence", "hashtag co-occurrence", "Twitter user and hashtag co-occurrence", and "Twitter user uses hashtag". Together, the number of nodes and edges in the entity network result in an average degree across all node and edge types of about 1.9 k. As shown in Figure 6.4 by the distribution of edges grouped by their type, most edges are term co-occurrences. Even a small number of text documents can result in huge term co-occurrence networks. Therefore, the large number of co-occurrences and the related high average degree seem plausible.

With regards to the model proposed in Chapter 3, some entities might only be related to the original document. For example, the author of a tweet might simply be connected to the original

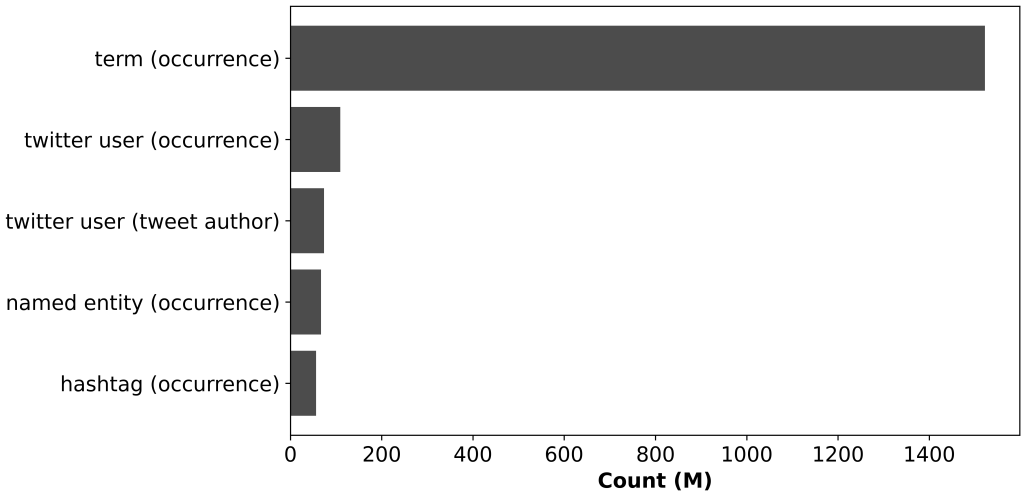


Figure 6.5: Number of node occurrences by type. Term occurrences contribute the most to the number of node occurrences.

tweet node in the entity network. From a model perspective, keeping track of the edges in the document-entity network that do not involve relationships connecting document nodes is necessary. Given that they also come with an implicit timestamp of the related document, we denote them as "node occurrences". The types of node occurrences (document-entity links) contained in the network are: term (occurrence), Twitter user (occurrence), Twitter user (tweet author), named entity (occurrence), and hashtag (occurrence). Figure 6.5 shows the distribution of node occurrences by their type. As described above, one can again see the dominance of the term-related type. By far the most node occurrences are of the term occurrence type. Together about 1.8 B node occurrences are part of the dataset.

Table 6.6: Schema of the network with respect to the dataset's source

	Twitter network	**News network**
Node types	named entity, term, hashtag, Twitter user	named entity
Edge types	Twitter user mentions Twitter user, Twitter user co-occurrence, term co-occurrence, named-entity co-occurrence, hashtag co-occurrence, Twitter user and hashtag co-occurrence, Twitter user uses hashtag	named-entity co-occurrence
Node occurrence types	hashtag (occurrence), Twitter user (occurrence), named entity (occurrence), term (occurrence), Twitter user (tweet author)	named entity (occurrence)

The networks extracted from the Twitter data and from the news data do not have equivalent schemata. Instead, they come with different node (occurrence) and edge types. Table 6.6 gives an overview of these types separated by data source type. As one can see, the Twitter network is much richer and contains more node and edge types than the news network. Also, not each possible node or edge is reflected in the networks. For example, term co-occurrences are not extracted for the news data as they would result in a tremendous amount of data which is infeasible to be stored given the computational restrictions. Statistics about two network subsets related to the Twitter and the news data are discussed in detail next.

6.1.5 EPINetz dataset

For the EPINetz platform, not the complete dataset can be leveraged. Restriction comes into play, especially for the data coming from Twitter. The main concern related to this data is about harmful content such as hate speech or spam. Given that school children are supposed to use the EPINetz application, such content should not be displayed on the platform. Therefore, only direct tweets from politicians are part of the dataset made accessible through the EPINetz platform. One would assume that politicians less likely spread harmful content. For the news articles, no restrictions are applied. Instead, all news articles are used for the analyses provided on the platform. Commonly, news outlets already have content policies in place. Therefore, the news article data should not contain any harmful content.

Table 6.7: Top 10 parties according to the number of tweets posted by their members

Party	**Number of tweets**
Bündnis 90/Die Grünen	2,739,818
SPD	2,171,265
DIE LINKE	1,661,592
CDU	1,249,018
FDP	927,054
AfD	849,451
CSU	285,665
Piratenpartei	80,191
Freie Wähler	54,024
DIE PARTEI	41,203

One needs to have a labeled set of Twitter user accounts to decide which tweets are posted by politicians. Such a dataset is provided by König et al. (2022). According to them, the dataset

contains "[...] Twitter accounts of German parliamentarians, minsters [*sic*], state secretaries, parties, and ministries on a state, federal, and European Union level for the year 2021" (König et al., 2022). Later updates to this dataset are also taken into account. By the 11 July 2023, the labeled dataset contains 2,857 Twitter user accounts. These accounts are used to filter the complete Twitter dataset. As a result, about 11 M tweets are used for the EPINetz platform. These tweets by politicians make up about 14 % of the complete Twitter dataset. Together with the about 1.8 M news, the leveraged EPINetz dataset contains about 13 M documents in total.

Table 6.8: Statistics of the network data used for the EPINetz platform

	Twitter network	**News network**	**Total EPINetz network**
Nodes	1,101,802	632,662	1,734,464
Node occurrences	43,475,540	23,059,723	66,535,263
Edges	118,788,786	1,002,017,118	1,120,805,904
Average degree	≈ 108	≈ 1,584	≈ 646

The politics-related metadata in the Twitter user account dataset provided by König et al. (2022) can be used to derive domain-specific insights. As such, Table 6.7 shows the number of posted tweets grouped by the author's party. The top 10 parties are shown according to their members' posting activity.

Furthermore, not the complete modeled network data described in Section 6.1.4 is contained in the EPINetz subset, given that the Twitter data is filtered based on politicians' tweets. Statistics on the network extracted from these tweets and the news-related network are given in Table 6.8. Term co-occurrences are not contained in the network statistics as these are not used for the EPINetz platform. Given the enormous volume of the term co-occurrence data, ad-hoc queries of this data would result in computational resource requirements that are unfeasible to accomplish within the EPINetz project's framework. Apart from that, one can see from the data shown in Table 6.8 that named-entity co-occurrences of the news network make up the majority of the total network's edges (see Table 6.6). These also cause a sizeable average degree within the news network of about 1.6 k.

6.2 EPINetz Platform

The media dataset presented in Section 6.1 can be used for various use cases, especially in the political science domain. In this regard, one might notice that different societal challenges, such as information overload, emerge due to the digital transformation of the media landscape. This new

environment demands new competencies from citizens that often lack the means to contextualize arguments or actors and to understand their interrelationships in complex topics. The EPINetz project is an approach to bridge the outlined skills gap by developing an appropriate political information system. It provides access to political news collected from multiple data sources, including social media, and offers various network exploration capabilities. Different entities, such as political actors or topics, are extracted from collected data and shown within their respective contexts modeled as weighted and time-varying information networks. Thereby, interested citizens and especially schoolchildren, can discover current political topics and understand relationships between relevant entities.

Reference: This section is based mainly on the following peer-reviewed publication:

John Ziegler, Alexander Brand, Julian Freyberg, Tim König, Wolf Schünemann, Marina Walther, and Michael Gertz. EPINetz: Exploration of Political Information Networks. *INFORMATIK 2021*, pages 1603–1609, 2021.

The remaining part of this section is structured as follows: After covering our contributions in Section 6.2.1, we summarize related projects, contrast them with the EPINetz approach, and outline relevant concepts of digital literacy in Section 6.2.2. This summary is succeeded by a description of the EPINetz platform itself in Section 6.2.3, along with some sample user stories. Finally, we conclude this section with a summary and outlook on the project's roadmap in Subsection 6.2.4.

6.2.1 Contributions

In the EPINetz project we develop a web-based platform[6] that allows users to explore political information networks. Rather than focusing only on a few select media outlets, the platform integrates data from various publicly accessible (German) sources, including Twitter and news outlets. Section 6.1 gives detailed information regarding the leverage dataset, with especially Section 6.1.5 focussing on the data used for the EPINetz web application. Overall, the project has the following objectives:

1. Integrate politically relevant information from different sources and make it accessible in the form of a unified information retrieval system.

2. Extract and visualize entities such as actors (e.g., politicians) and topics, as well as their relationships, in a temporal-sensitive way.

[6] EPINetz Plattform: `https://app.epinetz.de` (accessed 2023-07-19)

The named extraction process incorporates NLP-based methods. These methods allow to map collected documents (e.g., tweets and news articles) to interactively explorable information networks. In these networks, nodes represent entities and edges time-varying and weighted relationships, which aligns with the network-based analytics model examined in Chapter 3. According to the model's terminology, the named temporal entity networks are a type of entity network projection with attributed weights and timestamps. In this regard, the approach followed to develop the EPINetz political information system differs significantly from traditional search engines, where users are simply provided with a list of documents, barely offering any contextualized view.

6.2.2 Background

In this section, we first give an overview of related work, contrast the same with the approach taken in the EPINetz project, and, secondly, outline our understanding of digital literacy.

Related Work

To focus on the platform's core functionality, we narrow the scope of related work to political information systems. In the past, different projects have aimed at building tools similar to the EPINetz platform. These approaches differ from EPINetz in several ways. First and foremost, the Media Cloud[7] project offers a platform for general-purpose media analysis. Despite its diverse and very sophisticated capabilities, such as topic mapping and source management, it does not offer any of the network-based exploration functionality provided by the EPINetz project. This limitation also applies to the Vox Civitas tool (Diakopoulos et al., 2011) designed to analyze social media content around broadcast events. Similarly, the Europe Media Monitor Team (2023) provides basic statistics, trending topics, and activity detection to the end-user but again lacks sufficient exploration functionality. Furthermore, information from social media is not taken into account. This lack of social media incorporation is also true for the TopExNet (Spitz et al., 2019) tool, even though it offers network-based exploration capabilities for entity relationships extracted from news articles. Probably most similar to our work is the LeadLine system (Dou et al., 2012). It offers a visual analytics system for events extracted from news articles and social media. In contrast, EPINetz focuses on the less general domain of political information and, more specifically, targets the German media landscape.

[7] Media Cloud: https://mediacloud.org (accessed 2023-07-25)

Digital Literacy

EPINetz aims to improve both general and domain-specific digital literacy. As to our general conception of digital literacy, there is no canonical terminology to build on, but rather a great variety of concepts used in the field (van Deursen and van Dijk, 2009). The concept of literacy is generally oriented towards comprehensive understandings of citizen education and participation in the digital age. First, our concept emphasizes the informational skills that individual users need to develop and apply a critical understanding of their digital information ecosystems and lifeworlds. Thus, we can build on Jones-Kavalier's and Flannigan's definition of digital literacy (Jones-Kavalier and Flannigan, 2006) as "a person's ability to perform tasks effectively in a digital environment [...] Literacy includes the ability to read and interpret media (text, sound, images), to reproduce data and images through digital manipulation, and to evaluate and apply new knowledge gained from digital environments." Beyond this fundamental concept, recent developments in digitalization need to be reflected. Therefore, we include skills and evaluative capacities concerning datafication, algorithmic filtering, or machine learning that have been termed data literacy elsewhere (Pangrazio and Sefton-Green, 2020). Beyond processing and presenting information in context, EPINetz allows a "look behind the scenes" of datafication by providing opportunities to observe and practice data-scientific methods. All in all, our basic conception is compatible with strategies and frameworks issued by governance actors such as the European Union (Vuorikari et al., 2016) and the German Conference of Education Ministers (Kultusministerkonferenz, 2017) as it takes up key components defined in those frameworks, mostly covering the competence area "information and data literacy", but also citizen engagement and a critical understanding of media in a digital world. As to our domain-specific orientation, it is essential to add that in contrast to many other studies and projects on politics and policy using computational social science methodology, we do not operate in an exclusively data-driven way. EPINetz instead accounts for the pre-structuration of policy fields that resonates in public debates even when mediated in digital environments.

6.2.3 Platform

In the following, we describe the main components of the EPINetz platform. First of all, based on the leveraged dataset described in Section 6.1.5, we outline the data processing in Section "Data processing". Subsequently, in Section "Information networks and timelines", we detail the construction of the different types of information networks for text analysis and exploration purposes. Potential user stories and an explanation of the EPINetz platform's features are given in Section "User stories". To start with, Figure 6.6 serves as an overview of how the different software components interact. Accordingly, the flow diagram underlying the EPINetz application is as follows:

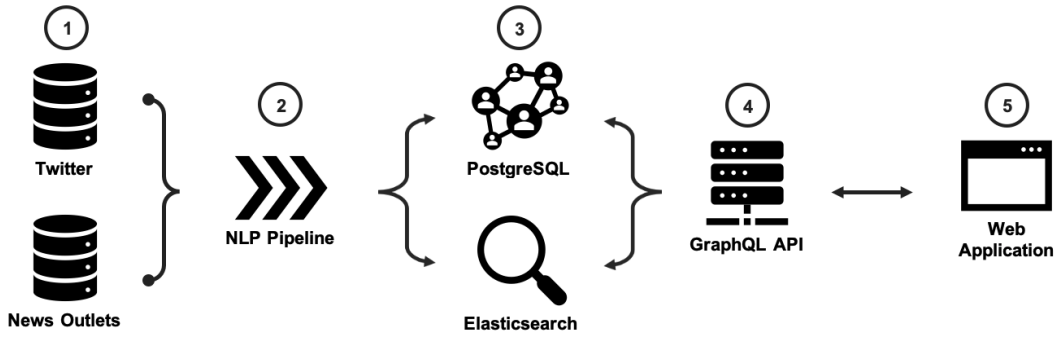


Figure 6.6: Simplified overview of the architecture of the EPINetz platform

1. Data is collected from multiple sources, such as Twitter and news outlets
2. Textual information of the data is processed by the NLP pipeline
3. Data is stored in PostgreSQL for network generation and Elasticsearch[8] for information retrieval
4. A GraphQL[9] API is used as a unifying data access layer
5. Information can be explored by the end-user via the web application

Data processing

Both news articles and tweets are processed via dedicated NLP pipelines, primarily using `spaCy`, for named-entity extraction (e.g., persons, organizations, or locations). For tweets, hashtags and user mentions are also extracted. By now, we refer to topics as densely connected sets of keywords and actors as occurring in our information networks. As illustrated in Figure 6.6, the processed data is managed via two systems, a PostgreSQL database used as the basis for the construction of our information networks and Elasticsearch for different information retrieval tasks.

To distinguish between different policy fields, e.g., public health and migration, we develop "policy parsing" as a novel method for identifying such fields. This method involves a mixture of supervised and unsupervised learning techniques, namely a pre-informed, enriched keyword-based grouping of the tagged dossiers of the *Bundeszentrale für politische Bildung* (Engl. Federal Agency for Civic Education) and a clustering of the node-embedded representations of the mentioned information networks.

[8] Elasticsearch Platform – Find real-time answers at scale | Elastic: `https://www.elastic.co` (accessed 2023-07-18)
[9] GraphQL | A query language for your API: `https://graphql.org` (accessed 2023-07-18)

Information networks and timelines

With fine-grained information about text data accessible in PostgreSQL, different types of information networks and timelines can be constructed. Some of these networks are built as soon as new documents come into the system, while other networks are constructed on the fly in response to user requests issued via the platform interface. Such networks show, for example, the Twitter users mentioned by a user, the hashtags used by a user, or co-occurrences of hashtags, named entities, and keywords. To reflect the temporal dimension, all relationships are assigned time-encoding attributes, enabling the analysis and contrasting of entities and relationships in a time-sensitive manner.

User stories

A fictitious user could leverage the EPINetz platform to keep up with ongoing political debates. By providing an aggregated view based on data from different sources, the end-user less likely suffers from information overload. Statistical information and timelines of the collected data serve as potential entry points to currently prevalent topics. For this use case, on the welcome page of the EPINetz platform, the user is provided with statistics on the dataset and recent trends extracted from the collected documents. These trends serve as a teaser and give the users inspiration on where to start their data exploration journey. Figure 6.7 shows exemplary statistics as shown on the news dataset's welcome page. Both data sources, the tweets and the news articles, have their own welcome page. The user can switch between the different data sources by clicking the button right below the search input field as shown in Figure 6.8. Getting back to the statistics shown in Figure 6.7, one can see that first of all, the user is provided with general dataset-related statistics like the number of documents or a timeline of the number of newly added documents. Further, currently trending entities are also given. By now, these are determined based on the occurrence volume within the past seven days. These trends allow the user to easily track current political discussions and eliminate the need to manually search through a large volume of documents to uncover recent developments in the political media landscape.

In addition to the statistics provided on the welcome pages, the users might want to retrieve data specifically related to an entity they are interested in. Various input types can thereby be used for a search query, such as hashtags, simple terms, or named entities. Not all entity types might be valid depending on the selected dataset. For example, given that, only named entities and terms are contained in the news article documents, appropriate search results will only be returned if the query consists of terms or named entities. For the tweets, the user can also search for hashtags or Twitter users. Also, next to the actual textual query input, the user has the option to restrict

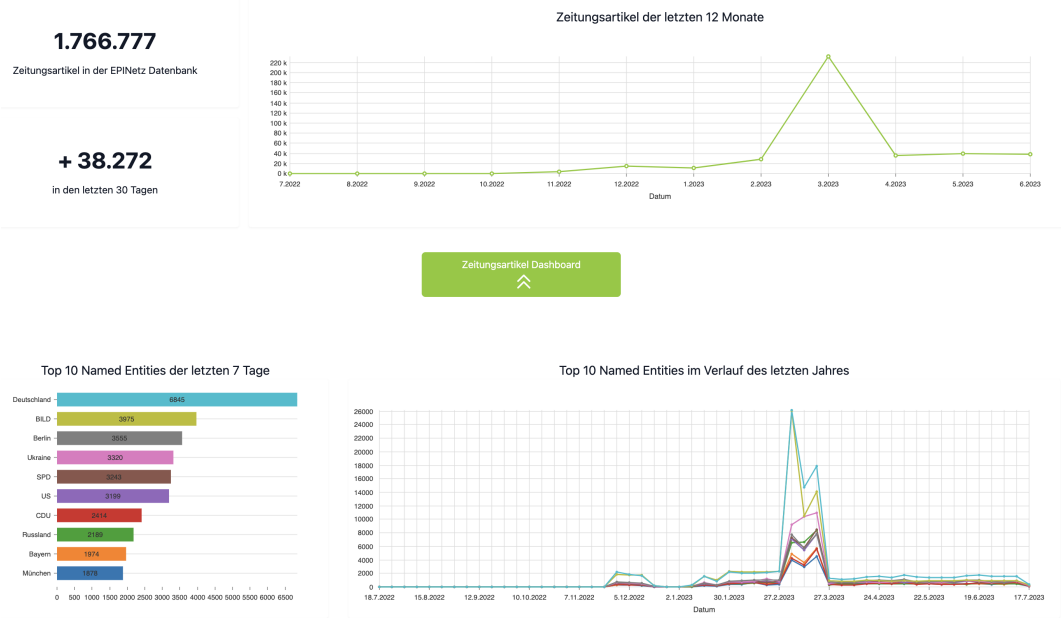


Figure 6.7: On the welcome page, data source-specific statistics are shown to facilitate the start of the user's data exploration process; Screenshot taken from the EPINetz web application on 19 July 2023

the search to a given time window and to select the data source that should be leveraged (tweets or news articles). The search bar shown in Figure 6.8 indicates how the user can adjust these filtering options.

Figures 6.8 and 6.9 show two exploration features of the EPINetz web application. In this case, the *covid19* hashtag related to the COVID-19 pandemic is used as a prototypical search query. First of all, the user is provided with a list of relevant documents as known from popular search engines (see Figure 6.8). Query-related parts of the documents' text are highlighted in the retrieved documents to support the user's visual exploration of the result list. Apart from the text contained in the resulting documents, they are also enriched by additional metadata such as the number of likes and retweets, as well as politics-specific information such as the author's party membership or incumbency state. This metadata is further used to give the user additional filtering options. With these, the user is able to filter the relevant documents based on attributes like the author's political party. Filtering options are displayed next to the search results as shown in Figure 6.8. Additionally, to better understand the temporal dynamics related to the searched entity, the EPINetz web application allows the user to change the chronological order of the retrieved document list.

Furthermore, with the network-based exploration capabilities shown in Figure 6.9, the user is able to understand the context of the analyzed entity better. For that, the searched-for entity is used as

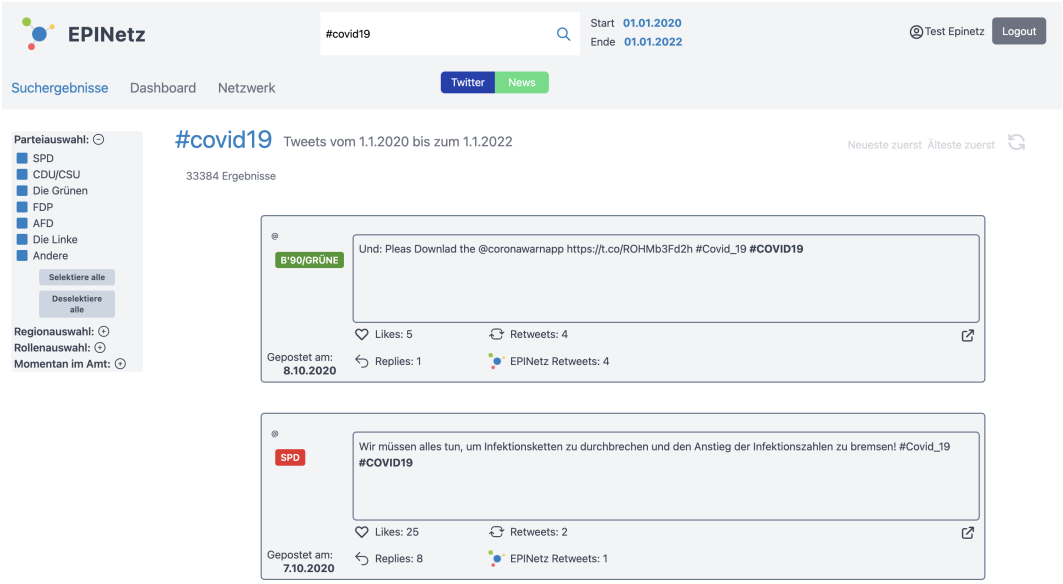


Figure 6.8: By conducting a search on the EPINetz platform, relevant documents can be retrieved in the form of a result list; Screenshot taken from the EPINetz web application on 19 July 2023

the root node. Starting from this root, the top 10 neighboring nodes as occurring in the EPINetz network are retrieved. Different relationship types might be used for that, depending on the root entity type. For example, the neighborhood of the *covid19* hashtag shown in Figure 6.9 might be derived based on *hashtag co-occurrence* or *Twitter user and hashtag co-occurrence* relationships. Depending on the selected link type, of course, different node types will be part of the neighborhood. The users can select the respective relationship type they are interested in. Regardless of the link type, the neighboring nodes are selected based on the number of existing edges to the root node that occur within the selected time frame. This number of edges is then also used to determine the edge weights in the resulting network. In contrast, the node weights are derived via the number of node occurrences in the filtered time window.

Given an initial network query, the users are able to click on additional nodes contained in the network. This way, the already explained neighborhood query step is repeated, and the additional adjacent nodes are added to the network. Following this mechanism, the users are able to traverse the EPINetz network and intuitively explore the context of the entities they are interested in. Usage scenarios could then be as simple as exploring topics and arguments over time or more complex, e.g., contrasting actors and arguments on different media outlets.

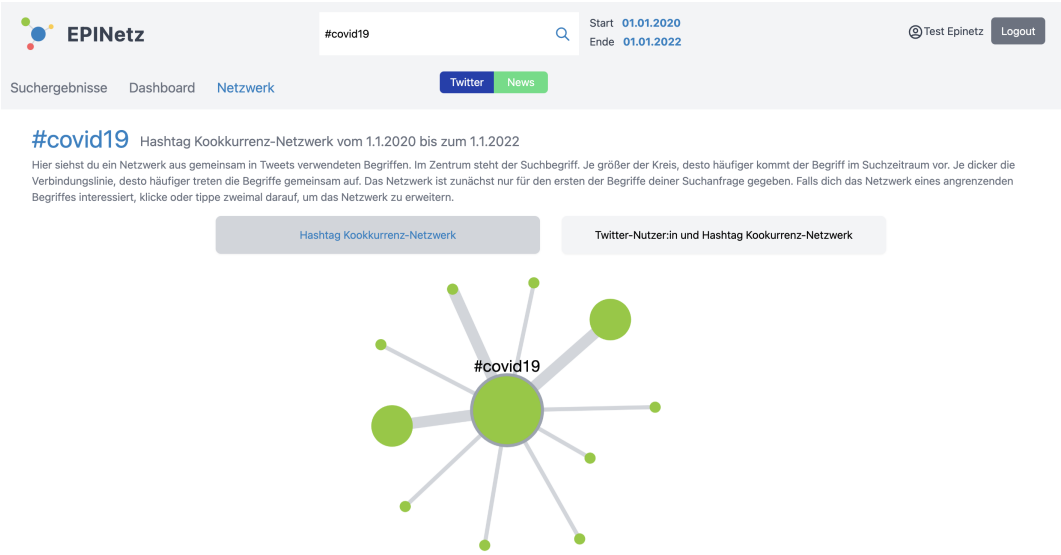


Figure 6.9: Entities of interest can also be examined using network-based exploration capabilities; Screenshot taken from the EPINetz web application on 19 July 2023

6.2.4 Conclusion and Future Work

With the steady increase of news outlets and channels in the media landscape, it will become more and more difficult for the public and citizens to deal with the amount and complexity of information, especially aspects related to political topics and debates. In the EPINetz project, we aim to address this problem by providing users with a web-based platform allowing them to search, explore, and contrast information surrounding political actors, topics, and debates at different scales, with data uniformly integrated from diverse sources, including social media. In this section, we outline how network-like structures provide a suitable means to explore actors and topics in context and over time while informing users from where and how the information is obtained. Underlying EPINetz is an extensive collection of German news outlets and a comprehensive Twitter dataset related to German politicians and organisations.

