Feasibility of Interactions and Network Inference of Online Social Networks

Effrosyni Papanastasiou

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS
DEPUIS 1257

Thèse de Doctorat de Sorbonne Université

École Doctorale Informatique, Télécommunications et Électronique

Institut des Systèmes Intelligents et de Robotique

Feasibility of Interactions and Network Inference of Online Social Networks

Effrosyni Papanastasiou

Defended publicly on **July 8, 2024**

Doctoral thesis defended in front of the following Committee:

Reviewers	Jean-Loup Guillaume	La Rochelle Université Professor
	Sylvain Lamprier	Université Angers Professor
Examiners	Clémence Magnien (Committee President)	CNRS, Sorbonne Université Research Director
	Tiphaine Viard	Télécom Paris Associate Professor
Supervisor	Nicolas Baskiotis	Sorbonne Université Associate Professor

This doctoral thesis has been prepared at:

Institut des Systèmes Intelligents et de Robotique
Campus Pierre et Marie Curie
Pyramide, Tour 55
4, place Jussieu
75005 Paris
France

Για τη μητέρα μου Κατερίνα..

Abstract

This thesis deals with the problem of network inference in the domain of Online Social Networks. The main premise of network inference problems is that the network we are observing is not the network that we really need. This is especially prevalent in today's digital space, where the abundance of information usually comes with crucial unreliability, in the form of noise and missing points in the data. However, existing approaches either ignore or do not guarantee to infer networks in a way that can explain the data we have at hand. As a result, there is an ambiguity around the meaning of the network that we are inferring, while also having little intuition or control over the inference itself. The goal of this thesis is to further explore this problem. To quantify how well an inferred network can explain a dataset, we introduce a novel quality criterion called feasibility. Our intuition is that if a dataset is feasible given an inferred network, we might also be closer to the ground truth. To verify this, we propose a novel network inference method in the form of a constrained, Maximum Likelihood-based optimization problem that guarantees 100% feasibility. It is tailored to inputs from Online Social Networks, which are well-known sources of unreliable and restricted data. We provide extensive experiments on one synthetic and one real-world dataset coming from Twitter/X. We show that our proposed method generates a posterior distribution of graphs that guarantees to explain the dataset while also being closer to the true underlying structure when compared to other methods. As a final exploration, we look into the field of deep learning for more scalable and flexible alternatives, providing a preliminary framework based on Graph Neural Networks and contrastive learning that gives promising results.

Keywords: online social networks - network inference - Bayesian modeling - expectation maximization - maximum likelihood - constrained optimization - graph neural networks - contrastive learning

Résumé

Cette thèse traite du problème de l'inférence de réseau dans le domaine des réseaux sociaux en ligne. L'hypothèse principale des problèmes d'inférence de réseau est que le réseau que nous observons n'est pas celui dont nous avons réellement besoin. Cela est particulièrement vrai dans l'espace numérique actuel, où l'abondance d'informations s'accompagne généralement d'un manque crucial de fiabilité, sous la forme de bruit et de points manquants dans les données. Cependant, les approches existantes ignorent ou ne garantissent pas l'inférence de réseaux d'une manière qui puisse expliquer les données dont nous disposons. Il en résulte une ambiguïté sur la signification du réseau inféré, en plus d'un manque d'intuition et de contrôle sur l'inférence elle-même. L'objectif de cette thèse est d'explorer plus avant ce problème. Pour quantifier la capacité d'un réseau inféré à expliquer un ensemble de données, nous introduisons un nouveau critère de qualité, appelé feasibility. Notre intuition est que si un ensemble de données est ``feasible'' en ce qui concerne le réseau inféré, celui-ci est un meilleur candidat que le cas échéant. Pour le vérifier, nous proposons une nouvelle méthode d'inférence de réseau sous la forme d'un problème d'optimisation contraint, basé sur le maximum de vraisemblance, qui garantit la feasibility à 100%. Cette méthode est adaptée aux données provenant des réseaux sociaux en ligne, qui sont des sources bien connues de données peu fiables et restreintes. Nous présentons des expériences sur un ensemble de données synthétiques et données réelles provenant de la plateforme Twitter/X. Nous montrons que la méthode proposée génère une distribution a posteriori des graphes qui garantit l'explication de l'ensemble de données tout en étant plus proche de la véritable structure sous-jacente. En guise d'exploration finale, nous nous penchons sur le domaine de l'apprentissage profond pour trouver des alternatives plus évolutives et plus flexibles, en fournissant un cadre préliminaire basé sur les réseaux neuronaux graphiques et l'apprentissage contrastif qui donne des résultats prometteurs.