In the future, it is planned to offer the user a way to save and share exploration results and foster collaborative work. Thereby transparency is not only guaranteed if results are potentially referenced in the future but different approaches of how the data is filtered and analyzed can be compared.

6.3 Temporal Network Data Management

To provide the data analysis capabilities of the EPINetz platform presented in Section 6.2 or in general of a real-world system that allows to study the dynamics of heterogeneous networks according to the model described in Chapter 3, one needs to focus, during realization, on the management of the data and performant analysis interfaces. By comparing various systems and setup configurations, this work aims to find an implementation that best fits the described requirements. Finding such an implementation is not straightforward, as analyzing temporal (social) networks comes with its own set of challenges:

> "Consequently, queries on time series social networks are more complicated, and due to the concept of the time dimension, there are many query problems that are unique to time-dependent social networks." (Wang et al., 2019)

The system's requirements are detailed in Section 6.3.1. Primarily, they are derived from the volume of data, how it is structured and the research questions that are asked about it. After that, in Section 6.3.2, related work is discussed, followed by an outline of the benchmark's technical setup in Section 6.3.3 before its results are analyzed in Section 6.3.4. Finally, the derived insights and alternative approaches are discussed in Section 6.3.5.

6.3.1 Requirements

Multiple real-world system requirements can be formulated. These requirements cover a large number of possible application scenarios. Depending on the exact use case, they can be prioritized differently. By now, the model outlined in Chapter 3 serves as the methodological basis in terms of the analysis use cases and potential research questions. Also, the real-world experience gained from realizing the EPINetz platform inspired the proposed requirements.

Large data volume: Semantic networks extracted from text and actor-networks based on social media data are often quite large. For example, take one relatively short tweet of 25 tokens and a sliding window of 10. Per sliding window $\sum_{i=1}^{9} i = 45$ term co-occurrence relationships can be extracted. Given that the sliding window can be moved 15 times, in total $16 \cdot 45 = 720$ edges have to be taken into account. Now, if one wants to analyze 10 million tweets, the extracted term co-occurrence network already consists of 7.2 billion edges. Even though more restrictive assumptions can be made to limit the number of allowed edges, the example gives an idea of how quickly extracted semantic networks can become quite large. Of course, in addition to simple terms, one might also keep track of @-mentions, hashtags, or other use case-specific entities, which

further increases the size of the network. Therefore, a real-world system to analyze (socio-semantic) networks should be capable of storing and analyzing millions of edges.

Memory efficiency: To minimize memory consumption, the network's data should be stored in an efficient way. Considering the large volumes of data, as outlined above, this requirement becomes even more pressing.

Multiple node and edge types: Required capabilities of the system also include the handling of multiple node and edge types. As analyzed networks are not homogeneous, but instead cover different node types, such as terms, hashtags, and user accounts, as well as multiple edge types (e.g., co-occurrence, mentioning), an appropriate system should be able to handle described heterogeneity. This capability is also necessary for the system to be applicable to the EPINetz platform that also relies on analyzing heterogeneous networks.

Adjustable time resolution: Different temporal or topological patterns might only become visible at different time resolutions. Therefore, the specified system should allow the data to be queried on a single occurrence basis, up to the complete temporally aggregated network. As an example, different phenomena occurring in social media data have to be analyzed on different time scales, e.g., on a daily, weekly, or monthly basis.

Reproducibility: So that research results can be verified, corresponding analyses need to be reproducible. Therefore, in the described system data should be safely stored and repeated deterministic requests should result in the same retrieved data.

Transparency: Related to the requirement of reproducibility, transparency also calls for capabilities to retrace analysis results. Further, as the network's data is often extracted from less structured data, it is required that after the processing step the link to the original data is still accessible and stored in the system. As an example, a hashtag co-occurrence link extracted from Twitter data should in the system be accompanied by a reference to the original tweet. Thereby, it is transparent to the user of the system where the extracted data came from.

6.3.2 Related work

The general fields of graph data analytics systems and database benchmarking cover an enormous amount of academic work. Therefore, we restrict the following discussion of related work to the most similar research topics. As such, we cover work that studies the performance benchmarking of GDBMSs and Time Series Databases (TSDBs), especially in comparison to traditional RDBMSs, as well as various technical approaches regarding graph processing systems, followed by more theoretical work on graph modeling.

GDBMS benchmarking

The work of Macak et al. (2020) investigates the applicability of graph databases for big data analysis. Technically, they compare the performance of a three-node PostgreSQL cluster and a Neo4j cluster in case of queries that specifically target large amounts of highly-connected data. They use three types of queries: the first uses simple joins across multiple tables, the second also includes filtering conditions and the third leverages string checks instead of equality checks as done by the second type. For most queries, PostgreSQL outperforms Neo4j with some exceptions. The graph database's performance varies greatly depending on the directionality of stored edges. In line with that work, Ding et al. (2019) benchmark multiple RDBMSs, including PostgreSQL and GDBMSs, like Neo4j. For that, they propose a unified benchmark consisting of relational queries as well as graph algorithms. Their results prove superior performance to RDBMSs in the case of the "group by", "sort" and "aggregation" operations. In contrast, GDBMSs perform better under workloads including multi-table joins, pattern matching, path identification, and their combinations. Further, the analysis reveals that the performance of the graph databases varies significantly depending on the used storage engine. Underlining the good performance of Neo4j in the case of graph-based queries, Miler et al. (2014) compare the performance of a shortest path algorithm for PostgreSQL and Neo4j, which reveals superiority to the graph database. Nevertheless, the better performance of Neo4j also comes with a higher memory footprint.

Not taking PostgreSQL into account, but still comparing a relational database, namely MySQL, with Neo4j, the work by Vicknair et al. (2010) proves superior performance to the graph database for graph traversal queries. Still, for integer-based equality operations, the relational database performs significantly better. Sun et al. (2015) take a slightly different approach even though they also use a relational database as a graph data store. To support the Gremlin graph query language, they translate Gremlin requests to SQL queries. By evaluating different database storage schemas and optimizing the translated Gremlin queries, they can show the relational database's superior performance compared to Titan and Neo4j. Nevertheless, at least regarding the LinkBench evaluation, they do not evaluate *temporal* network queries. Unfortunately, when reading the publication (21 July 2023), the link to access the Gremlin queries of the DBPedia benchmark is not working anymore. Another benchmark comparing SQL and NoSQL data stores based on Gremlin LPG queries is conducted by Anikin et al. (2019). Their evaluation considers not only a single node as well as a cluster-based JanusGraph setup but also a single machine PostgreSQL and H2 setting. The benchmarking is conducted using various graph datasets. Overall, the PostgreSQL-based setup shows the best performance. In contrast to an actual performance comparison of different databases, in their work Ligtenberg and Pei (2017) present a temporal interval graph benchmark

dataset. For that, they compare multiple properties, like clustering coefficient, during static and temporal graph analysis.

TSDB benchmarking

In their work, Hao et al. (2021) present a time series benchmark and conduct a performance analysis of different TSDBs, including TimescaleDB. The proposed benchmark covers multiple workloads: data loading, injection, and fetching. Even though their work is not tailored to the use case of temporal network analysis and instead focuses on IoT analytics, it gives exciting insights into the performance of various time series databases. Regarding the TimescaleDB system, their results reveal poor query performances compared to the other systems, mainly because it lacks automatic index creation. In contrast, TimescaleDB reaches high write performance for data injection workloads and demonstrates the highest import efficiency. Similarly, Rinaldi et al. (2019) also conduct a performance analysis of a TSDB, i.e., InfluxDB. For that, they check different metrics, like ingestion rate, storage usage, and multiple queries based on time series data. They focus their evaluation on comparing different data models, specifically how the metadata, which is extensively used for analysis queries, is stored in the database. The results show that query performance varies significantly depending on how the data is stored internally. Queries that rely on metadata, which is stored as indexed "tags", are executed much faster.

Graph processing systems

Apart from benchmark-related studies, technical work about systems to serve graph-based queries exist. As such, Bronson et al. (2013) present "TAO", a data store specifically designed to support requests to the social graph underlying the Facebook platform. Two of its core features are geographic distribution and efficient data access. According to their requirements, they defined a data access API. Queries included in this API are the retrieval, update, and deletion of relationships as well as the request of adjacent nodes given a root node. They model the data as a temporal heterogeneous information network by recording edge and node types as well as the timestamp of an occurring edge. In their production setting, they observe that reads to the data are much more common compared to writes and that the query frequency of individual nodes, along with their connectivity, follow long-tail distributions. Further, Steer et al. (2020) propose the "Raphtory" system. It is a distributed system to manage and analyze temporal graphs. By leveraging network event streams, they are able to keep the complete history of each node and edge. Through their API, this full archival data is made accessible to the end user. Similarly, Iyer et al. (2021) propose "TEGRA" as a system for efficient ad-hoc queries on time-evolving graphs. The user can query and analyze the graph's state based on arbitrarily selected time windows. Thereby, the system's

outstanding performance is achieved by efficiently leveraging different temporal states of the graph as well as intermediate computational results, which are kept in memory. Another temporal graph analysis system is developed by Rost et al. (2022). Their system, termed "GRADOOP", comes with a rich set of (temporal) graph operators and algorithms. The evaluations show that the framework is able to handle graphs with billions of edges. In line with the discussed graph processing systems, various other related works can be mentioned, such as the "GraphBolt" system by Mariappan and Vora (2019), "Chronograph" (Erb et al., 2017) or "Naiad" (Murray et al., 2013) even though the last one is not limited to the processing of graphs.

Graph models

In contrast to the mentioned technical approaches, there are also more theoretical works in the temporal graph analysis field. As such, Moffitt and Stoyanovich (2017) propose an evolving graph model along with an algebraic query language to facilitate temporal network analysis. Next to their theoretical contributions, they outline various use case scenarios of their model and algebra, such as examining a node's influence over time or analyzing the spread of information across the analyzed network. Also, Angles (2018) formally describes the property graph database model, which is leveraged by different GDBMSs, such as Neo4j. He describes the graphs that follow the property graph model as "labeled multigraphs where both nodes and edges can contain pairs of the form property-value".

6.3.3 Benchmark setup

The following section details the setup of the conducted graph data analysis benchmark. This specification includes the description of the leveraged dataset in Section "Dataset", the coverage of the benchmark graph queries in Section "Queries" and various details about the used configurations in Section "Technical Implementation".

Remark. Notably, the conducted benchmark is tailored to the specific use case of finding a suitable technical implementation that fits the requirements outlined in Section 6.3.1. These requirements also include the need for the system to work with the EPINetz platform examined in Section 6.2. Therefore, one might argue that the proposed benchmark is not a graph benchmark in the common sense (see Dominguez-Sal et al. (2010)) but aims at the fulfilment of these use case requirements and reveals various characteristics that go into the direction of time series analysis, e.g., the benchmarked queries are all tailored to *time-dependent* graphs.

Dataset

For the benchmark, a subset of the temporal heterogeneous information network dataset presented in Section 6.1 is used. With regards to the actual performance in the EPINetz platform production environment, fast access to the projected networks is most important. Therefore, this kind of network data is also used for the benchmark to compare the performance of various graph data management and analytics systems. In line with the model outlined in Chapter 3, the analyzed network projections can be described as entity network projections with edge-attributed timestamps of the source documents. Exemplary network projections are timestamped hashtag co-occurrence networks or Twitter mention graphs. The following network schema describes the dataset used for the benchmarking:

- **Node types:** Twitter user, hashtag
- **Edge types:** Twitter user and hashtag co-occurrence, Twitter user mentions Twitter user, Twitter user uses hashtag, Twitter user co-occurrence, Twitter user mentioned by Twitter user, hashtag and Twitter user co-occurrence, hashtag co-occurrence, hashtag used by Twitter user

One might notice from comparing the edge types listed here with the types listed in Section 6.1 that for the benchmark dataset, the inverted edge types (e.g., hashtag used by Twitter user) are also used. This bi-directionality is necessary to support neighborhood queries following both edge directions. In line with that, each co-occurrence edge is represented in the form of two directed edges between the adjacent nodes. Also, the nodes are identifiable based on IDs which are unique across all node types.

To meet the requirements specified in Section 6.3.1, various aspects are considered for the dataset curation. First of all, to satisfy the "large data volume" requirement, the dataset is available in multiple sizes. These size variants allow for investigating the systems' performance based on different dataset sizes and to check whether the system is able to adjust to a growing network dataset. Table 6.9 shows the different scales of the benchmark dataset along with their respective network size in terms of temporal edges contained in the network.

Another stated requirement is referred to as "multiple node and edge types" and describes the need for the system's capability to handle heterogeneous information networks. Such networks lay the foundation of the EPINetz data analysis platform described in Section 6.2. Therefore, the network processing system should be able to handle such types of networks as well. As one can see from the above list of node and edge types contained in the used network dataset, this requirement is fulfilled, too. Figure 6.10 visualizes the heterogeneity among the node types, which

Table 6.9: Different scales of the temporal network dataset used for the benchmarking. With increasing scale, more edges are part of the network.

Scale	Number of edges
0.1	100,000
1	1,000,000
10	10,000,000

shows the volume of nodes grouped by their type as appearing in the scale 1 network used for the benchmarking. As one can see, about 15 k nodes are hashtags (≈ 44 %), and about 19 k nodes are Twitter users (≈ 56 %). In total, the scale 1 benchmark network contains 34, 408 nodes.

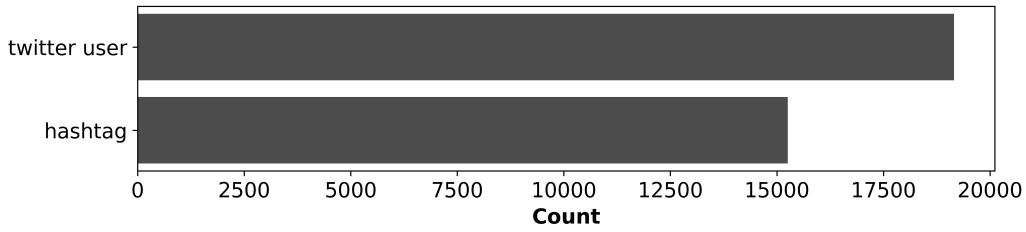


Figure 6.10: Number of nodes included in the scale 1 network used for the benchmark. Nodes are grouped by their type.

The heterogeneity among edge types appearing in the benchmark network is even more remarkable. Figure 6.11 shows the respective distribution of the edge types. Overall, each edge type makes up around 9 to 21 %, showing that the 1 M edges are distributed relatively equally. The top edge types, each making up about one-fifth of the overall edges, are "twitter user co-occurrence edges" and "hashtag co-occurrence edges". As already discussed in Section 6.3.1, the number of term co-occurrences extracted from textual documents increases very rapidly with an increasing sliding window. Similarly, multiple hashtags used or users mentioned in a Tweet result in numerous co-occurrence edges. Therefore, it is reasonable that two types of co-occurrence edges dominate the edge types appearing in the network.

Table 6.10 shows five exemplary edges occurring in the raw benchmark data used for the scale 1 network. As one can see, each edge comes with IDs of the start and end nodes, a timestamp, as well as the respective edge label. For example, the first edge in the table indicates a co-occurrence relationship between a Twitter user (node ID 10798904) and a hashtag (node ID 39178) that occurred on 14 December 2022 at 22:59:46.

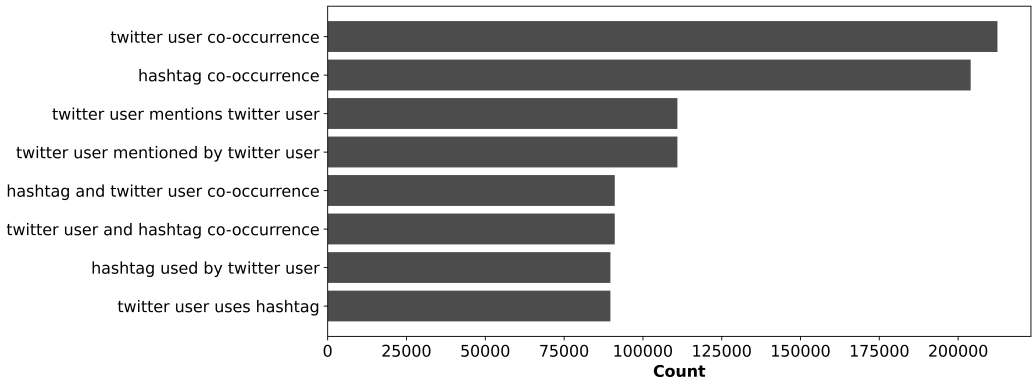


Figure 6.11: Number of edges included in the scale 1 network used for the benchmark. Edges are grouped by their type.

In sum, the benchmark set of scale 1 covers approximately two weeks of data, as shown by Figure 6.12. Of course, for the other scales, the covered time window is smaller (scale 0.1) or larger, respectively. For scale 1, the dataset starts from the beginning of December 2022 and lasts until the mid of the same month. Clearly, large fluctuations in the volume of data can be observed. These variations are probably caused by the day-night rhythm with the users posting less frequently at night. This daily rhythm is particularly dominant because the dataset mainly covers posts of German users that live in the same timezone.

Shifting the focus of the dataset description towards the network's topology, Figure 6.13 shows its degree distribution concerning outgoing edges. Given that for an ingoing edge, its respective outgoing counterpart is also included in the network, the degree distribution regarding ingoing links would look similar. As the added power law fit indicates, the degree distribution approximately follows a long tail distribution. Compared to the power law fit, fewer high-degree nodes and more low-degree nodes, as expected, exist. Also, the estimated power law exponent of 1.35 is outside of

Table 6.10: Sample of temporal edges contained in the benchmark dataset for scale 1

dt	start_id	end_id	label
2022-12-14 22:59:46	10798904	39178	twitter_user_hashtag_co_occurrence
2022-12-14 22:59:46	11654370	39178	hashtag_co_occurrence
2022-12-14 22:59:46	10794851	39178	twitter_user_hashtag_co_occurrence
2022-12-14 22:59:46	11060784	39178	twitter_user_uses_hashtag
2022-12-14 22:59:46	11060784	10794851	twitter_user_mentions_twitter_user

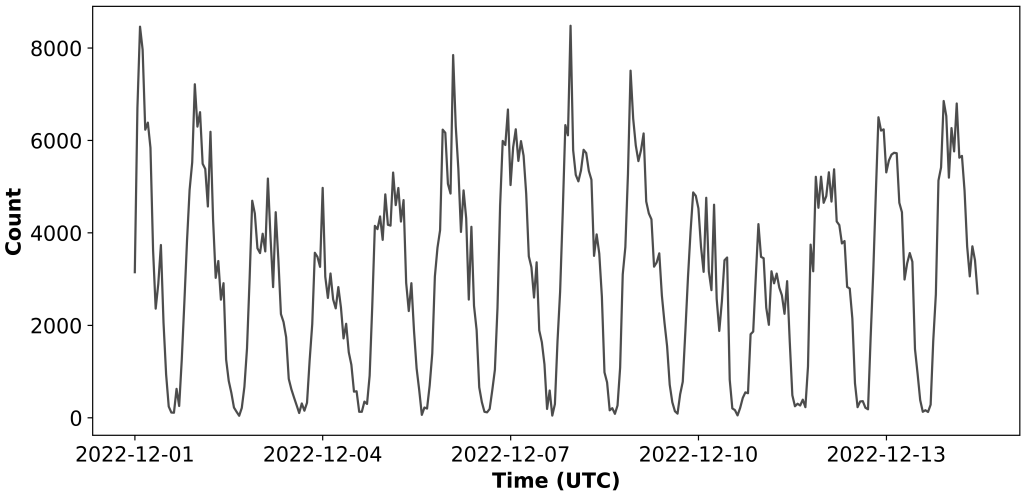


Figure 6.12: Temporal distribution of the edges included in the scale 1 network used for the benchmark

the (2, 3) range typically observed in real-world scale-free networks (Newman, 2010, Section 8.4). However, the distribution is similar to the query frequency and node connectivity distribution observed by Bronson et al. (2013) in the sense that it also represents a long tail characteristic. Typical social network distributions seem to be characterized this way. Further, an average degree across all node and edge types of 0.03 reveals that most nodes are not highly connected.

Queries

Optimizing a graph data management and analytics system with regard to all possible requirements is infeasible. Therefore, specifically selected data queries are used for the benchmarking. In the case of complex queries, the system's performance is highly affected by the leveraged algorithms (see Hao et al. (2021)). As a result, we focus on reasonably simple queries to ensure the actual system performance is measured. In line with the "adjustable time resolution" requirement outlined in Section 6.3.1, these queries are not tight to fixed time windows, but data of arbitrary time windows can be retrieved. The same is true for the queried edge types. These can also be set flexibly, which is essential to comply with the "multiple node and edge types" requirement. Further, none of the queries aims at measuring the data ingestion performance as this is not a critical factor for our use case. In contrast to data loading times, it is more important for the end user to retrieve the data in a fast way for our use case. In the following, a list of all the benchmarked queries is provided. The pseudocode of these queries is provided in Appendix A. Given that not all tested systems are SQL-compliant, the queries had to be translated into a hybrid SQL and openCypher version used

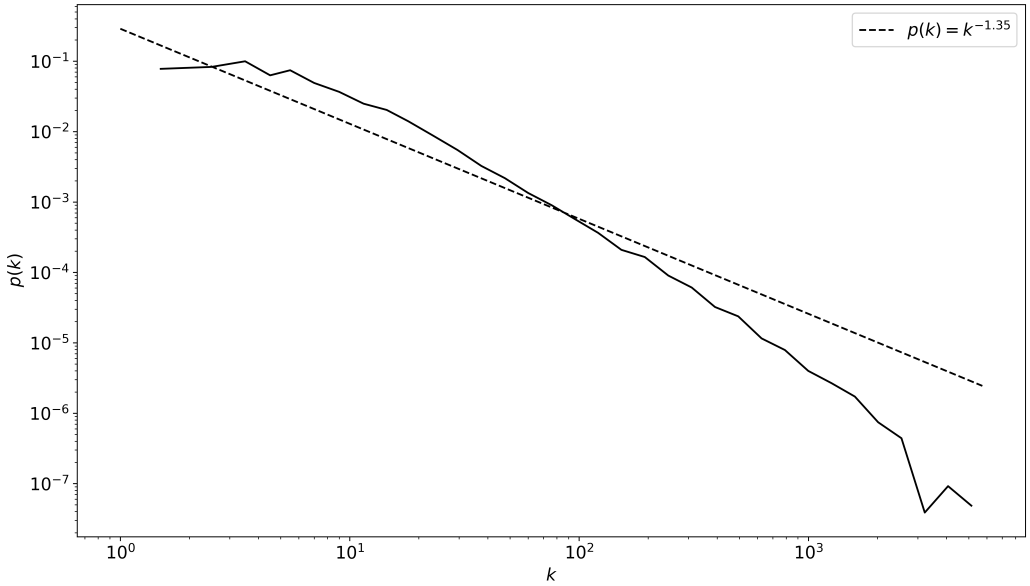


Figure 6.13: Outdegree distribution of the scale 1 network used for the benchmark

by Apache AGE and a pure Cypher version used by Neo4j. The respective SQL queries leveraged for the native PostgreSQL setup and the TimescaleDB setup are given in Section A.1.

(Q1) **Temporal degree:** Given a time window, an edge type, and a node ID, the query allows to compute the node's degree over time. The respective time granularity is set to one day.

(Q2) **Top-10 neighbors:** Given a time window, an edge type, and the ID of the queried seed node, its top ten neighbors (with respect to the set edge type and time window) are retrieved. The ranking is determined based on the number of relationships between the seed node and the adjacent nodes.

(Q3) **3-hop neighborhood:** With this query, the nodes that are three hops away from a given seed node are retrieved. Only the edges of the given types are considered for the individual hops. Also, only edges falling into the given time window are considered.

From a time series analysis perspective, the first query is an *evolution* metric as it measures the temporal development of the node degree metric (see Sections 2.1.5 and 3.3.2). In contrast, the other two queries represent snapshot-based metrics as they only take the data of the defined time window into account. Both types of queries are used on the EPINetz platform and, therefore, need to be part of the benchmark.

Table 6.11: Sample of benchmark query parameters used for scale 1

start_id	start_dt	end_dt	edge_type_1	edge_type_2	edge_type_3
11187212	2022-12-12 19:55:19.24 1237688	2022-12-13 15:50:17.97 8386113	twitter_user_ mentioned_by _twitter_user	twitter_user_ uses_hashtag	hashtag_co_ occurrence
750205	2022-12-11 20:57:11.55 5022970	2022-12-14 08:11:00.56 1121195	hashtag_co_ occurrence	hashtag_ twitter_user_ co_occurrence	twitter_user_ hashtag_co_ occurrence
1539	2022-12-09 02:46:03.30 4083018	2022-12-14 03:08:57.31 6909700	hashtag_co_ occurrence	hashtag_ used_by_ twitter_user	twitter_user_ mentioned_by _twitter_user

Next to the actual queries, parameters for these queries must also be available for the benchmark. In our case, we generate synthetic data that can be used to parametrize the queries described above. This generation is based on the temporal network data used for the various benchmark scales by randomly selecting and combining node and edge information. Thereby, specific restrictions are fulfilled by the generated data, such as that the target node type of an edge type matches the source node type of its adjacent edge type. In total, 1000 query parameter sets are generated per dataset scale. Table 6.11 gives three example query parameter sets used for the scale 1 benchmark. Its details are discussed in the following:

- **start_id:** All three benchmarked queries are centered around a seed node. This node is clearly identified by the given *start_id*. Also, the type of this node matches the source node type required by the *edge_type_1*. This matching is needed as graph traversal-based queries would not return any adjacent nodes otherwise. Of course, all used node IDs occur in the benchmark dataset used for the specific scale.

- **start_dt** and **end_dt:** Apart from the "Temporal degree" query, the two provided timestamps define the time window used for the temporal snapshot the queries are applied to. In contrast, for the derivation of the temporal degree, the time windows used for the network snapshots are set to one day, and these timestamps determine the starting and end date for which the metric is derived. In all cases, $start\_dt \leq end\_dt$ holds.

- **edge_type_1**, **edge_type_2** and **edge_type_3:** The three edge types are used to type-set the network queries. Only for the "3-hop neighborhood" query all edge types are leveraged. Thereby, the target node type of *edge_type_(n)* matches the source node type of *edge_type_(n+1)*. For example, a "twitter_user_ mentioned_by_twitter_user" edge type is

> followed by a "twitter_user_uses_hashtag" link type. Of course, all of the used edge types appear in the leveraged benchmark dataset.

In line with the exploration capabilities on the EPINetz platform, all of the benchmarked queries are centered around a single source node. Users on the platform also start their data exploration from a specific entity of interest. In a second step, they might then derive related statistics or traverse the graph for additional network investigations.

Technical implementation

In the presented benchmark, not only traditional GDBMSs are tested, but also relational DBMSs like PostgreSQL. This approach is based on past findings such as the one by Sun et al. (2015):

> "[...] existing mature, relational optimizers can be exploited with a novel schema to give better performance for property graph storage and retrieval than popular noSQL graph stores."

Seemingly, relational databases can also be used for performant graph analysis. At the same time, it is not sufficient to pick a good-performing data management system, but using correct indices and partitioning aligned with queries is equally essential (see Rinaldi et al. (2019)). For that, it is crucial to find typical patterns of how the data is accessed (Ceri et al., 1982). Therefore, different data management systems, as well as various configurations, are tested. For the benchmark, only open source systems, respectively, such that offer a self-hostable community edition, are selected. As such, the set of selected systems is not exhaustive, and the specific selection is based on the authors' technical know-how and past working experience.

The following section describes the technical implementation of the benchmarked systems in detail. First, the technical details applying to all PostgreSQL-based systems (native PostgreSQL, TimescaleDB, and ApacheAGE) are discussed, followed by investigations of the system-specific implementations, including the previously mentioned systems plus Neo4j. Finally, the leveraged virtual machine setup is described. In general, all system setups are based on Docker[10] to run the respective application containers. Therefore, a unified execution environment is guaranteed. Also, the Docker runtime is configured with a shared memory size (`shm-size`) of 20 GB (`20g`).

Setup for PostgreSQL-based systems: The PostgreSQL-based systems (i.e., TimescaleDB, Apache AGE, and the native PostgreSQL setup) have some implementation details in common. Concerning the `shared_buffers` setting, which determines the cache size (Ahuja, 2022), their configured values are below the shared memory setting of the Docker runtime. This lower value

[10] Docker: Accelerated Container Application Development: `https://www.docker.com` (accessed 2023-08-14)

is necessary for the application process(es) to not run out of memory (Mackay, 2023). Further, the database is configured to allow a maximum of 100 concurrent connections. Optimization strategies are applied to the definition of table indices. As a rule of thumb for multi-column indices, the column which is used to apply the most restrictive condition during data filtering is defined first (see Winand (2023)). Also, to speed up specific queries, filtering columns are defined as the leftmost columns during index creation. Another critical factor to consider if one wants consistent database performance is when database statistics gathering and vacuuming are applied. To not arbitrarily interrupt the benchmark, auto-vacuum, which removes no longer needed tuples (Ahmed, 2023), is switched off for the PostgreSQL-based systems. Instead, "vacuum analyze" is executed before every benchmark run. Finally, to optimize the physical layout of the data on disc, the tables are clustered according to the used indices (see Schönig (2020)). Of course, this can only be done if no conflicting indices are present. Unfortunately, GIN indices (see Section 2.4.1) cannot be clustered given that they do not impose a natural order on the data (see Travers (2018)). Given this clustering and the respective physical alignment of the data, randomized ingestion of the network data is not applied. The data would be ordered according to the applied index in any case.

Furthermore, to prevent performance deviations due to "cold-starting", the tables used to store the temporal graph data, as well as the query parameters, are pre-warmed using the `pg_prewarm` module (PostgreSQL Global Development Group, 2023c). Fetched data is loaded into the cache this way already. Also, because arbitrary time windows can be used during data analysis, the needed time resolution cannot be known in advance and leveraging pre-aggregated snapshots becomes infeasible. In the same regard, partitioning the graph stream table with respect to a particular time granularity is not practical.

PostgreSQL-specific setup: A native PostgreSQL database setup without any additional extension is also part of the benchmarked systems. For this setup, the database configuration needs to be adjusted to fit the temporal graph processing use case. The PGTune[11] online tool is used for parameter optimization. Given that the EPINetz platform is supposed to be used in various ways, its application type is set as "Mixed Type of Application" for the PGTune parameter recommendation. This type of application works for data warehousing as well as online transaction processing requirements. The detailed parameter recommendations are shown in Listing 6.1.

Listing 6.1: PostgreSQL PGTune recommendations

```
# DB Version: 14
# OS Type: linux
```

[11] PGTune – calculate configuration for PostgreSQL based on the maximum performance for a given hardware configuration: https://pgtune.leopard.in.ua (accessed 2023-08-15)

```
# DB Type: mixed
# Total Memory (RAM): 60 GB
# CPUs num: 8
# Connections num: 100
# Data Storage: ssd

max_connections = 100
shared_buffers = 15GB
effective_cache_size = 45GB
maintenance_work_mem = 2GB
checkpoint_completion_target = 0.9
wal_buffers = 16MB
default_statistics_target = 100
random_page_cost = 1.1
effective_io_concurrency = 200
work_mem = 19660kB
min_wal_size = 1GB
max_wal_size = 4GB
max_worker_processes = 8
max_parallel_workers_per_gather = 4
max_parallel_workers = 8
max_parallel_maintenance_workers = 4
```

Apart from the set configuration parameters, it is important to mention that the `postgres:14.3-alpine` PostgreSQL Docker container[12] is used. Also, the temporal network data is stored in the form of a temporal edge list. The following columns are used for the respective edge table: 1. *id:* unique identifier, 2. *dt:* timestamp, 3. *source:* source node ID, 4. *target:* target node ID, and 5. *label:* edge type.

Compared to an adjacency matrix, this storage format is more efficient as only the actual edges are stored (see Bianconi (2018, p. 109)). This advantage is significant in the case of sparse networks, which are used as benchmark data in our case. Commonly, sparse network data is stored in the form of an edge list (see Bianconi (2018, p. 106)).

When designing a database setup for a specific use case, such as temporal graph processing, one must consider various data indexing techniques for faster data querying. Therefore, multiple indexing strategies are benchmarked to test the effect on the edge data access performance. Table 6.12

[12] Image Layer Details – postgres:14.3-alpine | Docker Hub: https://hub.docker.com/layers/library/postgres/14.3-alpine/images/sha256-84c6ea4333ae18f25ea0fb18bb142156f2a2e545e0a779d93bbf08079e56bdaf?context=explore (accessed 2023-10-10)

shows the various indices and index combinations tested during the benchmark. Thereby, the column/table names in brackets reference the column names used in the pseudocoded queries (see Appendix A.1). Regarding the conducted experiments, the "baseline" setup is only tested for scale 0.1. For larger data volumes, data retrieval without indices would have taken extensive time.

Table 6.12: List of benchmarked indices applied to the edge stream (temporal edge) table for the native PostgreSQL setup

	Label	**Description**
1	baseline	no index
2	index (source)	B-tree index on *source* (source node) column
3	index (source) & index (target)	same as 2 + B-tree index on *target* (target node) column
4	index (dt)	B-tree index on *dt* (timestamp) column
5	index (dt) & index (target)	same as 4 + B-tree index on *target* (target node) column
6	index (label)	B-tree index on *label* (edge type) column
7	index (label) & index (target)	same as 6 + B-tree index on *target* (target node) column
8	index (source,label,dt)	multi-column B-tree index on *source* (source node), *label* (edge type) and *dt* (timestamp) columns
9	index (source,label,dt) & index (target)	same as 8 + B-tree index on *target* (target node) column
10	partial index (source,dt)	partial, multi-column B-tree index on *source* (source node) and *dt* (timestamp) columns with equality condition on *label* (edge type) column
11	partial index (source,dt) & index (target)	same as 10 + B-tree index on *target* (target node) column

TimescaleDB-specific setup: As already mentioned, the benchmarked systems are set up based on Docker. In the case of the TimescaleDB system setup, the container, `timescale/timescaledb:2.9.1-pg14`[13], is used. Further, the database's parameters are opti-

[13] Image Layer Details – timescale/timescaledb:2.9.1-pg14 | Docker Hub: https://hub.docker.com/layers/timescale/timescaledb/2.9.1-pg14/images/sha256-5e37b20e64a8e33443b7358159a4e022feffed1067fb320c7d820d4f8fe3149b?context=explore (accessed 2023-10-10)

mized using the TimescaleDB tuning tool[14]. Its recommendations are shown in Listing 6.2. The TimescaleDB database is configured accordingly.

Listing 6.2: TimescaleDB tuning tool recommendations

```
# Recommendations based on 60.00 GB of available memory and 8 CPUs for
    PostgreSQL 14

shared_buffers = 15GB
effective_cache_size = 45GB
work_mem = 19660kB
max_worker_processes = 27
max_parallel_workers_per_gather = 4
max_parallel_workers = 8
```

In line with the storage of the temporal network data, as implemented for the native PostgreSQL setup, for TimescaleDB, the data is also stored as a temporal edge list. This edge list is configured as a hypertable. For the hypertable configuration, it is crucial to set an appropriate `chunk_time_interval` which defines the time windows used for the time partitioning. For the scale 1 benchmark, the edge stream hypertable is about 88 MB large. According to the Timescale documentation, it is recommended that all recent hypertable chunks fit into 25 % of main memory (Timescale Inc., 2024). In our case, this fraction of memory equals about 15 GB. Therefore, up to scale 170 ($\frac{15\,\text{GB}}{88\,\text{MB}} \approx 170$) all the temporal network data fits into memory. The network data could even be completely stored in a single chunk. Still, it is split into multiple chunks to benefit from partitioning, which is especially crucial for larger data volumes. Therefore, the `chunk_time_interval` parameter is set to 1 month. Given that the scale 1 data covers about 15 days, the hypertable consists of multiple chunks for larger scales. For scale 10, five chunks are used.

Furthermore, similar to the setup used for the native PostgreSQL implementation, various indexing techniques are tested. Table 6.13 gives an overview of applied indices and index combinations. Apart from the "baseline" setup, which is only tested for scale 0.1, the others are benchmarked for all scales.

[14] GitHub – timescale/timescaledb-tune: A tool for tuning TimescaleDB for better performance by adjusting settings to match your system's CPU and memory resources.: https://github.com/timescale/timescaledb-tune (accessed 2023-08-16)

Table 6.13: List of benchmarked indices applied to the edge stream (temporal edge) table for the TimescaleDB setup

	Label	Description
1	baseline	no index
2	index (source,dt)	multi-column B-tree index on *source* (source node) and *dt* (timestamp) columns
3	index (source,dt) & index (target)	same as 2 + B-tree index on *target* (target node) column
4	index (label,dt)	multi-column B-tree index on *label* (edge type) and *dt* (timestamp) columns
5	index (label,dt) & index (target)	same as 4 + B-tree index on *target* (target node) column
6	index (source,label,dt)	multi-column B-tree index on *source* (source node), *label* (edge type) and *dt* (timestamp) columns
7	index (source,label,dt) & index (target)	same as 6 + B-tree index on *target* (target node) column

Apache AGE-specific setup: For the Apache AGE benchmark setup the `apache/age:v1.1.0` Docker container[15] is used. Also, the database is configured with the same parameters leveraged for the native PostgreSQL setup. Internally, Apache AGE uses a table layout optimized for graph data management. Table 6.14 gives an overview of the database design. It is important to note that a separate namespace is created for each newly added graph. Also, the `_ag_label_vertex` and `_ag_label_edge` tables work as parent tables of the respective node and edges of a graph (Farias, 2023). Compared to the way the data is stored for the TimescaleDB and the native PostgreSQL setup, the Apache AGE table layout is quite different. Instead of simply storing the data as a temporal edge list, it is more complex and focused on separating the graph data based on the various node and edge types.

Based on the described database design used internally by Apache AGE, various indexing techniques are tested (see Table 6.15). Since the edges are stored in separate tables according to their type, indices are created for each of these individually. Also, the edge and node properties are stored as a JSON-compliant data type. Therefore, B-tree indices on the edge timestamps must be created on the respective JSON object fields. Compared to the other system setups, GIN indices are also tested. They are created on the "properties" columns of the edge tables. Still, in line with

[15] Image Layer Details – apache/age:v1.1.0 | Docker Hub: https://hub.docker.com/layers/apache/age/v1.1.0/images/sha256-4cbc8ad05877772d8fbcfb05e5701eb9e414001e684f50f392d3e78d2cf9fe87?context=explore (accessed 2023-10-10)

Table 6.14: Internal table layout of Apache AGE

Table(s)	Description
ag_catalog.ag_graph	list of graphs
ag_catalog.ag_label	list of all node and edge types
⟨graph⟩._ag_label_vertex	all nodes of respective graph
⟨graph⟩._ag_label_edge	all edges of respective graph
⟨graph⟩.⟨node type⟩	all nodes of respective type in graph
⟨graph⟩.⟨edge type⟩	all edges of respective type in graph

the native PostgreSQL and the TimescaleDB setup, the "baseline" configuration is only tested for scale 0.1.

Neo4j-specific setup: With Neo4j being part of the benchmark, a system not based on PostgreSQL is also tested. For the benchmark, the `neo4j:5.8.0-community` Docker image[16] is used. Unfortunately, Neo4j does not come with any pre-warming capabilities. As a workaround, the database cache is warmed by querying all nodes and edges of a graph previous to the actual benchmark (Gordon, 2023). Nevertheless, statistics about the stored data needed to optimize the query plans are automatically kept up to date, and no manual statistics collection needs to be triggered (Neo4j, Inc., 2023a). Further, the database memory settings are configured by following the memory recommendations provided by the Neo4j `memory-recommendation` tool[17]. Listing 6.3 shows the recommendations for the used virtual machine setup. Also, the database's configuration is validated via the `validate-config` tool (Neo4j, Inc., 2023b).

Listing 6.3: Neo4j memory recommendations

```
# Based on the above, the following memory settings are recommended:
NEO4J_server_memory_heap_initial__size='23000m'
NEO4J_server_memory_heap_max__size='23000m'
NEO4J_server_memory_pagecache_size='26g'
#
# It is also recommended turning out-of-memory errors into full crashes,
# instead of allowing a partially crashed database to continue running:
NEO4J_server_jvm_additional='-XX:+ExitOnOutOfMemoryError'
```

[16] Image Layer Details – neo4j:5.8.0-community | Docker Hub: https://hub.docker.com/layers/library/neo4j/5.8.0-community/images/sha256-dc272726680f27f10425e6697cc405103478e358968976515aadfa11b319cbd4?context=explore (accessed 2023-10-10)

[17] Memory recommendations – Operations Manual: https://neo4j.com/docs/operations-manual/current/tools/neo4j-admin/neo4j-admin-memrec (accessed 2023-07-16)

Table 6.15: List of benchmarked indices applied to the Apache AGE graph database setup

	Label	Description
1	baseline	no index
2	index (source)	B-tree index on *source* (source node) column
3	index (source) & index (target)	same as 2 + B-tree index on *target* (target node) column
4	index (dt)	B-tree index on *dt* (timestamp) JSON object field of properties column
5	index (dt) & index (target)	same as 4 + B-tree index on *target* (target node) column
6	index (source,dt)	multi-column B-tree index on *source* (source node) column and *dt* (timestamp) JSON object field of properties column
7	index (source,dt) & index (target)	same as 6 + B-tree index on *target* (target node) column
8	GIN index (properties)	GIN index on *properties* (edge properties) column
9	GIN index (properties) & index (target)	same as 8 + B-tree index on *target* (target node) column

Since Neo4j also has various data indexing capabilities, multiple indexing strategies are tested for the Neo4j setup. These indices are shown in Table 6.16. Unfortunately, a composite index on node *and* edge properties is not possible. Instead, individual indices for node (IDs) and relationship (dates) information are used. Similar to the Apache AGE setup, indices must be applied to each node and edge type individually. Next to the indices applied to the actual graph data, a range index on the benchmark query data is also created. Therefore, the generated query parameter sets can be retrieved faster based on their IDs. In contrast to the setups discussed above, for the Neo4j setup, the "baseline" configuration is benchmarked for all scales.

Virtual machine setup: The complete benchmark is executed on a virtual machine that runs on Google Cloud[18] infrastructure. The Google service offers free credits of $300 to test their service. These are used for benchmarking. Still, for these test accounts, some usage restrictions exist. As such, only a maximum of 8 vCPUs for `n2` machines is allowed, and `c3`-type virtual machines are not allowed to be used. Manually increasing these usage quotas does not work. Given that more CPUs are better for parallelizing the database queries (e.g., for TimescaleDB), the maximum of

[18]Cloud Computing Services | Google Cloud: `https://cloud.google.com/?hl=en` (accessed 2023-08-16)

Table 6.16: List of benchmarked indices applied to the Neo4j graph database setup

	Label	Description
1	baseline	no index
2	index (rel:dt)	range index on *dt* (timestamp) relationship property
3	index (node:id),(rel:dt)	range index on *id* node and *dt* (timestamp) relationship property
4	index (node:id)	range index on *id* node property

8 vCPUs is used. Specifically, the `n2-highmem-8` general-purpose machine type[19] is used. The cores of this machine type are not shared, which is crucial for consistent resource availability. Also, these virtual machines come with 64 GB of RAM. Given that Docker and other processes also need a certain amount of memory, the benchmarked systems are set up with an expected available RAM of 60 GB. Regarding SSD memory, the usage quota is also completely taken advantage of, and 1.5 TB of local SSD is used. Additionally, the virtual machine is configured with a 125 GB persistent disk SSD for booting. Further configuration details of the virtual machine used for the benchmark are listed in Table 6.17.

Table 6.17: Provisioning of the virtual machine used for the benchmarking

CPU(s)	8
On-line CPU(s) list	0–7
Thread(s) per core	2
Socket(s)	1
NUMA node(s)	1
Model name	Intel(R) Xeon(R) CPU @ 2.80 GHz
NUMA node0 CPU(s)	0–7

6.3.4 Benchmark results

This section examines the results of the temporal graph data analysis benchmark. For the benchmark, the results are measured in "Transactions Per Second" (TPS). Therefore, the higher the values, the better the tested performance. Also, each setup, consisting of a database system, a dataset of a particular scale, a specific benchmark query and a data indexing strategy, is tested 100

[19] General-purpose machine family for Compute Engine | Compute Engine Documentation | Google Cloud: https://cloud.google.com/compute/docs/general-purpose-machines#n2_machine_types (accessed 2023-08-16)

times. For each run, the performance metrics are stored and given that most setups show highly skewed distributions of their TPS values due to variations in query performance, see Figure 6.14 for example, median values are used for comparison. In line with that, to quantitatively measure the variance in system performance, median absolute deviation (MAD) values are derived and used to discuss the benchmark results. Notably, concerning MAD values, one has to take into account the characteristic that these can be zero in cases of distributions where more than 50 % of the values are equal, even though some sort of "variance" is present in the data (Rosenmai, 2013). Apart from the benchmark measurement, the benchmark itself is conducted up to the dataset scale of 10 as described in Section 6.3.3. Testing the systems for an even larger dataset would have been desirable. Unfortunately, given the hardware constraints, benchmarking more significant amounts was unfeasible. Even the native PostgreSQL setup ran out of disk space for a potentially tested dataset of scale 100 with 100,000,000 temporal edges.

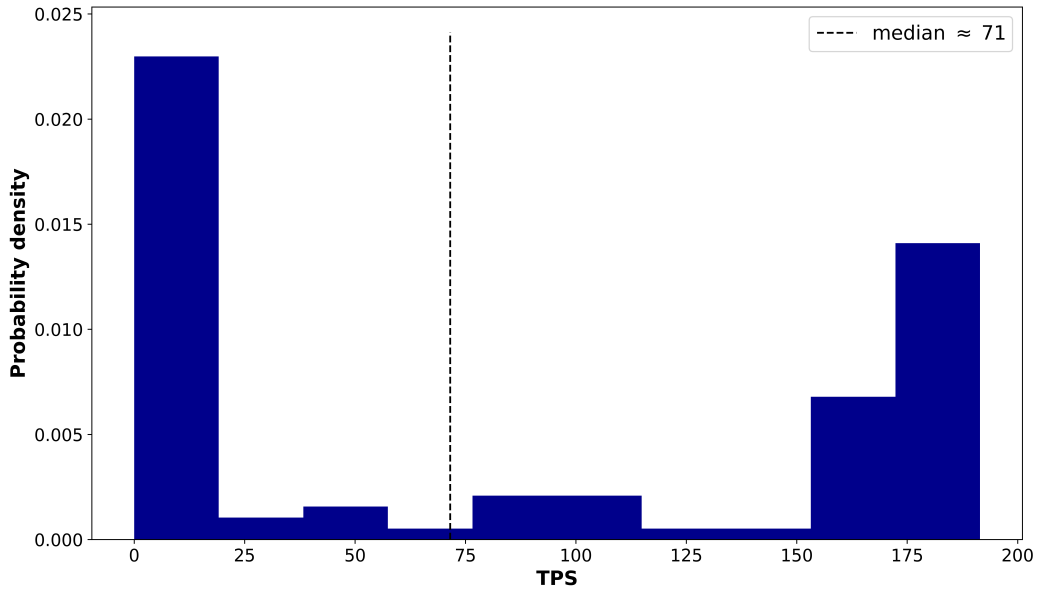


Figure 6.14: Probability density distribution of the performance results for the native PostgreSQL setup in case of the scale 10 benchmark dataset, the "3-hop neighborhood" query and the "index (source,label,dt) & index (target)" data indexing strategy. A non-normal distribution can be observed.

The benchmark results are presented in the following by first giving an overview in Section "Overview". Then, Section "Results by system" discusses the test results in detail on a per-system level. This discussion includes a comparison of the different dataset scales and queries, as well as an investigation of which data indexing strategies work best in which case.

Overview

To guarantee transparency, Appendix B shows the complete list of benchmarking results – see the respective section for more details. Still, gaining deeper insights from Table B.1 might not be easy. Therefore, more focused perspectives on the benchmark results are provided in the following. As such, Table 6.18 shows the best-performing system for each tested dataset size and benchmark query. For each database system, only the best-performing index setup is taken into account. Thereby, the setups are ranked by their median TPS values. As one can see, for the smallest dataset scale, the Neo4j system performs best for all queries, and for the other dataset scale and query combinations, the native PostgreSQL system performs best. In general, one can derive that the native PostgreSQL setup fulfills the outlined requirements best and should, therefore, also be used for the EPINetz platform – especially because an appropriate system needs to be capable of handling large volumes of data. Supposedly, PostgreSQL scales well for the graph analysis use case. In the following, the setup-specific results are discussed in detail.

Table 6.18: Best performing system by dataset scale and benchmark query

	Temporal degree	**3-hop neighborhood**	**Top-10 neighbors**
0.1	Neo4j	Neo4j	Neo4j
1	Native PostgreSQL	Native PostgreSQL	Native PostgreSQL
10	Native PostgreSQL	Native PostgreSQL	Native PostgreSQL

Figure 6.15 shows a performance comparison of the benchmarked systems for the scale 0.1 dataset. The best-performing index configuration, according to the maximum median TPS value, is considered for each system and benchmark query setup. As one can see, Neo4j clearly outperforms the other systems for all three benchmark queries. All queries start from a given seed node, and seemingly, for the small benchmark dataset, the Neo4j system is faster in accessing the properties related to this node. Also, the variance in system performance is relatively low in all cases except for the Neo4j system concerning the "3-hop neighborhood" query. As discussed in Section 6.3.3, the node degrees of the network used for the benchmark dataset approximately follow a long tail distribution. Depending on the degree of the randomly selected seed node, querying its neighborhood might be differently expensive. The Neo4j system seems to be especially prone to this in the case of the small benchmark dataset.