Mots-clés: réseaux sociaux - inférence de réseau - modélisation bayésienne - algorithme espérance-maximisation - maximum de vraisemblance - optimisation sous contrainte - réseaux neuronaux graphiques - apprentissage contrastif

Acknowledgements

The completion of this doctoral thesis marks the ending of an important period of my life, both academically and personally. Completing this PhD is not only about the hours spent studying and solving problems but also about overcoming all the challenges that come with growing up and moving to another country in pursuit of a better future. Nothing of this would have been possible without the support of my family, friends and many of the people that I met both inside and outside the university.

First, I would like to thank my supervisor Nicolas Baskiotis for providing his crucial help at all the pivotal points of this thesis and its completion. I would also like to thank all the committee members for taking part in the examination of this work. Jean-Loup Guillaume and Sylvain Lamprier for their extensive review of the manuscript and their very useful and constructive feedback and Clémence Magnien and Tiphaine Viard for very kindly agreeing to be part of the examination jury. I am especially thankful to Clémence Magnien for her unconditional support as a member of LIP6's direction and parity group, whose professionalism and compassion helped me immensely in completing this PhD.

Sometimes difficult circumstances can lead to the most beautiful surprises and one of them was getting to know Céline Ghibaudo. Our long-hour discussions, your determination, support, and inspiring energy helped me get through this thesis and the challenges that may come with womanhood. I am equally grateful for getting to know Garance and Melina, thanks for all the laughs, the support and all of our fun, empowering discussions.

I especially cherish the memories from my short but wonderful visit in the University of Siena, Italy. Thanks to Franco Scarselli, Monica Bianchini and all of the amazing students of the lab who welcomed me as if I were part of their family (SAILab is all you need!). A special mention to Barbara Corradini, thank you for making a whole journey from Italy to Paris just for the defense of this thesis. I can't thank you enough for all the support, our very constructive complaining sessions and all of the empowering discussions we had together.

My warm thanks to all the members of the MLIA team where I spent most of the time during this thesis - thank you for welcoming me and helping me whenever needed. A special thanks to Marie for all the laughs and support and Agnés, Lise, Nisma and Yuan for making the atmosphere of the lab so friendly. I would also like to thank all the stuff of LIP6 and ISIR for their continuous support and both parity groups who keep striving towards a better environment for women and underrepresented members of the lab.

Of course, the completion of this thesis would not have been possible without the unconditional help and love from my friends and family. A big thanks to Marilena, who was always there to listen and support me during this thesis. Yannis, one of the biggest presents that this city awaited for me; thank you for the genuine love and making life so much more fun and easy. Olga, my forever best friend, I could write a whole new thesis with all the ways I should thank you. A big thanks to Lazy Women for helping me put my feelings into words and actions - Zsofi, I am so grateful for getting to know you, thank you for trusting and supporting me. My warmest thanks to Alain, Marianne and Pauline, for making Paris feel like home. Antoine, thank you for everything - thank you for always being by my side and helping me become a better version of myself. All this wouldn't have been the same without you.

Finally, without the support and sacrifices of my family nothing would have been possible. A big thanks to my godmother Kitty, for planting the seeds for me to come to France and for creating a sense of family here. My grandmother Dimitra, always ahead of her time, for showing me what a woman is capable of. My grandfather Pantelis, for teaching me about the power of knowledge and education - I know you would be proud. Marina and Antoni, for all the never-ending love and help every single time I needed it. Eleni, for your unconditional love and support, thank you for lighting up my life and for all your valuable lessons about life and happiness.

Lastly, and most importantly: Panteli, thank you for being the best and most supporting brother I could ever have. Dad, thank you for always keeping my back - no words could describe how thankful I am for having you. And lastly, this thesis is devoted to my beloved mom whom I miss so much. Thank you for making me who I am today and for providing me with amounts of love that can last for an eternity - all my life's milestones will be forever dedicated to you. *We will always have Paris..* ♡

Contents

List of Figures

List of Tables

List of Abbreviations

AI Articifical Intelligence. 100, 110

API Application Programming Interface. 12, 13, 25, 53

CEM Constrained Expectation Maximization. 18, 20, 56, 69–72, 76–97, 99, 100, 113, 117, 118, 124, 127–129, 132, 133

DL Deep Learning. 100, 108, 109, 121, 122, 128

EM Expectation Maximization. 44–47, 50, 51, 66, 69, 70, 77, 96, 105, 129, 156, 157

ER Erdős–Rényi. 35–37, 46, 52, 56, 69, 70, 72, 73, 76, 79–91, 93–95, 132

FeasCL Feasible Contrastive Learning. xvii, 120, 122, 124–130, 133, 161

FN false negative. 38, 39, 127

FP false positive. 38, 39, 127, 128

GCN Graph Convolutional Network. 116

GNNs Graph Neural Networks. 6, 19, 103, 109–113, 115–117, 122, 133

ML Machine Learning. 5, 100, 101, 103, 104, 110, 115, 128, 129, 161

MLE Maximum Likelihood Estimation. 43–47

NI network inference. 7, 21, 45, 53, 59, 111

OSNs Online Social Networks. 11, 12, 15–24, 26, 27, 29–33, 35–40, 42, 48, 49, 52, 53, 59, 62, 99–107, 109, 113, 117, 129, 162