Compared to the performance results retrieved for the scale 0.1 benchmark dataset, the ranking changed significantly for the larger scale 1 dataset. Figure 6.16 shows the best-performing setups for this scale 1 grouped by benchmark query and system. The chart shows that the native PostgreSQL setup clearly outperforms all the other systems for all three benchmark queries. Also, performance

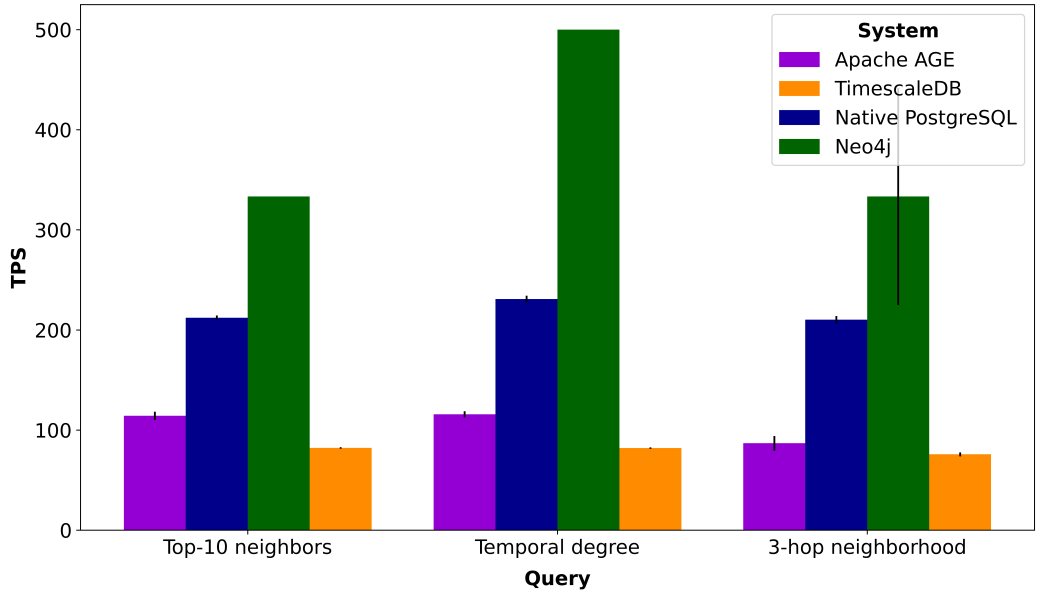


Figure 6.15: Performance comparison of the different systems for the scale 0.1 benchmark dataset. Results are grouped by benchmark query. Neo4j outperforms the other systems in all cases.

variance is low for all setups. Now, even the Neo4j system performs quite consistently for the 3-hop neighborhood query. By taking a look at Table 6.22, one can see that, compared to the scale 0.1 dataset, not the "index (node:id)", but using the index on the relationship timestamp leads to the best results for the Neo4j system. Seemingly, this database configuration leads to more robust performance results.

Figure 6.17 shows the performance overview for the benchmarks conducted on the scale 10 dataset. Even though the native PostgreSQL system still shows the best results, multiple differences can be observed compared to the scale 1 benchmark. First of all, the Neo4j setup is no longer in second place but is surpassed by the PostgreSQL-based TimescaleDB setup. Supposedly, PostgreSQL's efficient data-accessing methods play a significant role, especially for larger datasets. Nevertheless, the Apache AGE system does not benefit from these technical advantages. Given that Apache AGE internally uses a specific database layout that is not tailored to the use case of analyzing *dynamic* network data (see Table 6.14), these benefits might not come into action. Furthermore, it is again noticeable that performance variations are pretty large for the 3-hop neighborhood query in the case of the native PostgreSQL and the TimescaleDB setup. Again, this phenomenon might be related to the large variations in node degree, as explained above. Given that this type of query requires self-joins of the temporal edge table (see Appendix A.1), the size of the result set might become enormously large, and its size highly depends on the degrees of the involved nodes.

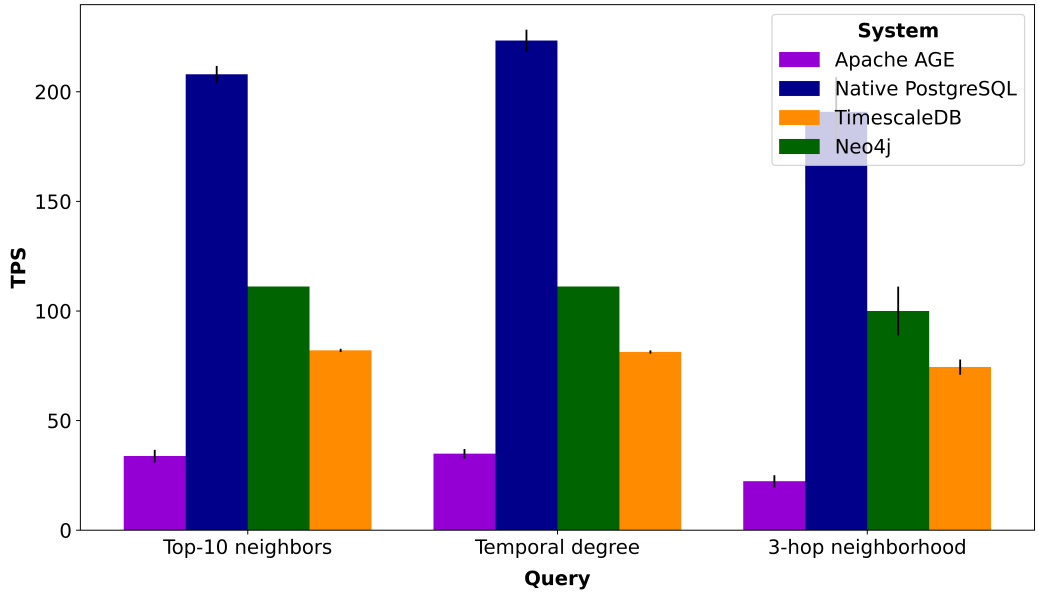


Figure 6.16: Performance comparison of the different systems for the scale 1 benchmark dataset. Results are grouped by benchmark query. PostgreSQL performs best.

The larger the degree of the nodes part of the 3-hop neighborhood, the larger the neighborhood becomes. This dependency might explain the large variations in system performance for the two PostgreSQL-based systems in the case of the neighborhood query.

Results by system

In the following, the results discussed in the previous section from an overview perspective are now examined in detail. For that, the derived benchmark metrics are discussed per system. For each system, including native PostgreSQL, TimescaleDB, Apache AGE, and Neo4j, their performance is compared across the tested queries and dataset scales. Further, for each setup, the best-performing data indexing strategy is discussed.

Native PostgreSQL-based: Figure 6.18 shows the benchmark performance of the native PostgreSQL system grouped by the different benchmark queries. For each scale and query combination, the best-performing data indexing strategy is shown. The performance is mostly around 200 TPS, and its variance is relatively low. A slightly decreasing performance of around 10 to 20 TPS for each step of increasing the benchmark dataset size can be observed. This behavior is consistent throughout the different benchmark queries and dataset scales, except for the scale 10 "3-hop neighborhood" query, which performs with only around 70 TPS and shows a variance of nearly

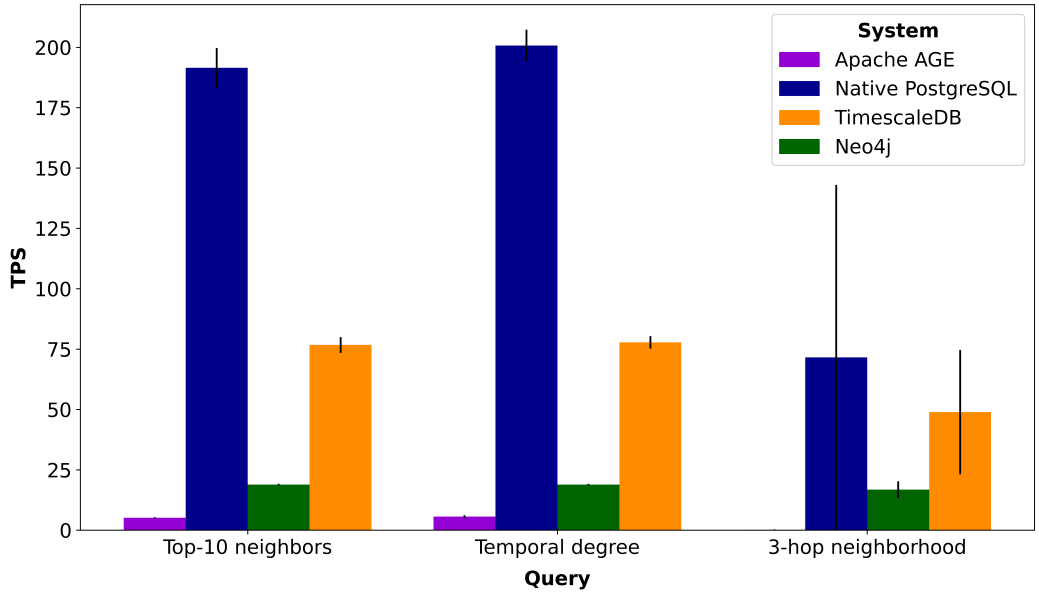


Figure 6.17: Performance comparison of the different systems for the scale 10 benchmark dataset. Results are grouped by benchmark query.

the same size. This high-performance variability is, as examined above, probably related to the highly skewed node degree distribution present in the leveraged benchmark dataset.

Next to the actual performance results, it might be interesting to investigate which indexing strategy led to these. For the native PostgreSQL system, the indices used for the best-performing tests with respect to the according dataset scale and benchmark query are given in Table 6.19. Except for the "3-hop neighborhood" query, the "index (source)" strategy leads to the best results independent of dataset scale. Seemingly, the overhead of having more than just the "source" column contained in the B-tree index outweighs the potentially gained performance improvements. Given that filtering the data based on an individual (source) node applies the most restrictive condition compared to edge label or timestamp filtering, it is probably most effective to have a lightweight index on this individual column that can be used to improve data access performance. Nevertheless, more complex indexing strategies seem to lead to the best results for the scales 1 and 10, and the "3-hop neighborhood" query. In these cases, using a multi-column B-tree index applied to the source node, edge type, and timestamp columns works better. Having the additional columns also considered during index creation, allowing to filter on them within the B-tree already, seems to outweigh the induced overhead.

TimescaleDB: A performance overview of the tests conducted for the TimescaleDB system is given by Figure 6.19. Again, results are grouped by the different benchmark queries, and each

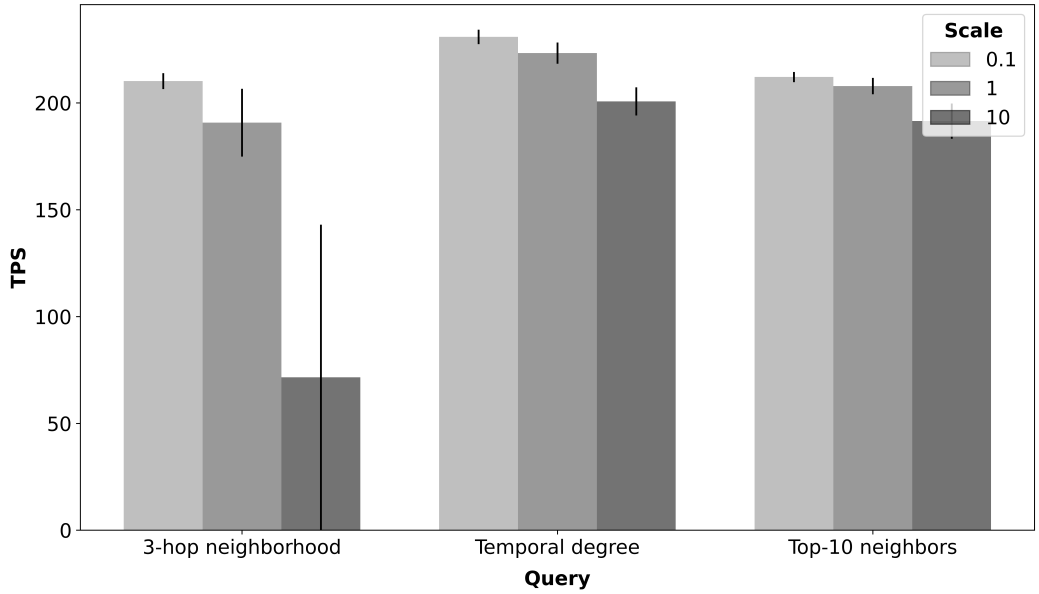


Figure 6.18: Performance comparison of the native PostgreSQL setup grouped by benchmark query and dataset scale. In most cases, the system's performance is around 200 TPS.

Table 6.19: Best-performing indices used for the native PostgreSQL setup. Results are separated by dataset scale and benchmark query. See Table 6.12 for a description of the different indices.

	Temporal degree	**3-hop neighborhood**	**Top-10 neighbors**
0.1	index (source)	index (source)	index (source)
1	index (source)	index (source,label,dt)	index (source)
10	index (source)	index (source,label,dt) & index (target)	index (source)

group contains the results of all three dataset scales. One can see that, in general, the TimescaleDB system performs worse compared to the native PostgreSQL setup, with the performance being, for the most part, around 80 TPS. Also, in most cases, performance variance is relatively low (< 10 TPS). The system shows significantly less performance only for the "3-hop neighborhood" query in combination with the scale 10 benchmark dataset. For this test, the observation of a low performance (≈ 50 TPS) combined with a high variance (≈ 25 TPS) is similar to the behavior of the native PostgreSQL system, which is plausible given that TimescaleDB is also built on top of PostgreSQL.

From Table 6.20, which shows the best-performing indexing strategy per benchmark for the TimescaleDB setup, one can see that indexing the source node column plays a crucial role. In most

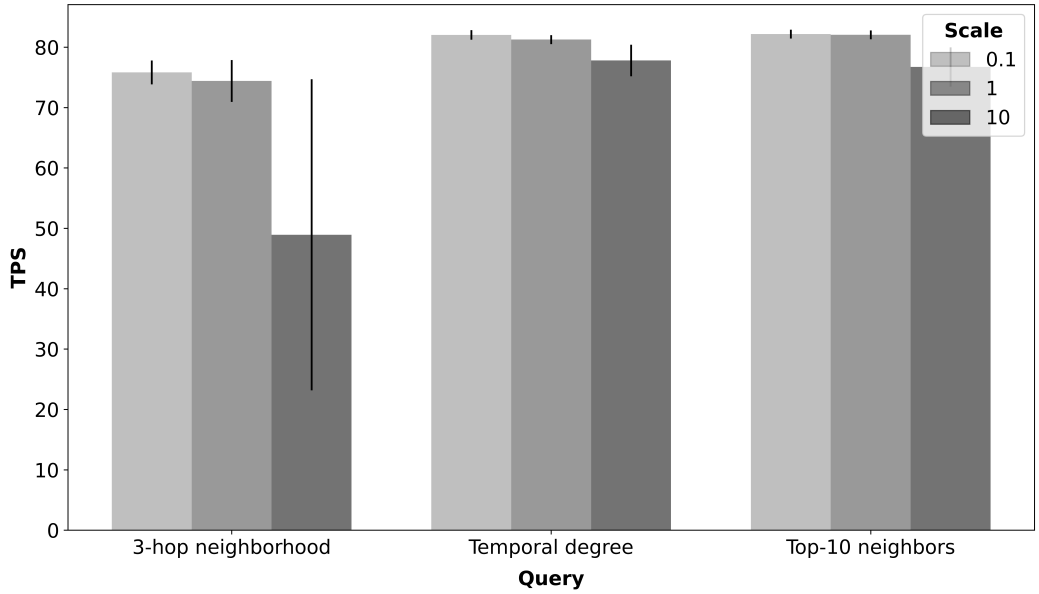


Figure 6.19: Performance comparison of the native TimescaleDB setup grouped by benchmark query and dataset scale. In most cases, the system's performance is around 80 TPS.

cases, indexing the source node and the relationship timestamp leads to the best results. Given that TimescaleDB internally partitions the data by time, it is required to include the timestamp column in all indices that span the entire hypertable. Interestingly, in various tests, the multi-column index, which also takes the edge type into account ("index (source,label,dt)"), performs best. Seemingly, having the edge-type information present within the index allows for faster filtering compared to the other indexing strategies in these circumstances.

Table 6.20: Best-performing indices used for the native TimescaleDB setup. Results are separated by dataset scale and benchmark query. See Table 6.13 for a description of the different indices.

	Temporal degree	**3-hop neighborhood**	**Top-10 neighbors**
0.1	index (source,dt)	index (label,dt)	index (source,label,dt)
1	index (source,dt)	index (source,dt)	index (source,dt)
10	index (source,label,dt)	index (source,label,dt)	index (source,label,dt)

Apache AGE: For the Apache AGE system, benchmark results are given in Figure 6.20. In general, one can observe that the system's performance decreases significantly with larger dataset scales. Even though the Apache AGE setup outperforms TimescaleDB for all three benchmark queries in the case of the scale 0.1 benchmark, its performance is much worse for the other dataset scales.

Still, the performance variance is low for all tests, not exceeding a value of 8 TPS. The worse performance in the case of the larger benchmark dataset compared to the TimescaleDB and native PostgreSQL setups might be related to the different database layout used internally by Apache AGE. As discussed above, the database's structure follows the idea of partitioning the graph data according to the different node and edge types. Nevertheless, the benchmarked queries are highly time-dependent, and fast filtering on the temporal dimension is therefore crucial. This different partitioning approach might cause inferior performance.

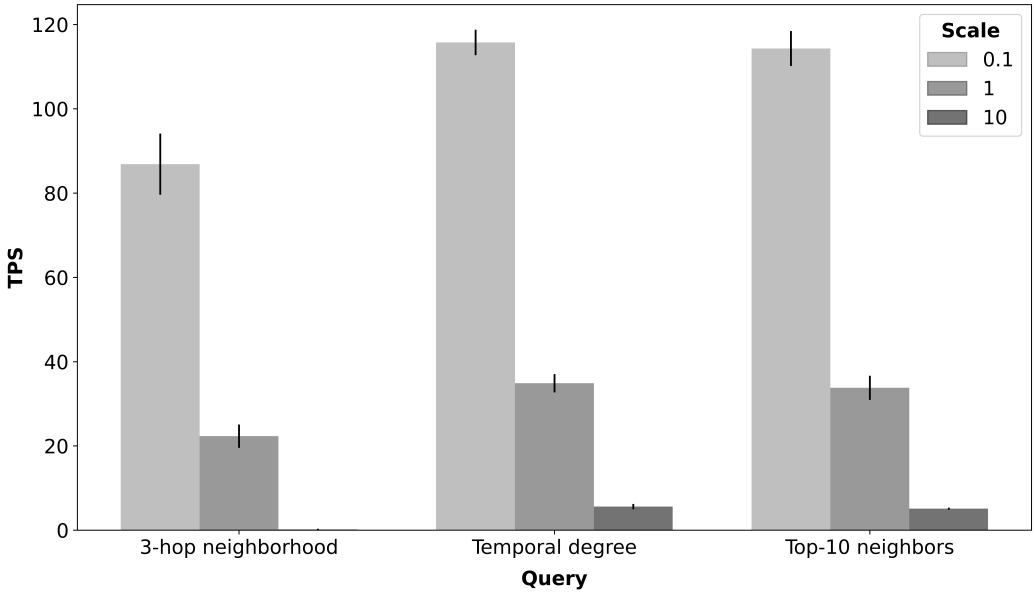


Figure 6.20: Performance comparison of the native Apache AGE setup grouped by benchmark query and dataset scale. As the amount of data increases, the performance decreases significantly.

Concerning the best-performing indexing strategies shown in Table 6.21 for the Apache AGE setup, one can observe that the two indices on the source and target node columns work well for the "Temporal degree" and "Top-10 neighbors" queries, whereas the multi-column index on the source node and the edge's timestamp works well for the "3-hop neighborhood" query. In all cases, indexing the source node helps to speed up the edge filtering, which is crucial for all the benchmarked queries.

Neo4j: From Figure 6.21, one can observe that the Neo4j setup outperforms all the other systems for the scale 0.1 benchmark dataset and also shows competitive results for the scale 1 dataset. Nevertheless, for scale 10, the performance is only between 15 to 19 TPS and, therefore, worse compared to TimescaleDB or the native PostgreSQL setup. Seemingly, Neo4j does not scale well for

Table 6.21: Best-performing indices used for the native Apache AGE setup. Results are separated by dataset scale and benchmark query. See Table 6.15 for a description of the different indices.

	Temporal degree	**3-hop neighborhood**	**Top-10 neighbors**
0.1	index (source) & index (target)	index (source,dt)	index (source) & index (target)
1	index (source) & index (target)	index (source,dt)	index (source) & index (target)
10	index (source,dt)	index (source,dt) & index (target)	index (source)

the tested benchmark queries and dataset. Further, except for the scale 0.1 "3-hop neighborhood" query, variance in performance is relatively low.

Apart from the actual performance results, it should be noted that the TPS values are calculated by summing up Noe4j's query result availability and consumption time. The complete time it takes the database client to consume the query results is taken into account. For a detailed explanation, we refer to Bowman (2023).

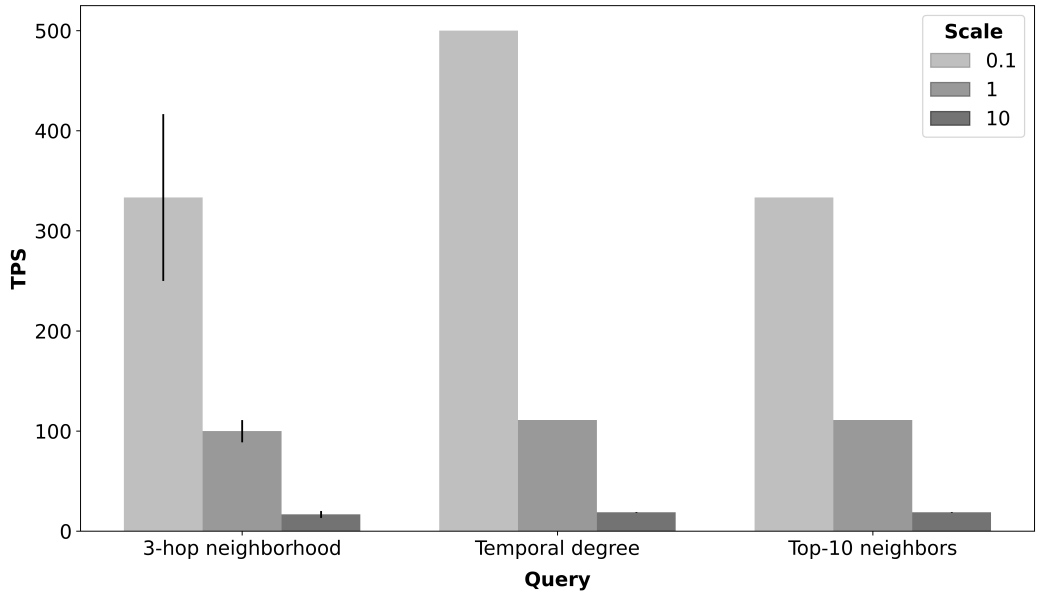


Figure 6.21: Performance comparison of the native Neo4j setup grouped by benchmark query and dataset scale. As the amount of data increases, the performance decreases significantly.

Furthermore, Table 6.22 shows the data indexing strategies which led to the best results for the Neo4j benchmark. It has to be noted that in multiple cases (e.g., scale 0.1 & "3-hop neighborhood"

or scale 10 & "Top-10 neighbors"), different indices led to very similar performance results with differences of less than 1 TPS. Still, only one index strategy is selected to be present in the table based on the highest median TPS value and lowest MAD value. If both values are the same, the first analyzed setup is selected. Nevertheless, these similar results make it hard to derive general rules on which data indexing method works well in which cases. At least, one might argue that indexing the node IDs and/or relationship timestamps works best in most cases.

Table 6.22: Best-performing indices used for the native Neo4j setup. Results are separated by dataset scale and benchmark query. See Table 6.16 for a description of the different indices.

	Temporal degree	**3-hop neighborhood**	**Top-10 neighbors**
0.1	index (node:id)	baseline	index (node:id),(rel:dt)
1	index (rel:dt)	index (node:id)	index (node:id)
10	baseline	index (node:id),(rel:dt)	index (rel:dt)

6.3.5 Summary and discussion

In this study, various temporal network data management systems have been tested. Thereby conducted tests aimed at finding a setup to analyze temporal heterogeneous network data in a performant way. Multiple graph queries, dataset scales, and data indexing strategies have been examined in line with the requirements discussed in Section 6.3.1. Further, related work and technical implementations are detailed in Sections 6.3.2 and 6.3.3, respectively. Overall, a customized setup based on native PostgreSQL showed the most promising results by performing best for all queries in cases of the two largest dataset scales. Therefore, this system is also recommended to be used for the EPINetz platform.

Nevertheless, the conducted benchmark does not claim generalizability, first and foremost, because it is designed with a focus on the specific use-case requirements in mind. It is not meant as a general temporal graph analytics benchmark. In this regard, a pre-selected set of systems is tested only. This set of systems does, for example, not include most proprietary systems or such that are not commonly used in academia. Furthermore, the benchmark results are merely evaluated based on the systems' query performance and other metrics, such as memory usage or CPU utilization, are not taken into account. Finally, several hardware constraints had to be considered while conducting the benchmark. Therefore, a graph dataset of a larger scale could not be tested.

Even though the goal of the conducted benchmark, to find a high-performant edge sequence-based data management system, is reached, overcoming the discussed shortcomings should be

part of future work. Further, more advanced graph analysis algorithms could be integrated into the benchmarking framework, such as those used for community detection or information diffusion.

6.4 Network-Based Trend Exploration

The study of trends in social media is highly relevant for many use cases. For example, from a business intelligence perspective, it is essential to track what the company's target group is interested in and how specific characteristics, e.g., environmental friendliness, of a product become more or less relevant over time. Also, in the field of political science, it is of great interest to study which political topics are discussed on social media, in *which context* and *how they evolve*. For these use cases, the temporal analysis of long-term trends extracted from social media data is crucial. Even though Section 4.1 already presents a framework for detecting and analyzing long-term social media trends, an actual system for the interactive exploration of the derived analysis results is still missing. Therefore, in this section *TrendTracker*, a web application[20] for the network-based and temporal exploration of long-term social media trends, is presented. Topical trends, represented as a series of hashtag co-occurrence networks, can interactively be explored while the user is provided with detailed trend analysis insights. This approach has several benefits compared to alternative trend visualization and exploration methods, such as ranked lists of trending keywords, as it provides the user with additional context-sensitive information. To showcase the TrendTracker application, we leverage a Twitter dataset of German political actors and demonstrate the system's capabilities in various ways. For example, the user is able to investigate a single trend from multiple perspectives, such as the trend's temporal development over time, including its topical shifts and changes in popularity. Also, given the network-based trend visualization, the user can intuitively understand the different facets of a trend and how these are interrelated. Thereby, individual hashtags and relationships can be tracked over time. Furthermore, the TrendTracker application allows the user to compare trends. This way, differences in the trends' temporal evolution or topical alignment can be uncovered.

Reference: This section is based mainly on the following peer-reviewed publication. The TrendTracker application was developed in collaboration with Johannes Sindlinger and Marina Walther in the context of their software practicals at the Heidelberg University's Data Science Group:

[20] TrendTracker: https://trend-tracker.ifi.uni-heidelberg.de (accessed 2023-10-23)

John Ziegler, Johannes Sindlinger, Marina Walther, and Michael Gertz. TrendTracker: Temporal, network-based exploration of long-term Twitter trends. In *Proceedings of the 2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Forthcoming.

The remaining part of this section is structured as follows: First, in Section 6.4.1, the contributions of this work to the Twitter trend visualization discipline are highlighted. Subsequently, in Section 6.4.2, related work is discussed and compared to the capabilities of the TrendTracker system. Further, in Section 6.4.3, the background, including the leveraged dataset, the long-term trend detection methodology and the technical implementation, is covered. Based on that, the analysis workflows made available to the user via the web application are described in Section 6.4.4. Finally, Section 6.4.5 gives a concise summary of this section.

Figure 6.22: Landing page of the TrendTracker web application. Descriptive explanations facilitate the onboarding of the user; Screenshot taken from the TrendTracker web application on 8 August 2023

6.4.1 Contributions

Given long-term trends extracted from social media data, from an end-user perspective, a suitable visual representation of these is crucial. Simple lists of trending topics, as shown in Figure 4.1, are insufficient for several reasons, as described by Bhulai et al. (2012). They do not clearly indicate the relative importance of topics, and the topics' temporal evolution is not represented at all.

Finally, multiple topics in these lists might be related and should therefore be clustered for a more descriptive representation as well (Bhulai et al., 2012). The TrendTracker system overcomes these shortcomings by providing the user with trend scores over time and by visualizing topics in the form of intuitively understandable temporal networks, giving the user additional context-sensitive information. Further advanced trend analysis features are provided. In summary, our contributions to the fields of Twitter trend visualization and analytics are manifold:

1. The studied trends are visualized as temporal hashtag co-occurrence networks. This visualization method allows users to easily grasp the context of the respective trend and how different aspects of the topic are related.

2. A series of temporal network snapshots represents the long-term temporal development of a trend. Thereby, users can study how a trend, i.e., its focus or relevant aspects, changes over time.

3. Additional trend analysis insights are provided to the user, such as the trend's popularity over time or its relative trend relevance score.

4. Individual hashtags and their relationships can be highlighted in the trend networks and can be tracked over time, allowing to investigate their dynamics regarding the explored trend.

5. Multiple concurrent trends can be explored simultaneously in a comparative manner. Thereby, temporal and topical differences between trends can be uncovered.

Regarding the terminology used, in our context, *long-term* trends must be understood in contrast to short-lived media content. We do not deal with breaking news but focus on topics prevalent in social media over a long period. We refer the interested reader to Section 4.1.2 for a detailed discussion on the used terminology.

6.4.2 Related work

Most similar to our work are related approaches from the field of Twitter trend visualization. Several systems have been presented for that in the past, most notably the work by Doshi et al. (2017). They propose a system called *Tweetanalyzer* that allows users to explore real-time trends extracted from Twitter. As trends, they use the most frequently occurring hashtags and usernames. Respective statistics are presented in the form of bar charts. Further, tweets related to named trends are displayed on a map to visualize their geographic location. In contrast to our TrendTracker system, they do not provide the user with network-based exploration capabilities, nor do they explore the long-term development of detected trends but are instead focused on their real-time

occurrence. Further, one of the early works in the field of trend visualization based on Twitter data was published by Bhulai et al. (2012). To visualize trending topics in the most informative way, they use so-called "dynamic squarified treemaps" that allow them to not only show the actual trends based on hashtags and terms but also incorporate information regarding the speed and acceleration of the trend development. Additionally, they cluster related topics for them to be more descriptive. Still, they do not incorporate information regarding the long-term development of trends and do not provide the user with network-based exploration capabilities. Further, Wanner et al. (2012) propose another visualization technique to track Twitter topics. Using equal-sided triangles to visualize the occurrence of tweets within a timeline allows them to represent the unevenly distributed time series data concisely. The color of named shapes indicates the sentiment of respective tweets. By using multiple timelines at once, the user can also compare different topics or rather different time windows and is, therefore, able to investigate a topic's temporal development. Even though the work of Wanner et al. (2012) is dealing with the challenge of visualizing Twitter data, their focus is not on network-based methods and does not explicitly target the tracking of long-term trends as done by our TrendTracker system. More recently, Stojanovski et al. (2014) present a web application called *TweetViz* to explore Twitter data visually. They focus on user- and hashtag-oriented visualizations but do not deal with exploration capabilities for trends. Similarly, Kant et al. (2020) publish a Python package called *TTLocVis* to work with Twitter data. Leveraging their package to visualize detected topics over time also allows to track their temporal prevalence. Still, they do not provide network-based exploration capabilities, nor do they deal with topical trends. The same is true for the *RIVA* social media analysis platform proposed by Wu et al. (2017). By counting the occurrence of hashtags, they are able to detect trends and visualize these in the form of pie charts. Nevertheless, further exploration capabilities are not provided.

6.4.3 Background

This section describes the German political Twitter dataset and the analysis methodology leveraged to showcase the TrendTracker application. For all the data analysis details, we refer the interested reader to the respective Section 4.1. Next to the data-related background, a concise summary of the system's technical implementation is given.

Dataset

The used dataset is based on tweets from German political actors as provided by the EPINetz Twitter Politicians Dataset 2021 König et al. (2022) and covers data from January 2021 until July 2022, in total approximately 1.8 million tweets. To retrieve the raw tweets, we rely on the Twitter search API v2 and extract timestamped information about the occurrence and co-occurrence of

hashtags. In line with the works of Asur et al. (2011) and Budak et al. (2011), we treat hashtags as representatives of topics and do not apply any additional topic extraction technique. In general, the methodology is not restricted to this dataset but can be applied to a broad set of use cases that deal with temporal occurrences of keywords.

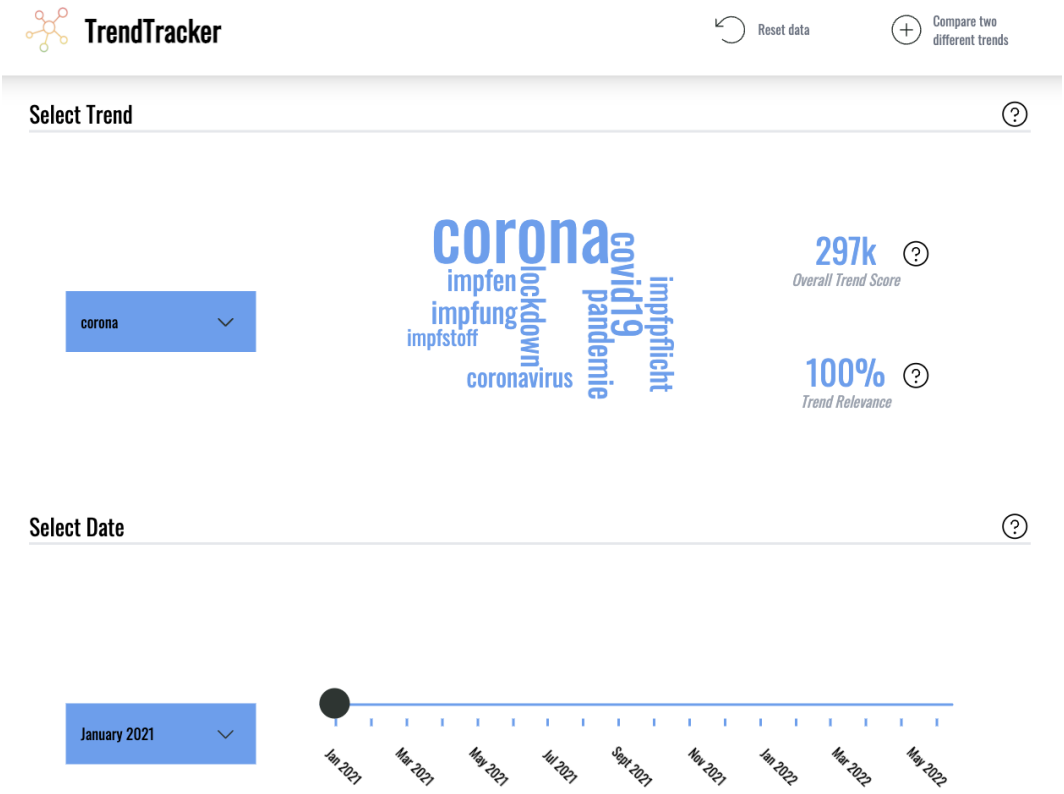


Figure 6.23: Settings where users can configure the trend and time period they want to explore; Screenshot taken from the TrendTracker web application on 8 August 2023

Trend detection

Taking the timestamped information about the usage of hashtags as described in the previous section, we construct hashtag co-occurrence networks, each covering the data of one month. To remove noise and focus on the most expressive hashtags, we remove all nodes with a degree lower than the network's median. Degrees follow a power law distribution. Therefore, we leverage the median as defined by Newman (2005). Additionally, edges are weighted by Pointwise Mutual Information to strengthen more semantically expressive relationships between hashtags (Role and Nadif, 2011). In the next step, to cluster related topics as done by Bhulai et al. (2012), we apply the Leiden community detection algorithm to each network (Traag et al., 2019). The induced

subgraphs of the ten most central hashtags of found communities make up the trend networks that the user can explore. To track topics over time, i.e., the communities of hashtags across the network snapshots, we apply the algorithm proposed by Lorenz et al. (2017). As trend scores, the community's cumulative sum of hashtag occurrences is taken. Finally, for the word clouds describing the complete trend, we leverage the joined networks of the temporal communities per trend and take the ten nodes with the highest PageRank scores as representatives per trend. For example, Figure 6.23 shows the extracted word cloud related to the COVID-19 trend. For all networks, node sizes are adjusted according to their normalized PageRank centrality (see Figure 6.25). Computations are done using the `igraph` network analysis library (Csárdi and Nepusz, 2006). On the TrendTracker website, the ten most prevalent trends over time are visualized.

Implementation

The TrendTracker web application is built with the `SvelteKit`[21] framework, along with `D3`[22] and `Chart.js`[23], that are used for the interactive visualizations. Data about the long-term trends is retrieved from a Python REST API. Figure 6.24 gives an overview of the software architecture behind the TrendTracker system.

6.4.4 Analysis workflows

By visiting the TrendTracker website, the user first reaches the landing page as shown in Figure 6.22. There, an introductory text explains the purpose of the application and facilitates the user's onboarding. Continuing the website visit, by scrolling down or clicking on the "Explore trends" link, the user reaches the actual data exploration part of the web app and is led to the analysis settings (see Figure 6.23). For explanatory purposes, most application features come with a tooltip that is indicated by an encircled question mark.

Single trend exploration

The mentioned analysis settings (see Figure 6.23) allow users to select a given trend and time window. Thereby, additional information and statistics are provided for the trend selection, such as a word cloud visualizing the trend, the overall trend score and its trend relevance. This relevance score is derived by comparing the trend's popularity to the prevalence of the other trends. It, therefore, acts as a relative importance indicator, which is, according to Bhulai et al. (2012), a

[21] SvelteKit • Web development, streamlined: `https://kit.svelte.dev` (accessed 2023-10-24)
[22] D3 by Observable | The JavaScript library for bespoke data visualization: `https://d3js.org` (accessed 2023-10-24)
[23] Chart.js | Open source HTML5 Charts for your website: `https://www.chartjs.org` (accessed 2023-10-24)

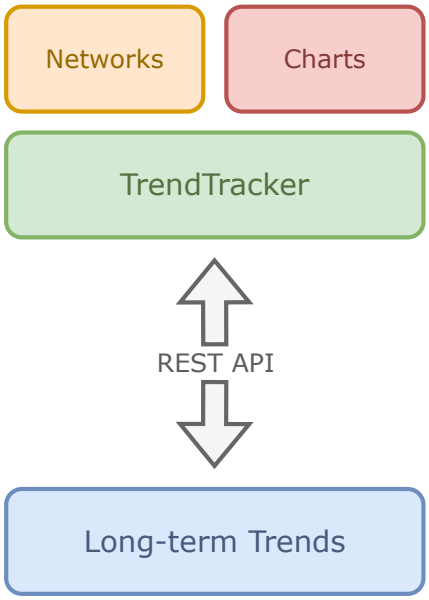


Figure 6.24: Illustration of theTrendTracker system's software architecture. The data about long-term trends is retrieved from a REST API and visualized in the form of interactively explorable networks and charts.

significant improvement compared to ranked lists. Furthermore, for the date selection, next to the dropdown list, the right-sided slider can be used to pick the analysis time window.

According to the selected trend and set time window, the trend network (see Figure 6.25) and temporal trend scores (see Figure 6.26) are presented to the user. These charts allow a network-based and temporal exploration of the trend. Also, the built-in reactivity of the application leads to an immediate update of the statistics if the user adjusts the settings. Once a user has scrolled past the configuration section, updating the settings is possible via the sticky header (see Figure 6.28). Further, the layout of the trend network can be adjusted to the user's preferences by dragging the respective nodes in the network.

Node and edge tracking

Not only can the users adjust the positioning of nodes in the trend networks, as mentioned above, but they can also track these across multiple network snapshots. By clicking on a node or a link, it is highlighted. If the user then changes the time window of the trend, the node/edge remains highlighted in the network in case it is still present. Thereby, the temporal tracking of individual aspects of a trend (i.e., represented by a hashtag) is facilitated. This way, the user can check whether some hashtags gain or lose importance for the trend over time. Besides the manual tracking of

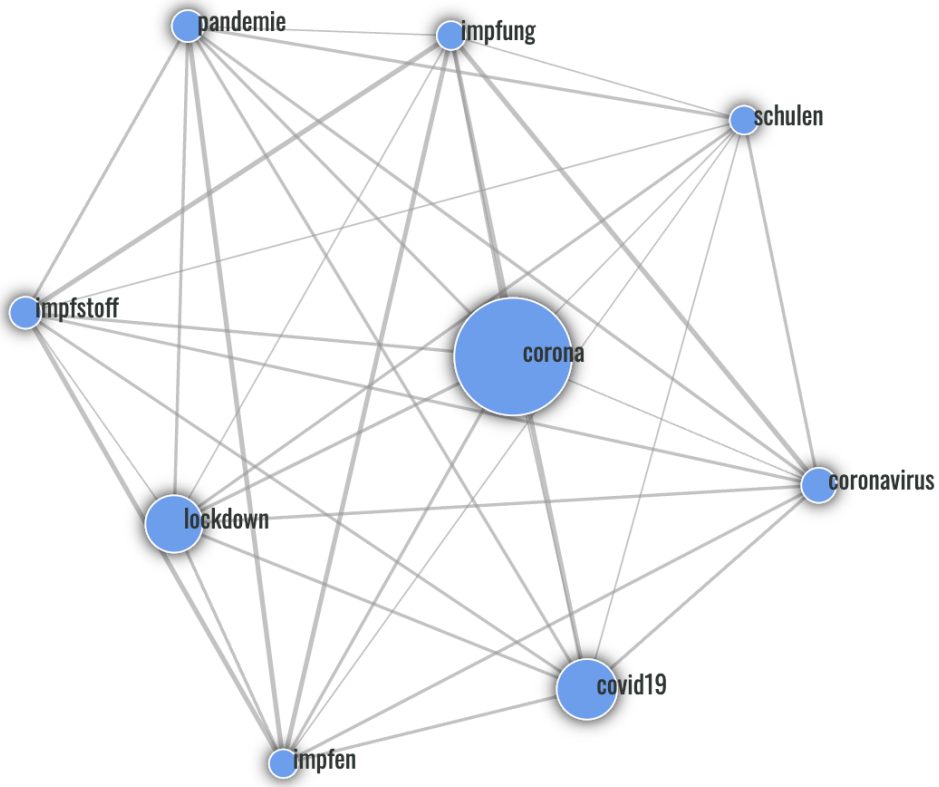


Figure 6.25: Exemplary COVID-19 trend network. To a large portion, the network consists of vaccination-related hashtags and covers discussions around the lockdown as an action to fight the pandemic; Screenshot taken from the TrendTracker web application on 8 August 2023

the highlighted entity, a line chart of temporal node/edge weights is also displayed. It indicates the evolution of the importance of the highlighted entity. Figure 6.27 gives an example of such a line chart. It shows the node weights of the *corona* hashtag occurring in the COVID-19 trend networks.

Trend comparison

Some use cases also benefit from a direct comparison of two trends. Therefore, the TrendTracker application offers the capability to conduct two analyses in parallel. For both analyses, the trend and time window can be set individually. This way, the users are not limited to comparing the *same* trend at arbitrary points in time, but they can also contrast the state of *different* trends at the same or different times. Figure 6.28 shows such a comparison which contrasts the COVID-19 trend as of January 2021 with the state of the EU trend in March 2022. Generally, while the users

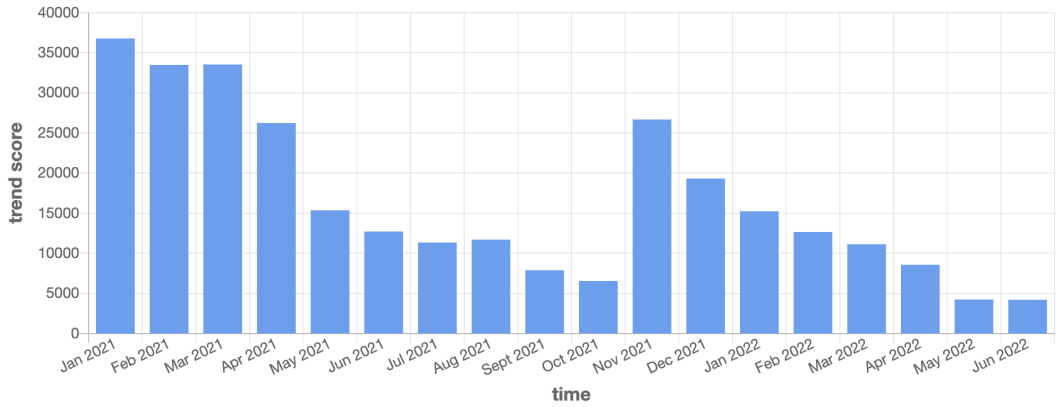


Figure 6.26: Trend scores of the COVID-19 trend. Two peaks in popularity are visible during January 2021 and November 2021; Screenshot taken from the TrendTracker web application on 8 August 2023

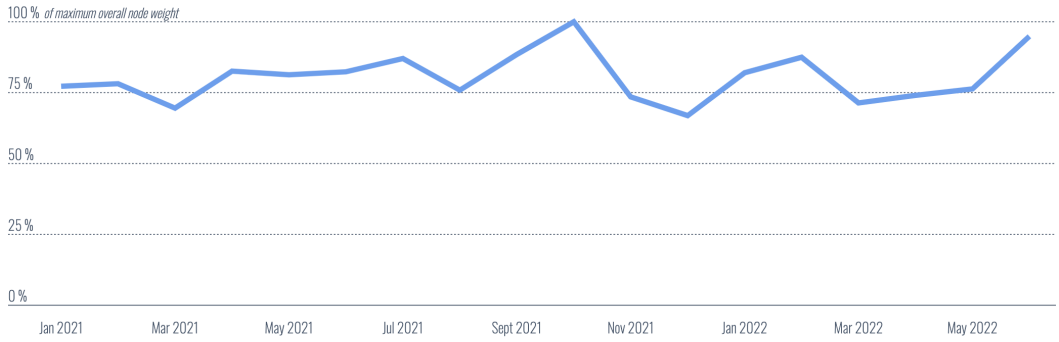


Figure 6.27: Temporal node weights of the *corona* hashtag as part of the COVID-19 trend networks; Screenshot taken from the TrendTracker web application on 8 August 2023

conduct a trend comparison, the same features, such as node/edge highlighting and the temporal trend scores, are still available and are applied to both investigated trends.

6.4.5 Conclusion

This section presents the TrendTracker web application to explore long-term social media trends. Its capabilities are demonstrated based on Twitter data of German political actors collected from January 2021 until July 2022. Users of the TrendTracker system can explore the temporal evolution of long-term prevalent topics in various ways. Especially the network-based visualizations give the user context-sensitive exploration capabilities, which is a significant benefit compared to existing Twitter trend visualization approaches. Additional features of the web application,

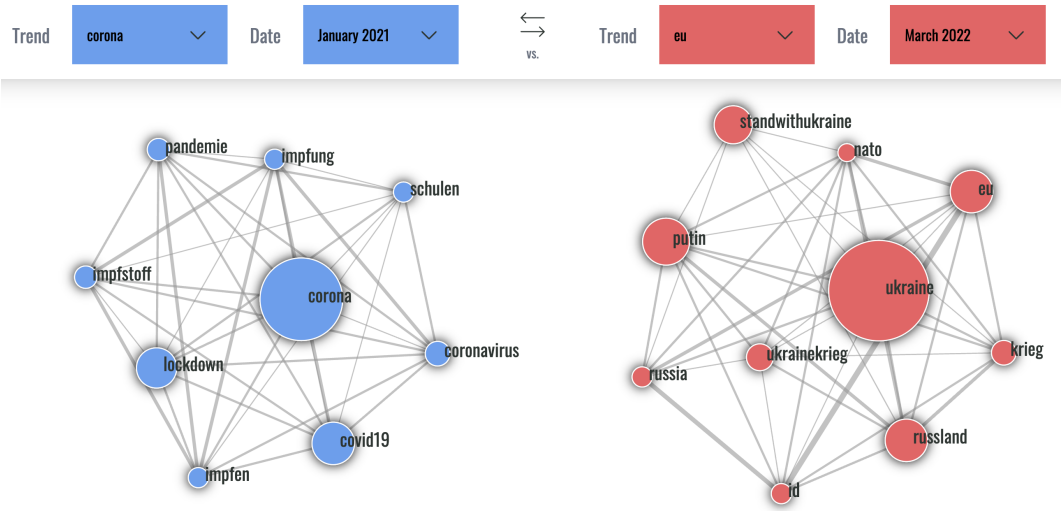


Figure 6.28: Exemplary comparison of two trends. Both the trend and the time window can be adjusted individually; Screenshot taken from the TrendTracker web application on 8 August 2023

such as the temporal trend scores, the trend comparison functionality or the entity highlighting, complement the application's capabilities.

7 Conclusion

In today's fast-evolving and highly connected world, keeping track of all changes, developments, and interdependencies becomes increasingly difficult. To some extent, content shared on various media platforms reflects these dynamics and the connectedness. Even though this content does not cover the real world in all its details and might even provide a biased perspective on many subjects, given its enormous influence not only for each person individually but also for our society as a whole, studying and understanding media-related phenomena becomes ever more critical. Derived findings might affect various disciplines, such diverse as sociology, politics, marketing, or communication. However, deriving such findings is not straightforward, and various challenges remain no matter which media phenomenon one wants to investigate, be it trends, conversations, or the spread of information in general. These challenges include the need for an efficient and performant data management system, the appropriate modeling of collected media data, and finally, gaining insights from the modeled data regarding the investigated research questions. The effort of this work is aimed at making contributions to named challenges.

7.1 Summary and Contributions

In this work, the analysis of media data is approached from two different perspectives in particular: dynamism and connectedness. This focus is reflected by the leveraged data modeling, which is built on concepts known from the field of temporal networks that allow taking the data's interdependent nature and time sensitivity into account. In sum, our effort led to various methodological and technical contributions. To the best of our knowledge, combined, they represent the most comprehensive network-based media analytics framework. All novelties and contributions are detailed in the following.

7.1.1 Methodological contributions

Our methodological contributions start with the developed data analysis model presented in Chapter 3. It is based on temporal network concepts, specifically tailored to the use case of media

analytics and comes with several properties that facilitate extracting informative insights. These properties include different data granularities, which allow the analysis of the data based on differing topological structures – from individual nodes and edges up to the complete network – as well as based on varying temporal resolutions. Also, capabilities to project the network via user-defined network paths are part of the proposed model. Together, the mentioned properties provide the user not only with a lot of modeling flexibility and powerful data analysis tools but also allow for use case-specific data investigations. Further, regarding the network-based modeling of the data, the model is specifically designed to integrate heterogeneous data from different sources, which is particularly important for analyzing media data. The data is structurally processed and represented homogeneously through the defined schema and the resulting topology of the constructed network. In more detail, the model's extensibility is achieved by its robust design based on document-centric networks, which are further enriched by extracted entities. This leads to latent relationships between otherwise unconnected documents and, therefore, to a structural harmonization of the analyzed data. Next, the data's and therefore, also the network's time-dependent nature is incorporated into the model. Specifically, we rely on constructing temporal network snapshots and go into great detail concerning appropriate network sampling techniques. Not only do we consider the commonly leveraged fixed time window sampling but we also examine alternative approaches, such as volume-based techniques or techniques based on use case-specific metrics, e.g., the duration of trends (see Section 3.3.1). In sum, we contribute to the temporal and network-based modeling of heterogeneous media data, including methodology for advanced media analytics.

Based on this model, we conducted various studies about different social media phenomena, and developed novel methods along the way. In this regard, Chapter 4 details our contributions towards a better understanding of social media trends. First, we studied long-term trends, which are prevalent over larger periods compared to commonly known short-lived trends that are only popular for a few days (see Section 4.1). Specific techniques for detecting such trends are developed, and advanced visualizations support the communication of derived insights. Further, Section 4.2 investigates the actor-centric perspective concerning trends. For that, the actor-networks underlying trends are studied. Methods for detecting different trend phases, determining trend durations, and for a trend-aware temporal sampling of studied networks are developed. In particular, we design a framework for the network-based analysis of actor interactions, including the analysis across similar trends, called inter-trend analysis, as well as the analysis across the individual trend life cycles, called intra-trend analysis. More broadly speaking, our work on trends paves the way for a tighter connection between the trend and network analysis disciplines.

Next to the studies on trends, we leveraged the proposed model for the analysis of conversations. Respective results are outlined in Chapter 5. To derive these results, we tackled the analysis of conversational data by taking its structure, content, and dynamics into account, which is a more holistic approach compared to that of existing studies. In detail, we first analyzed the users' posting activity and developed an empirical exponential saturation model. Further, a volume-based sampling technique for the adaptive creation of temporal network snapshots was developed by building on this posting model. This sampling technique allowed to further analyze the topological evolution of extracted conversation networks. We employed our novel temporal Wiener index metric for that. Finally, we derived insights into the semantic changes of a conversation by analyzing the evolution of hashtag "usage". Together, this leads to a more comprehensive perspective on online conversations compared to existing approaches. The developed methods allow for incorporating the structural dimension based on actor-centric and network-topological information, the temporal dimension based on the network's evolution, and the semantic dimension based on the modeled content.

7.1.2 Technical contributions

In contrast to the aforementioned chapters, in Chapter 6, the technical contributions of this work are detailed. This includes the detailed examination of the unique temporal media dataset that is leveraged through this thesis, such as for the study of long-term trends in Section 4.1 or the EPINetz platform (see Section 6.2). For the dataset's description, a large emphasis is placed on its statistics, network schema, and covered dynamics. When it comes to comparable media datasets, specifically for the German political media landscape, to the best of our knowledge, the dataset presented in this work is unique in its scale, i.e., number of collected documents, diversity, i.e., the covered media platforms, and high data quality that is derived by building on a list of curated user accounts.

Next, the EPINetz platform is presented. It provides an interface for the temporal and network-based exploration of the mentioned dataset – capabilities that no existing media analytics platform provides in such a way. Users can learn about ongoing and historic political topics while improving their media literacy skills. Therefore, we make contributions to the field of computational political science not only in a technical sense but also from an educative perspective.

Furthermore, we contribute to benchmarking graph data management and analysis systems, as developing the EPINetz platform required testing various systems and technical configurations. Overall, the results show that a customized system setup based on PostgreSQL works best for our

requirements. All the temporal graph analysis benchmark results are presented and discussed in Section 6.3.

Finally, the TrendTracker application showcased in Section 6.4 constitutes contributions to an improvement of trend visualization and exploration techniques. In contrast to simple trend representations like lists, the TrendTracker application provides additional features that help its users to better understand the relative importance of trends, their dynamics, and how they are interrelated.

7.2 Limitations and Outlook

Even though this thesis includes several contributions to different research disciplines, such as trend research, online conversation analysis, or graph analysis benchmarking, it does not come without limitations. These limitations, together with potential extensions, are discussed in the following.

7.2.1 Methodological shortcomings

Methodologically, some weaknesses and, thus, opportunities for future work can be identified. With regard to the network-based media analytics model leveraged for the studies of this thesis, a few shortcomings exist. First, the network dynamics are modeled using a simple snapshot-based approach. Even though this approach captivates with its simplicity and is sufficient for a diverse range of use cases, it might reach its limits for others. In the current model, interlinks connecting nodes of different snapshots are not considered. Therefore, due to the temporal discretization of the network, some information is lost, which might need to be preserved in specific scenarios where, for example, continuous temporal network metrics are required. In this sense, gaining insights into temporally far-reaching dependencies is currently only possible by adjusting the snapshot sampling accordingly. However, one must note that an appropriate temporal sampling of the network can minimize the mentioned information loss of the current model. We examined various sampling techniques in great detail throughout this work (see Section 3.3.1).

Furthermore, by now, the data's temporal information is incorporated in the form of timestamps. However, some dynamics might be better modeled using interval graphs, especially if links are active for specific periods instead of only a given point in time. An extension of the current model could go in this direction. Also, currently, the data modeling and analysis efforts are primarily directed towards the data's temporal and topological properties and less so towards the analyzed content itself. Although entities, such as hashtags or usernames, are considered in the proposed

network model, more advanced content analyses, e.g., by using NLP methods, are currently not conducted. Such analyses might focus on a better semantic understanding of the content or on analyzing its sentiment. Together, future extensions should extend the holistic perspective on the analyzed data by taking a combination of temporal, topological, and content properties into account.

Also, concerning the mentioned content, our efforts are currently limited to textual data, e.g., text extracted from social media posts or news articles. Data of other modes, such as images or videos, are not analyzed. Even though analyzing such content types requires different methods, such as those from the computer vision discipline, our holistic media analysis framework would greatly benefit from such multi-model data analysis. In this regard, many possible research questions could be approached.

7.2.2 Potential extensions

As mentioned in the previous section, current methodological limitations also come with opportunities for future research efforts. A few additional extensions can be outlined next to the already mentioned potential extensions. First, one might adapt the temporal and network-based model to study data of non-media domains as well, e.g., law or business. In this thesis, we demonstrated the model's applicability in various media-related contexts. Even though we argue that its core concepts can be applied to other domains as well, this still needs to be proven. Furthermore, the data analyzed in this thesis was primarily collected from social media, Twitter in particular, and online news sources. This selection leaves many media data sources untouched, such as other social media platforms or even offline media sources. A comparison of analysis results for different platforms or even between online and offline media presents only one possible future extension of our work. Also, in this work, we blended the worlds of network and time series analysis, e.g., by studying trends in Chapter 4. We expect a treasure trove full of additional potential extensions lying at the intersection of these two research disciplines.

Apart from the mentioned methodological and data-focused extensions, the technical contributions of this work might also be extended. Starting with the conducted benchmark of the temporal network data analysis systems, given that only simple queries were tested by now, the systems might also be compared based on more complex queries. For these types of queries, an efficient implementation is crucial. Therefore, varying implementations of the same query might be tested as well. Nevertheless, the insights gained from the current benchmark can already be used to build a general-purpose temporal network data analysis system. The found data indexing and partitioning strategies can be leveraged for that. By providing built-in network analytics capabilities and aiming

for a good developer experience, academic or commercial users might find such a system helpful for their network science projects. Similarly, the built EPINetz platform can be extended. For example, integrating the capabilities of the TrendTracker application into the existing EPINetz platform would provide additional exploration tools and trend information to the user – in Section 6.4, we already examine respective capabilities. Similarly, features for the self-service analysis of individual online conversations might be valuable extensions. Also, the realized analysis platform might be adjusted to work for other application domains. In a business environment, it could, for example, be used to analyze operational data from a temporal, network-based perspective.

Clearly, we only touched the surface of possible applications. Our developed approaches provide a solid foundation for more temporal, network-based analytics projects to come, and the road ahead is full of opportunities.

A Benchmark Queries

In the following, the queries used for the benchmarking discussed in Section 6.3 are provided. Various temporal network data management and analysis systems are compared in the respective benchmark. Accordingly, the following queries can be used to derive temporal, network-based properties from analyzed data. Given that not all tested systems "understand" the same query language, multiple versions of the queries are provided: based on SQL in Section A.1, based on a combination of SQL and Cypher in Section A.2, and solely based on Cypher in Section A.3.

A.1 SQL Queries

Listing A.1: Temporal degree

```
SELECT te.timestamp::date AS day, COUNT(*)
FROM temporal_edge AS te
WHERE te.type = given edge type
   AND te.source_node = given source node
   AND te.timestamp BETWEEN given start timestamp AND given end timestamp
GROUP BY day
ORDER BY day ASC
```

Listing A.2: Top-10 neighbors

```
SELECT te.target_node, COUNT(*)
FROM temporal_edge AS te
WHERE te.type = given edge type
   AND te.source_node = given source node
   AND te.timestamp BETWEEN given start timestamp and given end timestamp
GROUP BY te.target_node
ORDER BY COUNT(*) DESC
LIMIT 10
```

Listing A.3: 3-hop neighborhood

```
SELECT DISTINCT ON (te3.target_node) te3.target_node
FROM temporal_edge AS te1, temporal_edge AS te2, temporal_edge AS te3
WHERE te1.type = given edge type 1
   AND te2.type = given edge type 2
   AND te3.type = given edge type 3
   AND te1.source_node = given source node
   AND te1.target_node = te2.source_node
   AND te2.target_node = te3.source_node
   AND te1.dt BETWEEN given start timestamp AND given end timestamp
   AND te2.dt BETWEEN given start timestamp AND given end timestamp
   AND te3.dt BETWEEN given start timestamp AND given end timestamp
```

A.2 SQL/Cypher Queries

Listing A.4: Temporal degree

```
SELECT *
FROM cypher(graph, $$
   MATCH (n1 {__id__: given source node})-[r: given edge type]->(n2)
   WHERE check_between(r.timestamp, given start timestamp, given end timestamp)
   RETURN dt_trunc('day', r.timestamp), COUNT(*)
$$) AS (day agtype, count agtype)
```

Listing A.5: Top-10 neighbors

```
SELECT *
FROM cypher(graph, $$
   MATCH (n1 {__id__: given source node})-[r: given edge type]->(n2)
   WHERE check_between(r.timestamp, given start timestamp, given end timestamp)
   RETURN n2, COUNT(*)
   ORDER BY COUNT(*) DESC
   LIMIT 10
$$) AS (target_node agtype, count agtype)
```

Listing A.6: 3-hop neighborhood

```
SELECT *
FROM cypher(graph, $$
```

```
   MATCH (n1 {__id__: given source node})
      -[r1: given edge type 1]->(n2)
      -[r2: given edge type 2]->(n3)
      -[r3: given edge type 3]->(n4)
   WHERE check_between(r1.timestamp, given start timestamp, given end
       timestamp)
      AND check_between(r2.timestamp, given start timestamp, given end
          timestamp)
      AND check_between(r3.timestamp, given start timestamp, given end
          timestamp)
   RETURN DISTINCT n4
$$) AS (target_node agtype)
```

A.3 Cypher Queries

Listing A.7: Temporal degree

```
MATCH (n1)-[r]->(n2)
WHERE type(r) = given edge type
   AND n1.id = given source node
   AND r.dt >= given start timestamp AND r.dt <= given end timestamp
RETURN datetime.truncate('day', r.timestamp), COUNT(*)
```

Listing A.8: Top-10 neighbors

```
MATCH (n1)-[r]->(n2)
WHERE type(r) = given edge type
   AND n1.id = given source node
   AND r.dt >= given start timestamp AND r.dt <= given end timestamp
RETURN n2, COUNT(*)
ORDER BY COUNT(*) DESC
LIMIT 10
```

Listing A.9: 3-hop neighborhood

```
MATCH (n1)-[r1]->(n2)-[r2]->(n3)-[r3]->(n4)
WHERE type(r1) = given edge type 1
   AND type(r2) = given edge type 2
   AND type(r3) = given edge type 3
   AND n1.id = given source node
```

```
   AND r1.timestamp >= given start timestamp AND r1.timestamp <= given end
       timestamp
   AND r2.timestamp >= given start timestamp AND r2.timestamp <= given end
       timestamp
   AND r3.timestamp >= given start timestamp AND r3.timestamp <= given end
       timestamp
RETURN DISTINCT n4
```

B Benchmark Results

The following table shows for each benchmark run the respective tested system, dataset scale, and used indices, along with the retrieved performance results. For example, the data shown in the first row means that the Apache AGE system was tested with the scale 0.1 benchmark dataset for the "3-hop neighborhood" query (see Listing A.6) with a GIN index used on the "properties" column. This setup reached a median performance of 6.4 TPS with a MAD of 3.41.

Table B.1: Complete list of benchmarking results. TPS values are rounded to two decimals.

System	**Scale**	**Benchmark query**	**Indices**	**TPS** (median)	**TPS** (MAD)
Apache AGE	0.1	3-hop neighborhood	GIN index (properties)	6.4	3.41
Apache AGE	0.1	3-hop neighborhood	GIN index (properties) & index (target)	6.86	3.86
Apache AGE	0.1	3-hop neighborhood	baseline	6.85	3.33
Apache AGE	0.1	3-hop neighborhood	index (dt)	4.75	1.19
Apache AGE	0.1	3-hop neighborhood	index (dt) & index (target)	5.34	1.94
Apache AGE	0.1	3-hop neighborhood	index (source)	86.54	9.8
Apache AGE	0.1	3-hop neighborhood	index (source) & index (target)	83.63	8.63
Apache AGE	0.1	3-hop neighborhood	index (source,dt)	86.86	7.26
Apache AGE	0.1	3-hop neighborhood	index (source,dt) & index (target)	83.32	7.53
Apache AGE	0.1	Temporal degree	GIN index (properties)	13.87	1.75
Apache AGE	0.1	Temporal degree	GIN index (properties) & index (target)	13.44	1.99
Apache AGE	0.1	Temporal degree	baseline	14.07	2.03
Apache AGE	0.1	Temporal degree	index (dt)	13.96	2.1
Apache AGE	0.1	Temporal degree	index (dt) & index (target)	13.92	2.18
Apache AGE	0.1	Temporal degree	index (source)	114.18	4.38
Apache AGE	0.1	Temporal degree	index (source) & index (target)	115.76	3.03
Apache AGE	0.1	Temporal degree	index (source,dt)	114.47	4.32

Apache AGE	0.1	Temporal degree	index (source,dt) & index (target)	113.82	3.12
Apache AGE	0.1	Top-10 neighbors	GIN index (properties)	14.58	1.96
Apache AGE	0.1	Top-10 neighbors	GIN index (properties) & index (target)	14.32	2.15
Apache AGE	0.1	Top-10 neighbors	baseline	12.94	2.35
Apache AGE	0.1	Top-10 neighbors	index (dt)	14.58	1.68
Apache AGE	0.1	Top-10 neighbors	index (dt) & index (target)	13.93	1.61
Apache AGE	0.1	Top-10 neighbors	index (source)	113.54	4.31
Apache AGE	0.1	Top-10 neighbors	index (source) & index (target)	114.32	4.13
Apache AGE	0.1	Top-10 neighbors	index (source,dt)	113.35	4.36
Apache AGE	0.1	Top-10 neighbors	index (source,dt) & index (target)	110.55	4.52
Apache AGE	1	3-hop neighborhood	GIN index (properties)	0.46	0.17
Apache AGE	1	3-hop neighborhood	GIN index (properties) & index (target)	0.49	0.22
Apache AGE	1	3-hop neighborhood	index (dt)	0.45	0.13
Apache AGE	1	3-hop neighborhood	index (dt) & index (target)	0.49	0.17
Apache AGE	1	3-hop neighborhood	index (source)	21.8	10.25
Apache AGE	1	3-hop neighborhood	index (source) & index (target)	5.34	5.32
Apache AGE	1	3-hop neighborhood	index (source,dt)	22.32	2.76
Apache AGE	1	3-hop neighborhood	index (source,dt) & index (target)	14.81	10.32
Apache AGE	1	Temporal degree	GIN index (properties)	1.28	0.2
Apache AGE	1	Temporal degree	GIN index (properties) & index (target)	1.39	0.17
Apache AGE	1	Temporal degree	index (dt)	1.29	0.25
Apache AGE	1	Temporal degree	index (dt) & index (target)	1.32	0.19
Apache AGE	1	Temporal degree	index (source)	32.88	3.46
Apache AGE	1	Temporal degree	index (source) & index (target)	34.88	2.16
Apache AGE	1	Temporal degree	index (source,dt)	33.17	2.9
Apache AGE	1	Temporal degree	index (source,dt) & index (target)	34.68	1.61
Apache AGE	1	Top-10 neighbors	GIN index (properties)	1.45	0.14
Apache AGE	1	Top-10 neighbors	GIN index (properties) & index (target)	1.44	0.15
Apache AGE	1	Top-10 neighbors	index (dt)	1.47	0.13
Apache AGE	1	Top-10 neighbors	index (dt) & index (target)	1.44	0.15
Apache AGE	1	Top-10 neighbors	index (source)	33.5	2.96
Apache AGE	1	Top-10 neighbors	index (source) & index (target)	33.8	2.87

Apache AGE	1	Top-10 neighbors	index (source,dt)	32.85	3.21
Apache AGE	1	Top-10 neighbors	index (source,dt) & index (target)	33.5	2.94
Apache AGE	10	3-hop neighborhood	GIN index (properties)	0.04	0.02
Apache AGE	10	3-hop neighborhood	GIN index (properties) & index (target)	0.04	0.03
Apache AGE	10	3-hop neighborhood	index (dt)	0.04	0.03
Apache AGE	10	3-hop neighborhood	index (dt) & index (target)	0.04	0.02
Apache AGE	10	3-hop neighborhood	index (source)	0.1	0.09
Apache AGE	10	3-hop neighborhood	index (source) & index (target)	0.18	0.17
Apache AGE	10	3-hop neighborhood	index (source,dt)	0.14	0.13
Apache AGE	10	3-hop neighborhood	index (source,dt) & index (target)	0.18	0.18
Apache AGE	10	Temporal degree	GIN index (properties)	0.15	0.05
Apache AGE	10	Temporal degree	GIN index (properties) & index (target)	0.15	0.05
Apache AGE	10	Temporal degree	index (dt)	0.15	0.04
Apache AGE	10	Temporal degree	index (dt) & index (target)	0.18	0.02
Apache AGE	10	Temporal degree	index (source)	5.54	0.44
Apache AGE	10	Temporal degree	index (source) & index (target)	5.6	0.27
Apache AGE	10	Temporal degree	index (source,dt)	5.61	0.62
Apache AGE	10	Temporal degree	index (source,dt) & index (target)	5.5	0.17
Apache AGE	10	Top-10 neighbors	GIN index (properties)	0.18	0.02
Apache AGE	10	Top-10 neighbors	GIN index (properties) & index (target)	0.19	0.01
Apache AGE	10	Top-10 neighbors	index (dt)	0.16	0.04
Apache AGE	10	Top-10 neighbors	index (dt) & index (target)	0.18	0.03
Apache AGE	10	Top-10 neighbors	index (source)	5.12	0.25
Apache AGE	10	Top-10 neighbors	index (source) & index (target)	4.97	0.23
Apache AGE	10	Top-10 neighbors	index (source,dt)	5.12	0.24
Apache AGE	10	Top-10 neighbors	index (source,dt) & index (target)	5.08	0.33
Native PostgreSQL	0.1	3-hop neighborhood	baseline	81.89	7.57
Native PostgreSQL	0.1	3-hop neighborhood	index (dt)	137.26	71.05
Native PostgreSQL	0.1	3-hop neighborhood	index (dt) & index (target)	174.4	27.03
Native PostgreSQL	0.1	3-hop neighborhood	index (label)	145.65	29.01
Native PostgreSQL	0.1	3-hop neighborhood	index (label) & index (target)	120.08	21.94
Native PostgreSQL	0.1	3-hop neighborhood	index (source)	210.22	3.71
Native PostgreSQL	0.1	3-hop neighborhood	index (source) & index (target)	200.04	4.62

Native PostgreSQL	0.1	3-hop neighborhood	index (source,label,dt)	202.94	3.27
Native PostgreSQL	0.1	3-hop neighborhood	index (source,label,dt) & index (target)	192.99	4.5
Native PostgreSQL	0.1	3-hop neighborhood	partial index (source,dt)	187.14	3.12
Native PostgreSQL	0.1	3-hop neighborhood	partial index (source,dt) & index (target)	176.2	2.9
Native PostgreSQL	0.1	Temporal degree	baseline	85.32	4.84
Native PostgreSQL	0.1	Temporal degree	index (dt)	181.35	43.25
Native PostgreSQL	0.1	Temporal degree	index (dt) & index (target)	168.59	52.6
Native PostgreSQL	0.1	Temporal degree	index (label)	181.16	6.34
Native PostgreSQL	0.1	Temporal degree	index (label) & index (target)	178.56	4.71
Native PostgreSQL	0.1	Temporal degree	index (source)	230.92	3.38
Native PostgreSQL	0.1	Temporal degree	index (source) & index (target)	227.45	3.46
Native PostgreSQL	0.1	Temporal degree	index (source,label,dt)	224.82	3.46
Native PostgreSQL	0.1	Temporal degree	index (source,label,dt) & index (target)	221.61	3.39
Native PostgreSQL	0.1	Temporal degree	partial index (source,dt)	199.52	2.83
Native PostgreSQL	0.1	Temporal degree	partial index (source,dt) & index (target)	194.23	2.66
Native PostgreSQL	0.1	Top-10 neighbors	baseline	84.08	4.01
Native PostgreSQL	0.1	Top-10 neighbors	index (dt)	180.6	28.84
Native PostgreSQL	0.1	Top-10 neighbors	index (dt) & index (target)	148.8	52.63
Native PostgreSQL	0.1	Top-10 neighbors	index (label)	168.93	6.58
Native PostgreSQL	0.1	Top-10 neighbors	index (label) & index (target)	166.06	6.43
Native PostgreSQL	0.1	Top-10 neighbors	index (source)	212.13	2.38
Native PostgreSQL	0.1	Top-10 neighbors	index (source) & index (target)	208.07	2.43
Native PostgreSQL	0.1	Top-10 neighbors	index (source,label,dt)	204.73	3.1
Native PostgreSQL	0.1	Top-10 neighbors	index (source,label,dt) & index (target)	204.65	2.18
Native PostgreSQL	0.1	Top-10 neighbors	partial index (source,dt)	185.01	2.45
Native PostgreSQL	0.1	Top-10 neighbors	partial index (source,dt) & index (target)	181.8	2.53
Native PostgreSQL	1	3-hop neighborhood	index (dt)	40.03	25.34
Native PostgreSQL	1	3-hop neighborhood	index (dt) & index (target)	46.31	34.31
Native PostgreSQL	1	3-hop neighborhood	index (label)	37.5	17.1
Native PostgreSQL	1	3-hop neighborhood	index (label) & index (target)	38.69	18.24
Native PostgreSQL	1	3-hop neighborhood	index (source)	146.88	61.36

Native PostgreSQL	1	3-hop neighborhood	index (source) & index (target)	131.25	66.65
Native PostgreSQL	1	3-hop neighborhood	index (source,label,dt)	190.77	15.84
Native PostgreSQL	1	3-hop neighborhood	index (source,label,dt) & index (target)	187.58	7.6
Native PostgreSQL	1	3-hop neighborhood	partial index (source,dt)	177.24	12.15
Native PostgreSQL	1	3-hop neighborhood	partial index (source,dt) & index (target)	169.38	9.4
Native PostgreSQL	1	Temporal degree	index (dt)	51.55	13.23
Native PostgreSQL	1	Temporal degree	index (dt) & index (target)	53.91	15.07
Native PostgreSQL	1	Temporal degree	index (label)	63.04	4.74
Native PostgreSQL	1	Temporal degree	index (label) & index (target)	63.63	5.01
Native PostgreSQL	1	Temporal degree	index (source)	223.31	4.97
Native PostgreSQL	1	Temporal degree	index (source) & index (target)	219.97	4.28
Native PostgreSQL	1	Temporal degree	index (source,label,dt)	221.21	3.06
Native PostgreSQL	1	Temporal degree	index (source,label,dt) & index (target)	217.49	2.92
Native PostgreSQL	1	Temporal degree	partial index (source,dt)	194.04	3.58
Native PostgreSQL	1	Temporal degree	partial index (source,dt) & index (target)	192.49	2.5
Native PostgreSQL	1	Top-10 neighbors	index (dt)	53.12	11.79
Native PostgreSQL	1	Top-10 neighbors	index (dt) & index (target)	54.23	13.4
Native PostgreSQL	1	Top-10 neighbors	index (label)	60.98	4.94
Native PostgreSQL	1	Top-10 neighbors	index (label) & index (target)	62.32	3.71
Native PostgreSQL	1	Top-10 neighbors	index (source)	207.9	3.83
Native PostgreSQL	1	Top-10 neighbors	index (source) & index (target)	205.36	3.08
Native PostgreSQL	1	Top-10 neighbors	index (source,label,dt)	203.17	2.74
Native PostgreSQL	1	Top-10 neighbors	index (source,label,dt) & index (target)	199.76	2.65
Native PostgreSQL	1	Top-10 neighbors	partial index (source,dt)	180.47	2.26
Native PostgreSQL	1	Top-10 neighbors	partial index (source,dt) & index (target)	178.25	2.57
Native PostgreSQL	10	3-hop neighborhood	index (dt)	9.5	9.01
Native PostgreSQL	10	3-hop neighborhood	index (dt) & index (target)	4.36	4.3
Native PostgreSQL	10	3-hop neighborhood	index (label)	5.6	4.37
Native PostgreSQL	10	3-hop neighborhood	index (label) & index (target)	6.29	4.78
Native PostgreSQL	10	3-hop neighborhood	index (source)	32.66	32.65
Native PostgreSQL	10	3-hop neighborhood	index (source) & index (target)	16.21	16.2

Native PostgreSQL	10	3-hop neighborhood	index (source,label,dt)	34.1	34.03
Native PostgreSQL	10	3-hop neighborhood	index (source,label,dt) & index (target)	71.53	71.48
Native PostgreSQL	10	3-hop neighborhood	partial index (source,dt)	20.89	20.89
Native PostgreSQL	10	3-hop neighborhood	partial index (source,dt) & index (target)	62.09	62.08
Native PostgreSQL	10	Temporal degree	index (dt)	14.83	8.99
Native PostgreSQL	10	Temporal degree	index (dt) & index (target)	19.96	12.3
Native PostgreSQL	10	Temporal degree	index (label)	26.9	3.08
Native PostgreSQL	10	Temporal degree	index (label) & index (target)	24.17	5.54
Native PostgreSQL	10	Temporal degree	index (source)	200.72	6.6
Native PostgreSQL	10	Temporal degree	index (source) & index (target)	190.75	6.28
Native PostgreSQL	10	Temporal degree	index (source,label,dt)	199.72	2.7
Native PostgreSQL	10	Temporal degree	index (source,label,dt) & index (target)	200.04	8.71
Native PostgreSQL	10	Temporal degree	partial index (source,dt)	191.74	3.03
Native PostgreSQL	10	Temporal degree	partial index (source,dt) & index (target)	189.65	3.16
Native PostgreSQL	10	Top-10 neighbors	index (dt)	11.24	5.99
Native PostgreSQL	10	Top-10 neighbors	index (dt) & index (target)	12.6	6.82
Native PostgreSQL	10	Top-10 neighbors	index (label)	23.19	3.88
Native PostgreSQL	10	Top-10 neighbors	index (label) & index (target)	28.27	1.96
Native PostgreSQL	10	Top-10 neighbors	index (source)	191.5	8.22
Native PostgreSQL	10	Top-10 neighbors	index (source) & index (target)	183.28	6.68
Native PostgreSQL	10	Top-10 neighbors	index (source,label,dt)	182.78	2.37
Native PostgreSQL	10	Top-10 neighbors	index (source,label,dt) & index (target)	183.22	3.43
Native PostgreSQL	10	Top-10 neighbors	partial index (source,dt)	173.84	2.76
Native PostgreSQL	10	Top-10 neighbors	partial index (source,dt) & index (target)	174.23	2.72
Neo4j	0.1	3-hop neighborhood	baseline	333.33	83.33
Neo4j	0.1	3-hop neighborhood	index (node:id)	333.33	133.33
Neo4j	0.1	3-hop neighborhood	index (node:id),(rel:dt)	333.33	108.33
Neo4j	0.1	3-hop neighborhood	index (rel:dt)	333.33	83.33
Neo4j	0.1	Temporal degree	baseline	333.33	0.0
Neo4j	0.1	Temporal degree	index (node:id)	500.0	0.0
Neo4j	0.1	Temporal degree	index (node:id),(rel:dt)	333.33	0.0

Neo4j	0.1	Temporal degree	index (rel:dt)	333.33	0.0
Neo4j	0.1	Top-10 neighbors	baseline	333.33	0.0
Neo4j	0.1	Top-10 neighbors	index (node:id)	333.33	125.0
Neo4j	0.1	Top-10 neighbors	index (node:id),(rel:dt)	333.33	0.0
Neo4j	0.1	Top-10 neighbors	index (rel:dt)	333.33	0.0
Neo4j	1	3-hop neighborhood	baseline	66.96	33.04
Neo4j	1	3-hop neighborhood	index (node:id)	100.0	11.11
Neo4j	1	3-hop neighborhood	index (node:id),(rel:dt)	71.43	39.68
Neo4j	1	3-hop neighborhood	index (rel:dt)	100.0	11.11
Neo4j	1	Temporal degree	baseline	100.0	0.0
Neo4j	1	Temporal degree	index (node:id)	111.11	0.0
Neo4j	1	Temporal degree	index (node:id),(rel:dt)	111.11	0.0
Neo4j	1	Temporal degree	index (rel:dt)	111.11	0.0
Neo4j	1	Top-10 neighbors	baseline	100.0	0.0
Neo4j	1	Top-10 neighbors	index (node:id)	111.11	0.0
Neo4j	1	Top-10 neighbors	index (node:id),(rel:dt)	111.11	5.56
Neo4j	1	Top-10 neighbors	index (rel:dt)	100.0	9.09
Neo4j	10	3-hop neighborhood	baseline	16.26	2.97
Neo4j	10	3-hop neighborhood	index (node:id)	16.81	3.29
Neo4j	10	3-hop neighborhood	index (node:id),(rel:dt)	16.81	3.47
Neo4j	10	3-hop neighborhood	index (rel:dt)	15.28	3.96
Neo4j	10	Temporal degree	baseline	18.87	0.35
Neo4j	10	Temporal degree	index (node:id)	18.87	0.36
Neo4j	10	Temporal degree	index (node:id),(rel:dt)	18.87	0.36
Neo4j	10	Temporal degree	index (rel:dt)	18.87	0.36
Neo4j	10	Top-10 neighbors	baseline	18.52	0.35
Neo4j	10	Top-10 neighbors	index (node:id)	18.87	0.36
Neo4j	10	Top-10 neighbors	index (node:id),(rel:dt)	18.87	0.36
Neo4j	10	Top-10 neighbors	index (rel:dt)	18.87	0.36
TimescaleDB	0.1	3-hop neighborhood	baseline	45.87	15.16
TimescaleDB	0.1	3-hop neighborhood	index (label,dt)	75.82	1.98
TimescaleDB	0.1	3-hop neighborhood	index (label,dt) & index (target)	71.56	2.53
TimescaleDB	0.1	3-hop neighborhood	index (source,dt)	75.05	1.19
TimescaleDB	0.1	3-hop neighborhood	index (source,dt) & index (target)	74.04	0.95
TimescaleDB	0.1	3-hop neighborhood	index (source,label,dt)	75.71	1.38

TimescaleDB	0.1	3-hop neighborhood	index (source,label,dt) & index (target)	73.9	0.96
TimescaleDB	0.1	Temporal degree	baseline	63.14	8.19
TimescaleDB	0.1	Temporal degree	index (label,dt)	80.85	1.85
TimescaleDB	0.1	Temporal degree	index (label,dt) & index (target)	78.69	2.0
TimescaleDB	0.1	Temporal degree	index (source,dt)	82.03	0.79
TimescaleDB	0.1	Temporal degree	index (source,dt) & index (target)	81.18	0.8
TimescaleDB	0.1	Temporal degree	index (source,label,dt)	81.68	0.86
TimescaleDB	0.1	Temporal degree	index (source,label,dt) & index (target)	80.31	0.92
TimescaleDB	0.1	Top-10 neighbors	baseline	63.24	6.69
TimescaleDB	0.1	Top-10 neighbors	index (label,dt)	79.81	2.13
TimescaleDB	0.1	Top-10 neighbors	index (label,dt) & index (target)	79.37	1.9
TimescaleDB	0.1	Top-10 neighbors	index (source,dt)	80.4	1.82
TimescaleDB	0.1	Top-10 neighbors	index (source,dt) & index (target)	79.93	1.14
TimescaleDB	0.1	Top-10 neighbors	index (source,label,dt)	82.17	0.74
TimescaleDB	0.1	Top-10 neighbors	index (source,label,dt) & index (target)	80.99	0.73
TimescaleDB	1	3-hop neighborhood	index (label,dt)	51.16	20.02
TimescaleDB	1	3-hop neighborhood	index (label,dt) & index (target)	50.01	17.26
TimescaleDB	1	3-hop neighborhood	index (source,dt)	74.41	3.47
TimescaleDB	1	3-hop neighborhood	index (source,dt) & index (target)	71.63	3.04
TimescaleDB	1	3-hop neighborhood	index (source,label,dt)	72.07	3.81
TimescaleDB	1	3-hop neighborhood	index (source,label,dt) & index (target)	72.04	1.91
TimescaleDB	1	Temporal degree	index (label,dt)	66.1	9.21
TimescaleDB	1	Temporal degree	index (label,dt) & index (target)	64.36	9.22
TimescaleDB	1	Temporal degree	index (source,dt)	81.26	0.74
TimescaleDB	1	Temporal degree	index (source,dt) & index (target)	79.21	0.78
TimescaleDB	1	Temporal degree	index (source,label,dt)	80.25	0.88
TimescaleDB	1	Temporal degree	index (source,label,dt) & index (target)	78.87	0.63
TimescaleDB	1	Top-10 neighbors	index (label,dt)	63.76	9.7
TimescaleDB	1	Top-10 neighbors	index (label,dt) & index (target)	64.2	9.44
TimescaleDB	1	Top-10 neighbors	index (source,dt)	82.05	0.73
TimescaleDB	1	Top-10 neighbors	index (source,dt) & index (target)	80.69	0.6
TimescaleDB	1	Top-10 neighbors	index (source,label,dt)	80.85	0.6

TimescaleDB	1	Top-10 neighbors	index (source,label,dt) & index (target)	79.96	0.65
TimescaleDB	10	3-hop neighborhood	index (label,dt)	17.99	16.41
TimescaleDB	10	3-hop neighborhood	index (label,dt) & index (target)	14.44	14.3
TimescaleDB	10	3-hop neighborhood	index (source,dt)	37.14	31.54
TimescaleDB	10	3-hop neighborhood	index (source,dt) & index (target)	16.5	16.49
TimescaleDB	10	3-hop neighborhood	index (source,label,dt)	48.93	25.76
TimescaleDB	10	3-hop neighborhood	index (source,label,dt) & index (target)	26.76	26.75
TimescaleDB	10	Temporal degree	index (label,dt)	34.67	9.71
TimescaleDB	10	Temporal degree	index (label,dt) & index (target)	33.19	8.21
TimescaleDB	10	Temporal degree	index (source,dt)	68.52	2.25
TimescaleDB	10	Temporal degree	index (source,dt) & index (target)	76.03	2.93
TimescaleDB	10	Temporal degree	index (source,label,dt)	77.81	2.61
TimescaleDB	10	Temporal degree	index (source,label,dt) & index (target)	74.44	3.3
TimescaleDB	10	Top-10 neighbors	index (label,dt)	32.35	9.2
TimescaleDB	10	Top-10 neighbors	index (label,dt) & index (target)	32.9	7.82
TimescaleDB	10	Top-10 neighbors	index (source,dt)	69.41	2.74
TimescaleDB	10	Top-10 neighbors	index (source,dt) & index (target)	65.46	3.04
TimescaleDB	10	Top-10 neighbors	index (source,label,dt)	76.72	3.24
TimescaleDB	10	Top-10 neighbors	index (source,label,dt) & index (target)	76.03	2.96

Acronyms

BST	Binary Search Tree
CODY	COnversation DYnamics
DBMS	Database Management System
ER	Erdős and Rényi
GDBMS	Graph Database Management System
GIN	Generalized Inverted Index
HIN	Heterogeneous Information Network
JVM	Java Virtual Machine
LPG	Labeled Property Graph
MAD	Median Absolute Deviation
NLP	Natural Language Processing
RDBMS	Relational Database Management System
THIN	Temporal Heterogeneous Information Network
TPS	Transactions Per Second
TSDB	Time Series Database

Glossary

$\mathbb{A}$	Adjacency matrix
$\tilde{G}$	Aggregated multi-slice network
π	Attribute mapping
G^P	Augmented reduced graph
$\langle k \rangle$	Average degree
C, c	Centrality
$\mathbb{C}$	Community
λ	Completion rate
k	Degree
d	Document
D	Documents
l	Edge
$\mathbb{L}$	Edge list
w	Edge weight
W	Edge weights
L	Edges
E	Entities
e	Entity
ϵ	Entity extraction
G^{ER}	Erdős and Rényi graph
f	Frequency
G	Graph

h	Hashtag
u_h	Hashtag usage
k^{in}	In-degree
$\mathbb{k}^{in}$	In-strength
m	Index of layer in multilayer network
$\ddot{\mathscr{G}}$	Interlink networks
$\dot{\mathscr{G}}$	Intralink networks
$\mathscr{M}$	Layers in multilayer network
r	Link type
ψ	Link type mapping
R	Link types
P	Meta path
r_P	Meta path composite relation
p	Meta path instance
$\mathbb{P}$	Meta path instances
$\mathcal{Q}$	Modularity
M	Multilayer network
S_G	Network schema
o	Node occurrence
n_L	Number of edges
$n_{\mathscr{M}}$	Number of layers in multilayer network
n_V	Number of nodes
a	Object type
φ	Object type mapping
A	Object types
k^{out}	Out-degree
$\mathbb{k}^{out}$	Out-strength
α	Overlap coefficient
$\ddot{P}$	Partition

P, p	Probability
χ^2_ν	Reduced chi-square
β	Resolution parameter
Γ	Saturation time constant
$\bar{S}$	Shortest path
δ	Shortest path length
$\Bbbk$	Strength
G^s	Supernetwork
$\mathbb{T}$	Time domain
Δt	Time interval
T	Time period
w	Time window
t	Timestamp
ξ	Trend detection
τ	Trend score
v	Vertex
V	Vertices
$\mathscr{W}$	Walk
$n_{\mathscr{W}}$	Walk length
$\mathbb{A}^W$	Weighted adjacency matrix
G^W	Weighted graph
$\mathbb{w}$	Wiener index

Bibliography

Russell L. Ackoff. From Data to Wisdom. *Journal of Applied Systems Analysis*, 16:3–9, 1989.

Ryan Prescott Adams and David J. C. MacKay. Bayesian Online Changepoint Detection. *arXiv preprint arXiv:0710.3742*, 2007.

Cham Aggarwal, Yan Xie, and Philip S. Yu. On Dynamic Link Inference in Heterogeneous Networks. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 415–426, 2012.

Ibrar Ahmed. PostgreSQL Vacuuming Command to Optimize Database Performance. https://www.percona.com/blog/postgresql-vacuuming-to-optimize-database-performance-and-reclaim-space, 2023. Accessed: 2023-10-10.

Yong-Yeol Ahn, Sebastian E. Ahnert, James P. Bagrow, and Albert-László Barabási. Flavor network and the principles of food pairing. *Scientific Reports*, pages 1–7, 2011.

Tushar Ahuja. How to tune PostgreSQL for memory | EDB. https://www.enterprisedb.com/postgres-tutorials/how-tune-postgresql-memory, 2022. Accessed: 2023-10-10.

Alberto Aleta and Yamir Moreno. Multilayer Networks in a Nutshell. *Annual Review of Condensed Matter Physics*, 10:45–62, 2019.

Jeff Alstott, Ed Bullmore, and Dietmar Plenz. powerlaw: A Python Package for Analysis of Heavy-Tailed Distributions. *PLOS ONE*, 9:1–11, 2014.

Leandro Anghinoni, Liang Zhao, Donghong Ji, and Heng Pan. Time series trend detection and forecasting using complex network topology analysis. *Neural Networks*, 117:295–306, 2019.

Renzo Angles. The Property Graph Database Model. In *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management*, 2018.

Dmitry Anikin, Oleg Borisenko, and Yaroslav Nedumov. Labeled Property Graphs: SQL or NoSQL? In *2019 Ivannikov Memorial Workshop (IVMEM)*, pages 7–13, 2019.

Issa Annamoradnejad and Jafar Habibi. A Comprehensive Analysis of Twitter Trending Topics. In *2019 5th International Conference on Web Research (ICWR)*, pages 22–27, 2019.

Wenceslao Arroyo-Machado, Daniel Torres-Salinas, and Nicolas Robinson-Garcia. Identifying and characterizing social media communities: a socio-semantic network approach to altmetrics. *Scientometrics*, 126(11):9267–9289, 2021.

Sitaram Asur, Bernardo A. Huberman, Gabor Szabo, and Chunyan Wang. Trends in Social Media: Persistence and Decay. *Proceedings of the International AAAI Conference on Web and Social Media*, 5(1):434–437, 2011.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, pages 722–735. Springer, 2007.

Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

Bogdan Batrinca and Philip C. Treleaven. Social media analytics: a survey of techniques, tools and platforms. *AI & Society*, 30:89–116, 2015.

Rudolf Bayer and Edward McCreight. Organization and Maintenance of Large Ordered Indices. In *Proceedings of the 1970 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*, pages 107–141, 1970.

Ferenc Béres, Róbert Pálovics, Anna Oláh, and András A. Benczúr. Temporal walk based centrality metric for graph streams. *Applied Network Science*, 3:1–26, 2018.

Sandjai Bhulai, Peter Kampstra, Lidewij Kooiman, Ger Koole, Marijn Deurloo, and Bert Kok. Trend Visualization on Twitter: What's Hot and What's Not? In *1st International Conference on Data Analytics*, pages 43–48, 2012.

Ranran Bian, Yun Sing Koh, Gillian Dobbie, and Anna Divoli. Network Embedding and Change Modeling in Dynamic Heterogeneous Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 861–864, 2019.

Ginestra Bianconi. *Multilayer Networks: Structure and Function*. Oxford University Press, 2018.

Roi Blanco and Christina Lioma. Graph-based term weighting for information retrieval. *Information Retrieval*, 15:54–92, 2012.

Javier Borge-Holthoefer, Raquel A. Baños, Carlos Gracia-Lázaro, and Yamir Moreno. Emergence of consensus as a modular-to-nested transition in communication dynamics. *Scientific Reports*, 7:1–9, 2017.

Nicholas Botzer and Tim Weninger. Entity graphs for exploring online discourse. *Knowledge and Information Systems*, page 3591–3609, 2023.

Andrew Bowman. Explanation of the "consumed after" message in query results – Knowledge Base. https://neo4j.com/developer/kb/explanation-of-consumed-after-message-in-query-results, 2023. Accessed: 2023-09-13.

Marco Brambilla, Alireza Javadian Sabet, Kalyani Kharmale, and Amin Endah Sulistiawati. Graph-Based Conversation Analysis in Social Media. *Big Data and Cognitive Computing*, 6(4):113, 2022.

Nathan Bronson, Zach Amsden, George Cabrera, Prasad Chakka, Peter Dimov, Hui Ding, Jack Ferris, Anthony Giardullo, Sachin Kulkarni, Harry Li, Mark Marchukov, Dmitri Petrov, Lovro Puzar, Yee Jiun Song, and Venkat Venkataramani. TAO: Facebook's Distributed Data Store for the Social Graph. In *2013 USENIX Annual Technical Conference (USENIX ATC 13)*, pages 49–60, 2013.

Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. Structural Trend Analysis for Online Social Networks. *Proceedings of the VLDB Endowment*, 4(10):646–656, 2011.

Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

Rémy Cazabet, Hideaki Takeda, Masahiro Hamasaki, and Frédéric Amblard. Using dynamic community detection to identify trends in user-generated content. *Social Network Analysis and Mining*, 2:361–371, 2012.

Stefano Ceri, Mauro Negri, and Giuseppe Pelagatti. Horizontal Data Partitioning in Database Design. In *Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data*, pages 128–136, 1982.

Ismail Chabini and Shan Lan. Adaptations of the A* Algorithm for the Computation of Fastest Paths in Deterministic Discrete-Time Dynamic Networks. *IEEE Transactions on Intelligent Transportation Systems*, 3:60–74, 2002.

Andrew Chadwick. The Political Information Cycle in a Hybrid News System: The British Prime Minister and the "Bullygate" Affair. *The International Journal of Press/Politics*, 16:3–29, 2011.

Bongsug Chae and Eunhye Park. Corporate Social Responsibility (CSR): A Survey of Topics and Trends Using Twitter Data and Topic Modeling. *Sustainability*, 10(7):2231, 2018.

Nuttapong Chairatanakul, Xin Liu, and Tsuyoshi Murata. PGRA: Projected graph relation-feature attention network for heterogeneous information network embedding. *Information Sciences*, 570:769–794, 2021.

Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary Clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 554–560, 2006.

Jonathan Chang and David Blei. Relational Topic Models for Document Networks. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 81–88. PMLR, 2009.

Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. PME: Projected Metric Embedding on Heterogeneous Networks for Link Prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1177–1186, 2018.

Peter Cogan, Matthew Andrews, Milan Bradonjic, W. Sean Kennedy, Alessandra Sala, and Gabriel Tucci. Reconstruction and Analysis of Twitter Conversation Graphs. In *Proceedings of the 1st ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research*, pages 25–31, 2012.

Jean-Philippe Cointet and Camille Roth. Socio-semantic Dynamics in a Blog Network. In *2009 International Conference on Computational Science and Engineering*, volume 4, pages 114–121, 2009.

Douglas Comer. The Ubiquitous B-Tree. *ACM Computing Surveys*, 11(2):121–137, 1979.

Nick Crossley, Elisa Bellotti, Gemma Edwards, Martin G. Everett, Johan Koskinen, and Mark Tranmer. *Social Network Analysis for Ego-Nets*. SAGE Publications, 2015.

Jonathan D. Cryer and Kung-Sik Chan. *Time Series Analysis: With Applications in R*, volume 2. Springer, 2008.

Gábor Csárdi and Tamás Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, pages 1–9, 2006.

Hao Cui and János Kertész. "Born in Rome" or "Sleeping Beauty": Emergence of hashtag popularity on the Chinese microblog Sina Weibo. *Physica A: Statistical Mechanics and its Applications*, 619, 2023.

Alfredo Cuzzocrea and Francesco Folino. Community Evolution Detection in Time-Evolving Information Networks. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 93–96, 2013.

Gregorio D'Agostino and Antonio Scala. *Networks of Networks: The Last Frontier of Complexity*. Springer, 2014.

Mrinmay Dey and Michelle Conlin. Elon Musk says Twitter's blue bird to be replaced by an X | Reuters. https://www.reuters.com/technology/elon-musk-says-twitter-change-logo-adieu-all-birds-2023-07-23, 2023. Accessed: 2023-12-14.

Nicholas Diakopoulos, Mor Naaman, Tayebeh Yazdani, and Funda Kivran-Swaine. *Social Media Visual Analytics for Events*, pages 189–209. Springer, 2011.

Pengjie Ding, Yijian Cheng, Wei Lu, Hao Huang, and Xiaoyong Du. Which Category Is Better: Benchmarking the RDBMSs and GDBMSs. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 207–215, 2019.

David Dominguez-Sal, Norbert Martinez-Bazan, Victor Muntes-Mulero, Pere Baleta, and Josep Lluis Larriba-Pey. A Discussion on the Design of Graph Database Benchmarks. In *Technology Conference on Performance Evaluation and Benchmarking*, pages 25–40, 2010.

Zeel Doshi, Subhash Nadkarni, Kushal Ajmera, and Neepa Shah. TweetAnalyzer: Twitter Trend Detection and Visualization. In *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, pages 1–6. IEEE, 2017.

Kyt Dotson. Twitter is closing free access to its API starting Feb. 9 – SiliconANGLE. https://siliconangle.com/2023/02/02/twitter-closing-free-access-api-starting-february-9, 2023. Accessed: 2023-07-12.

Wenwen Dou, Xiaoyu Wang, Drew Skau, William Ribarsky, and Michelle X. Zhou. LeadLine: Interactive Visual Analysis of Text Data through Event Identification and Exploration. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 93–102, 2012.

Daniel Edler, Anton Holmgren, and Martin Rosvall. The MapEquation software package. https://mapequation.org, 2023.

Lisa Ehrlinger and Wolfram Wöß. Towards a Definition of Knowledge Graphs. *SEMANTICS 2016: Posters and Demos Track*, pages 1–4, 2016.

Benjamin Erb, Dominik Meißner, Jakob Pietron, and Frank Kargl. Chronograph: A Distributed Processing Platform for Online and Batch Computations on Event-sourced Graphs. In *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, pages 78–87, 2017.

P. Erdős and A. Rényi. On random graphs I. *Publicationes Mathematicae Debrecen*, pages 290–297, 1959.

P. Erdős and A. Rényi. ON THE EVOLUTION OF RANDOM GRAPHS. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–60, 1960.

Europe Media Monitor Team. EMM Overview. https://emm.newsbrief.eu/overview.html, 2023. Accessed: 2023-12-12.

Matheus Farias. postgresql – How and where data is actually stored in Apache AGE? – Stack Overflow. https://stackoverflow.com/a/75828713, 2023. Accessed: 2023-07-16.

Christina Feilmayr and Wolfram Wöß. An analysis of ontologies and their success factors for application to business. *Data & Knowledge Engineering*, 101:1–23, 2016.

Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

Richard Gartner. *Metadata: Shaping Knowledge from Antiquity to the Semantic Web*. Springer, 2016.

Noé Gaumont, Maziyar Panahi, and David Chavalarias. Reconstruction of the socio-semantic dynamics of political activist Twitter networks — Method and application to the 2017 French presidential election. *PLOS ONE*, 13(9):1–38, 2018.

Bill Gerhardt, Kate Griffin, and Roland Klemann. Unlocking Value in the Fragmented World of Big Data Analytics. Technical report, Cisco Internet Business Solutions Group, 2012.

Peter A. Gloor, Jonas Krauss, Stefan Nann, Kai Fischbach, and Detlef Schoder. Web Science 2.0: Identifying Trends through Semantic Social Network Analysis. In *2009 International Conference on Computational Science and Engineering*, volume 4, pages 215–222, 2009.

Dave Gordon. Warm the cache to improve performance from cold start – Knowledge Base. https://neo4j.com/developer/kb/warm-the-cache-to-improve-performance-from-cold-start, 2023. Accessed: 2023-07-16.

Miha Grčar, Nejc Trdin, and Nada Lavrač. A Methodology for Mining Document-Enriched Heterogeneous Information Networks. *The Computer Journal*, 56(3):321–335, 2013.

Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A. Zighed. Information Diffusion in Online Social Networks: A Survey. *ACM Sigmod Record*, 42(2):17–28, 2013.

Steve Hanneke, Wenjie Fu, and Eric P. Xing. Discrete temporal models of social networks. *Electronic Journal of Statistics*, 4:585–605, 2010.

Yuanzhe Hao, Xiongpai Qin, Yueguo Chen, Yaru Li, Xiaoguang Sun, Yu Tao, Xiao Zhang, and Xiaoyong Du. TS-Benchmark: A Benchmark for Time Series Databases. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 588–599, 2021.

Zellig S. Harris. Distributional Structure. *WORD*, 10:146–162, 1954.

Anat Hashavit, Roy Levin, Ido Guy, and Gilad Kutiel. Effective Trend Detection within a Dynamic Search Context. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 817–820, 2016.

Iina Hellsten and Loet Leydesdorff. Automated Analysis of Actor–Topic Networks on Twitter: New Approaches to the Analysis of Socio-Semantic Networks. *Journal of the Association for Information Science and Technology*, 71:3–15, 2020.

Iina Hellsten, Tobias Opthof, and Loet Leydesdorff. N-mode network approach for socio-semantic analysis of scientific publications. *Poetics*, 78, 2020.

Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

Mohammad Mehdi Hosseinzadeh, Mario Cannataro, Pietro Hiram Guzzi, and Riccardo Dondi. Temporal networks in biology and medicine: a survey on models, algorithms, and tools. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 12, 2023.

Shenyang Huang, Yasmeen Hitti, Guillaume Rabusseau, and Reihaneh Rabbany. Laplacian Change Point Detection for Dynamic Graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 349–358, 2020.

Mark Huberty. Can we vote with our tweet? On the perennial difficulty of election forecasting with social media. *International Journal of Forecasting*, 31(3):992–1007, 2015.

Wenyu Huo and Vassilis J. Tsotras. Efficient Temporal Shortest Path Queries on Evolving Social Graphs. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, pages 1–4, 2014.

Anand Padmanabha Iyer, Qifan Pu, Kishan Patel, Joseph E. Gonzalez, and Ion Stoica. TEGRA: Efficient Ad-Hoc Analytics on Evolving Graphs. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 337–355, 2021.

Muhammad Aqib Javed, Muhammad Shahzad Younis, Siddique Latif, Junaid Qadir, and Adeel Baig. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, 108:87–111, 2018.

Yantao Jia, Yuanzhuo Wang, Xiaolong Jin, Zeya Zhao, and Xueqi Cheng. Link Inference in Dynamic Heterogeneous Information Network: A Knapsack-Based Approach. *IEEE Transactions on Computational Social Systems*, 4(3):80–92, 2017.

B. Jones-Kavalier and S. L. Flannigan. Connecting the Digital Dots: Literacy of the 21st Century. https://er.educause.edu/articles/2006/1/connecting-the-digital-dots-literacy-of-the-21st-century, 2006. Accessed: 2023-07-18.

Gillian Kant, Christoph Weisser, and Benjamin Säfken. TTLocVis: A Twitter Topic Location Visualization Package. *Journal of Open Source Software*, 5(54), 2020.

Hikmat Ullah Khan, Shumaila Nasir, Kishwar Nasim, Danial Shabbir, and Ahsan Mahmood. Twitter trends: A ranking algorithm analysis on real time data. *Expert Systems with Applications*, 164, 2021.

Jongkwang Kim and Thomas Wilhelm. What is a complex graph? *Physica A: Statistical Mechanics and its Applications*, 387(11):2637–2652, 2008.

Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, 2014.

Tim König, Wolf J. Schünemann, Alexander Brand, Julian Freyberg, and Michael Gertz. The EPINetz Twitter Politicians Dataset 2021. A New Resource for the Study of the German Twittersphere and Its Application for the 2021 Federal Elections. *Politische Vierteljahresschrift*, 63: 529–547, 2022.

Vassilis Kostakos. Temporal graphs. *Physica A: Statistical Mechanics and its Applications*, 388(6): 1007–1023, 2009.

Caglar Koylu. Modeling and visualizing semantic and spatio-temporal evolution of topics in interpersonal communication on Twitter. *International Journal of Geographical Information Science*, 33(4):805–832, 2019.

Kultusministerkonferenz. Bildung in der digitalen Welt: Strategie der Kultusministerkonferenz. https://www.kmk.org/fileadmin/pdf/PresseUndAktuelles/2018/Digitalstrategie_2017_mit_Weiterbildung.pdf, 2017. Accessed: 2023-07-18.

Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8(61), 2018.

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

Gotthold Ephraim Lessing. *Hamburgische Dramaturgie: Zweyter Band*. In Commission bey J. H. Cramer, in Bremen, 1769.

Yuchen Li, Zhengzhi Lou, Yu Shi, and Jiawei Han. Temporal Motifs in Heterogeneous Information Networks. In *MLG Workshop@KDD*, 2018.

Wouter Ligtenberg and Yulong Pei. Introduction to a Temporal Graph Benchmark. *arXiv preprint arXiv:1703.02852*, 2017.

Austin P. Logan, Phillip M. LaCasse, and Brian J. Lunday. Social network analysis of Twitter interactions: a directed multilayer network approach. *Social Network Analysis and Mining*, 13, 2023.

Philipp Lorenz, Frederik Wolf, Jonas Braun, Nataša Djurdjevac Conrad, and Philipp Hövel. Capturing the Dynamics of Hashtag-Communities. In *Complex Networks & Their Applications VI*, pages 401–413, 2017.

Philipp Lorenz-Spreen, Frederik Wolf, Jonas Braun, Gourab Ghoshal, Nataša Djurdjevac Conrad, and Philipp Hövel. Tracking online topics over time: understanding dynamic hashtag communities. *Computational Social Networks*, 5:1–18, 2018.

Martin Macak, Matus Stovcik, and Barbora Buhnova. The Suitability of Graph Databases for Big Data Analysis: A Benchmark. In *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security (IoTBDS 2020)*, pages 213–220, 2020.

Josh Mackay. PostgreSQL®, Docker, and Shared Memory – Instaclustr. https://www.instaclustr.com/blog/postgresql-docker-and-shared-memory, 2023. Accessed: 2023-12-21.

Axel Maireder and Julian Ausserhofer. Political Discourses on Twitter: Networking Topics, Objects, and People. *Twitter and Society*, 89:305–318, 2014.

Zahra Majdabadi, Behnam Sabeti, Preni Golazizian, Seyed Arad Ashrafi Asli, Omid Momenzadeh, and Reza Fahmi. Twitter Trend Extraction: A Graph-based Approach for Tweet and Hashtag Ranking, Utilizing No-Hashtag Tweets. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6213–6219, 2020.

David Marangoni-Simonsen and Yao Xie. Sequential Changepoint Approach for Online Community Detection. *IEEE Signal Processing Letters*, 22(8):1035–1039, 2015.

Mugilan Mariappan and Keval Vora. GraphBolt: Dependency-Driven Synchronous Processing of Streaming Graphs. In *Proceedings of the 14th EuroSys Conference 2019*, pages 1–16, 2019.

Binny Mathew, Unnikrishnan A, Tanmoy Chakraborty, Niloy Ganguly, and Samik Datta. Mining Twitter Conversations around E-Commerce Promotional Events. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, pages 345–348, 2016.

David Melamed. Community Structures in Bipartite Networks: A Dual-Projection Approach. *PLOS ONE*, 9(5):1–5, 2014.

Amin Milani Fard, Ebrahim Bagheri, and Ke Wang. Relationship Prediction in Dynamic Heterogeneous Information Networks. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019*, pages 19–34, 2019.

Mario Miler, Damir Medak, and Dražen Odobašić. The Shortest Path Algorithm Performance Comparison in Graph and Relational Database on a Transportation Network. *Promet – Traffic&Transportation*, 26:75–82, 2014.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, 2002.

Vera Zaychik Moffitt and Julia Stoyanovich. Temporal Graph Algebra. In *Proceedings of the 16th International Symposium on Database Programming Languages*, pages 1–12, 2017.

Derek G. Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martín Abadi. Naiad: A Timely Dataflow System. In *Proceedings of the 24th ACM Symposium on Operating Systems Principles*, pages 439–455, 2013.

Seth A. Myers, Aneesh Sharma, Pankaj Gupta, and Jimmy Lin. Information Network or Social Network? The Structure of the Twitter Follow Graph. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 493–498, 2014.

Shamkant Navathe, Stefano Ceri, Gio Wiederhold, and Jinglie Dou. Vertical Partitioning Algorithms for Database Design. *ACM Transactions on Database Systems*, 9(4):680–710, 1984.

Neo4j, Inc. Statistics and execution plans – Operations Manual. https://neo4j.com/docs/operations-manual/current/performance/statistics-execution-plans/#neo4j-statistics, 2023a. Accessed: 2023-07-16.

Neo4j, Inc. Validate configurations – Operations Manual. https://neo4j.com/docs/operations-manual/current/tools/neo4j-admin/validate-config, 2023b. Accessed: 2023-07-16.

Mark Newman, Albert-László Barabási, and Duncan J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, 2006.

Mark E. J. Newman. Assortative Mixing in Networks. *Physical Review Letters*, 89(20), 2002.

Mark E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46 (5):323–351, 2005.

Mark E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

Mark E. J. Newman. *Networks: An Introduction*. Oxford University Press, 2010.

Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2004.

Matthew Newville, Till Stensitzki, Daniel B. Allen, Michal Rawlik, Antonino Ingargiola, and Andrew Nelson. Lmfit: Non-Linear Least-Square Minimization and Curve-Fitting for Python. *Astrophysics Source Code Library*, 2016.

Kim Norgaard and Eoin McSweeney. Christian Eriksen suffered cardiac arrest during Euros match and 'was gone' before resuscitation, doctor says | CNN. https://edition.cnn.com/20

21/06/13/football/christian-eriksen-stable-spt-intl/index.html, 2021. Accessed: 2023-05-11.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab, 1999.

Luci Pangrazio and Julian Sefton-Green. The social utility of 'data literacy'. *Learning, Media and Technology*, 45(2):208–220, 2020.

Alex Petrov. *Database Internals: A Deep Dive into How Distributed Data Systems Work*. O'Reilly Media, Inc., 2019.

PostgreSQL Global Development Group. PostgreSQL: Documentation: 16: 70.4. Implementation. https://www.postgresql.org/docs/current/gin-implementation.html, 2023a. Accessed: 2023-12-22.

PostgreSQL Global Development Group. PostgreSQL: Documentation: 15: 70.1. Introduction. https://www.postgresql.org/docs/current/gin-intro.html, 2023b. Accessed: 2023-07-26.

PostgreSQL Global Development Group. PostgreSQL: Documentation: 16: F.30. pg_prewarm — preload relation data into buffer caches. https://www.postgresql.org/docs/current/pgprewarm.html, 2023c. Accessed: 2023-12-13.

Quote Investigator. Everything Is Connected To Everything Else – Quote Investigator®. https://quoteinvestigator.com/2022/03/31/connected, 2022. Accessed: 2023-11-03.

Tommaso Radicioni, Tiziano Squartini, Elena Pavan, and Fabio Saracco. Networked partisanship and framing: A socio-semantic network analysis of the Italian debate on migration. *PLOS ONE*, 16(8):1–24, 2021.

Muhammad Rafi, Farnaz Amin, and Mohammad Shahid Shaikh. Document clustering using graph based document representation with constraints. *arXiv preprint arXiv:1412.1888*, 2014.

Douglas Reynolds. *Gaussian Mixture Models*, pages 659–663. Springer, 2009.

Stefano Rinaldi, Federico Bonafini, Paolo Ferrari, Alessandra Flammini, Emiliano Sisinni, and Devis Bianchini. Impact of Data Model on Performance of Time Series Database for Internet of Things Applications. In *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6, 2019.

Marko A. Rodriguez and Peter Neubauer. Constructions from Dots and Lines. *arXiv preprint arXiv:1006.2361*, 2010.

Egor Rogov. Indexes in PostgreSQL — 4 (Btree) / Habr. https://habr.com/p/443284, 2019. Accessed: 2023-12-12.

Francois Role and Mohamed Nadif. HANDLING THE IMPACT OF LOW FREQUENCY EVENTS ON CO-OCCURRENCE BASED MEASURES OF WORD SIMILARITY – A Case Study of Pointwise Mutual Information. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, pages 218–223, 2011.

Peter Rosenmai. Using the Median Absolute Deviation to Find Outliers. https://eurekastatistics.com/using-the-median-absolute-deviation-to-find-outliers, 2013. Accessed: 2024-01-09.

Giulio Rossetti and Rémy Cazabet. Community Discovery in Dynamic Networks: A Survey. *ACM Computing Surveys*, 51(2):1–37, 2018.

Christopher Rost, Kevin Gomez, Matthias Täschner, Philip Fritzsche, Lucas Schons, Lukas Christ, Timo Adameit, Martin Junghanns, and Erhard Rahm. Distributed temporal graph analytics with GRADOOP. *The VLDB Journal*, 31:375–401, 2022.

Christopher Rost, Kevin Gomez, Peter Christen, and Erhard Rahm. Evolution of Degree Metrics in Large Temporal Graphs. *BTW 2023*, 2023.

Martin Rosvall and Carl T. Bergstrom. Mapping Change in Large Networks. *PLOS ONE*, 5:1–7, 2010.

Martin Rosvall, Daniel Axelsson, and Carl T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178:13–23, 2009.

Camille Roth and Jean-Philippe Cointet. Social and Semantic Coevolution in Knowledge Networks. *Social Networks*, 32:16–29, 2010.

François Rousseau and Michalis Vazirgiannis. Graph-of-Word and TW-IDF: New Approach to Ad Hoc IR. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 59–68, 2013.

Sina Sajadmanesh, Sogol Bazargani, Jiawei Zhang, and Hamid R. Rabiee. Continuous-Time Relationship Prediction in Dynamic Heterogeneous Information Networks. *ACM Transactions on Knowledge Discovery from Data*, 13(4):1–31, 2019.

Nicola Santoro, Walter Quattrociocchi, Paola Flocchini, Arnaud Casteigts, and Frederic Amblard. Time-Varying Graphs and Social Network Analysis: Temporal Indicators and Metrics. *arXiv preprint arXiv:1102.0629*, 2011.

Martin Saveski, Brandon Roy, and Deb Roy. The Structure of Toxic Conversations on Twitter. In *Proceedings of the Web Conference 2021*, pages 1086–1097, 2021.

Michael Schuhmacher and Simone Paolo Ponzetto. Knowledge-Based Graph Document Modeling. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 543–552, 2014.

Hans-Jürgen Schönig. CLUSTER: Improving PostgreSQL performance – CYBERTEC. https://www.cybertec-postgresql.com/en/cluster-improving-postgresql-performance, 2020. Accessed: 2023-08-14.

Sandro Serpa and Carlos Miguel Ferreira. Micro, Meso and Macro Levels of Social Analysis. *International Journal of Social Science Studies*, 7:120–124, 2019.

Niladri Sett, Saptarshi Basu, Sukumar Nandi, and Sanasam Ranbir Singh. Temporal link prediction in multi-relational network. *World Wide Web*, 21:395–419, 2018.

Shilpy Sharma, David A. Swayne, and Charlie Obimbo. Trend analysis and change point techniques: a survey. *Energy, Ecology and Environment*, 1(3):123–130, 2016.

Chuan Shi and Philip S. Yu. *Heterogeneous Information Network Analysis and Applications*. Springer, 2017.

Chuan Shi, Ran Wang, Yitong Li, Philip S. Yu, and Bin Wu. Ranking-Based Clustering on General Heterogeneous Information Networks by Network Projection. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, pages 699–708, 2014.

Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. A Survey of Heterogeneous Information Network Analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29: 17–37, 2017.

solid IT gmbh. DB-Engines Ranking – popularity ranking of relational DBMS. https://db-engines.com/en/ranking/relational+dbms, 2023. Accessed: 2023-07-07.

Shahar Somin, Yaniv Altshuler, Goren Gordon, Alex 'Sandy' Pentland, and Erez Shmueli. Network Dynamics of a Financial Ecosystem. *Scientific Reports*, 10, 2020.

Andreas Spitz. *Implicit Entity Networks: A Versatile Document Model*. PhD thesis, Heidelberg University, Faculty of Mathematics and Computer Science, 2019.

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

Andreas Spitz and Michael Gertz. Entity-Centric Topic Extraction and Exploration: A Network-Based Approach. In *Advances in Information Retrieval*, pages 3–15, 2018.

Andreas Spitz, Satya Almasian, and Michael Gertz. TopExNet: Entity-Centric Network Topic Exploration in News Streams. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, page 798–801, 2019.

Statista, Inc. Biggest social media platforms 2023 | Statista. https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users, 2023. Accessed: 2023-11-02.

Benjamin Steer, Felix Cuadrado, and Richard Clegg. Raphtory: Streaming analysis of distributed temporal graphs. *Future Generation Computer Systems*, 102:453–464, 2020.

Dario Stojanovski, Ivica Dimitrovski, and Gjorgji Madjarov. TweetViz: Twitter Data Visualization. *Proceedings of the Data Mining and Data Warehouses Conference 2014*, 2014.

Michael Stonebraker and Lawrence A. Rowe. The Design of POSTGRES. *SIGMOD Record*, 15 (2):340–355, 1986.

Wen Sun, Achille Fokoue, Kavitha Srinivas, Anastasios Kementsietsidis, Gang Hu, and Guotong Xie. SQLGraph: An Efficient Relational-Based Property Graph Store. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1887–1901, 2015.

Yizhou Sun and Jiawei Han. Mining Heterogeneous Information Networks: A Structural Analysis Approach. *ACM SIGKDD Explorations Newsletter*, 14(2):20–28, 2013.

Yizhou Sun, Yintao Yu, and Jiawei Han. Ranking-Based Clustering of Heterogeneous Information Networks with Star Network Schema. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 797–806, 2009.

Yizhou Sun, Jie Tang, Jiawei Han, Manish Gupta, and Bo Zhao. Community Evolution Detection in Dynamic Heterogeneous Information Networks. In *Proceedings of the 8th Workshop on Mining and Learning with Graphs*, pages 137–146, 2010.

Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011.

Timescale, Inc. Timescale Docs. https://docs.timescale.com, 2023. Accessed: 2023-07-07.

Timescale Inc. Timescale Documentation | About hypertables. https://docs.timescale.com/use-timescale/latest/hypertables/about-hypertables, 2024. Accessed: 2024-01-09.

Vincent A. Traag, Ludo Waltman, and Nees Jan Van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9:1–12, 2019.

Chris Travers. postgresql – Postgres CLUSTER on JSONB GIN index? – Stack Overflow. https://stackoverflow.com/a/51978696, 2018. Accessed: 2023-08-14.

Andranik Tumasjan, Timm Sprenger, Philipp Sandner, and Isabell Welpe. Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. *Proceedings of the International AAAI Conference on Web and Social Media*, 4:178–185, 2010.

Jonathan H. Turner. *Theoretical Principles of Sociology, Volume 1: Macrodynamics*. Springer, 2010.

Twitter, Inc. Twitter API v2 example payloads | Docs | Twitter Developer Platform. https://developer.twitter.com/en/docs/twitter-api/data-dictionary/example-payloads, 2022. Accessed: 2022-10-25.

Twitter, Inc. Counting characters | Docs | Twitter Developer Platform. https://developer.twitter.com/en/docs/counting-characters, 2023a. Accessed: 2023-01-12.

Twitter, Inc. Twitter Trends FAQ – trending hashtags and topics. https://help.twitter.com/en/using-twitter/twitter-trending-faqs, 2023b. Accessed: 2023-01-12.

Alexander J. A. M. van Deursen and Jan A. G. M. van Dijk. Using the Internet: Skill related problems in users' online behavior. *Interacting with Computers*, 21(5–6):393–402, 2009.

Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A Comparison of a Graph Database and a Relational Database: A Data Provenance Perspective. In *Proceedings of the 48th Annual Southeast Regional Conference*, pages 1–6, 2010.

M. K. Vijaymeena and K. Kavitha. A Survey on Similarity Measures in Text Mining. *Machine Learning and Applications: An International Journal (MLAIJ)*, 3:19–28, 2016.

Riina Vuorikari, Yves Punie, Stephanie Carretero, and Lieve Van den Brande. DigComp 2.0: The Digital Competence Framework for Citizens. Update Phase 1: The Conceptual Reference Model. Technical report, European Union, 2016.

Sibo Wang, Wenqing Lin, Yi Yang, Xiaokui Xiao, and Shuigeng Zhou. Efficient Route Planning on Public Transportation Networks: A Labelling Approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 967–982, 2015.

Xiao Wang, Yuanfu Lu, Chuan Shi, Ruijia Wang, Peng Cui, and Shuai Mou. Dynamic Heterogeneous Information Network Embedding With Meta-Path Based Proximity. *IEEE Transactions on Knowledge and Data Engineering*, 34(3):1117–1132, 2022.

Yishu Wang, Ye Yuan, Yuliang Ma, and Guoren Wang. Time-Dependent Graphs: Definitions, Applications, and Algorithms. *Data Science and Engineering*, 4:352–366, 2019.

Franz Wanner, Andreas Weiler, and Tobias Schreck. Topic Tracker: Shape-based Visualization for Trend and Sentiment Tracking in Twitter. In *Task-Driven Analysis of Social Media: The 2nd Workshop on Interactive Visual Text Analytics*, 2012.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

Harry Wiener. Structural Determination of Paraffin Boiling Points. *Journal of the American Chemical Society*, 69:17–20, 1947.

Wikimedia Foundation, Inc. Media (communication) – Wikipedia. `https://en.wikipedia.org/wiki/Media_(communication)`, 2023. Accessed: 2023-11-10.

Markus Winand. The right column order in multi-column indexes. `https://use-the-index-luke.com/sql/where-clause/the-equals-operator/concatenated-keys`, 2023. Accessed: 2023-08-14.

Yong-Ting Wu, He-Yen Hsieh, Xanno K. Sigalingging, Kuan-Wu Su, and Jenq-Shiou Leu. RIVA: A Real-time Information Visualization and Analysis Platform for Social Media Sentiment

Trend. In *2017 9th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 256–260, 2017.

Tao Xie, Yangjun Xu, Liang Chen, Yang Liu, and Zibin Zheng. Sequential Recommendation on Dynamic Heterogeneous Information Network. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2105–2110, 2021.

Mengjia Xu. Understanding Graph Embedding Methods and Their Applications. *SIAM Review*, 63(4):825–853, 2021.

B. Bui Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2): 267–285, 2003.

Vikas Yadav and Steven Bethard. A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. *arXiv preprint arXiv:1910.11470*, 2019.

Jaewon Yang and Jure Leskovec. Patterns of Temporal Variation in Online Media. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pages 177–186, 2011.

Weiwei Yang, Jordan Boyd-Graber, and Philip Resnik. A Discriminative Topic Model using Document Network Structure. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 686–696, 2016.

Yang Yang, Nitesh Chawla, Yizhou Sun, and Jiawei Hani. Predicting Links in Multi-relational and Heterogeneous Networks. In *2012 IEEE 12th International Conference on Data Mining*, pages 755–764, 2012.

Yang Yujie. A Survey on Information Diffusion in Online Social Networks. In *Proceedings of the 2020 European Symposium on Software Engineering*, pages 181–186, 2020.

Daniel Zeng, Hsinchun Chen, Robert Lusch, and Shu-Hsing Li. Social Media Analytics and Intelligence. *IEEE Intelligent Systems*, 25(6):13–16, 2010.

Leihan Zhang, Jichang Zhao, and Ke Xu. Who creates Trends in Online Social Media: The Crowd or Opinion Leaders? *Journal of Computer-Mediated Communication*, 21:1–16, 2016.

Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. A Data-Driven Graph Generative Model for Temporal Interaction Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 401–411, 2020.

Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. Bipartite network projection and personal recommendation. *Physical Review E*, 76(4), 2007.

Arkaitz Zubiaga, Damiano Spina, Raquel Martínez, and Víctor Fresno. Real-time classification of Twitter trends. *Journal of the Association for Information Science and Technology*, 66(3): 462–473, 2015.