RNNs Recurrent Neural Networks. 107–109

SBM Stochastic Block Model. 37, 38, 45–48, 52, 56, 69, 71–73, 76, 78–96, 132

SNA Social Network Analysis. 26, 31, 40

SSL self-supervised learning. 111

TN true negative. 38, 39, 127

TP true positive. 38, 39, 127, 128

1 Introduction

Contents

The main object of interest in this thesis is *graphs*, mathematical structures used to represent relationships between any kind of mathematical or real-life entity. The term comes from the ancient Greek word *graphē*, meaning *the process of writing* or *drawing*. A common modern-day example of a graph is the structure of the Internet, with web pages linking to each other. The invention of online social media provides a rich terrain for exploration as well: The way users interact with each other is inherently graph-like and can provide valuable insights into human behavior and dynamics. In natural sciences, graphs can be used to represent interactions between particles and other physical entities. Graphs are also widely

used in industrial applications, representing road networks and telecommunication infrastructures among others. Thanks to their inherent flexibility, the potential applications of graphs are endless, promising a better understanding of interconnected and often intricate systems.

In this thesis, we begin by introducing the scientific evolution of graphs and exploring the various perspectives from which they can be studied. We then highlight the importance of real-world data in the study of graphs, stressing some important issues that arise and that have been underexplored in literature. We introduce *network inference* as a class of methods that can address these problems, particularly focusing on their usage within online social networks. This subject will persist as the central theme throughout the remainder of the thesis.

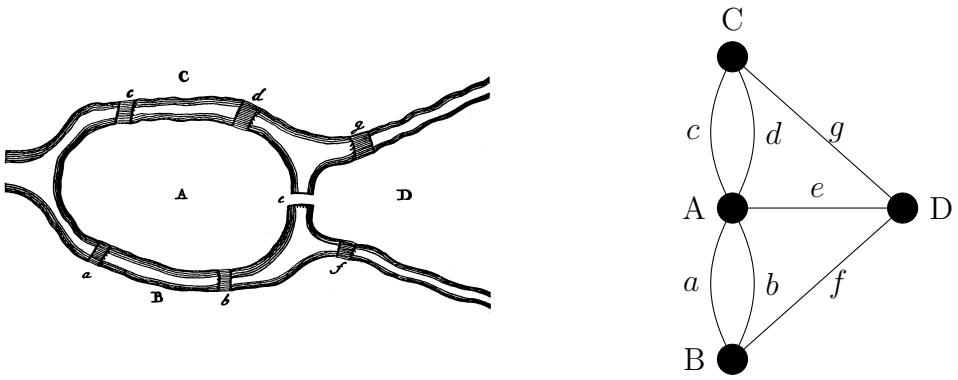
1.1 A brief history of graphs

1.1.1 The first problem using graphs

Graph theory is the field of mathematics dedicated to the study of graphs. The very first instance of a graph theory problem emerged in a rather recreational context, sparked by a series of historical events: During the 18th century, the Prussian city of Königsberg had become a thriving trading center, benefiting from its strategic position around the Pregel River. The economic prosperity of the locals allowed them to build seven bridges across the river, connecting the two mainland regions to two islands. It is claimed that the citizens of Königsberg spent their free time walking around the city, trying to devise a round-trip that crossed each of the seven bridges only once. Failing to find a solution, a Prussian mayor sought assistance from the renowned mathematician Leonhard Euler. Ironically, he considered the problem trivial, believing that its "discovery does not depend on any mathematical principle". Nevertheless, it intrigued him enough to continue thinking about a solution (quoted in Hopkins and Wilson, 2004):

> "*This question is so banal, but seemed to me worthy of attention in that [neither] geometry, nor algebra, nor even the art of counting was sufficient to solve it.*"

Although he did not explicitly mention graphs at the time, he drew what we would today identify as a graph, consisting of a set of *vertices* or *nodes* and a set of *edges* or *links* that connect them, to solve the problem: He represented each of the four land areas with letters $\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}$ and connected with lines each piece of land that had a bridge between them (Fig. 1.1a). With the help of this diagram, Euler



(a) *Euler's diagram* (b) *The Königsberg graph*

Fig. 1.1. – *The Seven Bridges of Königsberg.*

proved in his seminal paper that there can be no continuous walking path that crosses all the bridges while never crossing the same bridge twice (Euler, 1741).

The importance of Euler's diagram. Euler's proof is considered historically significant as the first time someone used a graph to solve a mathematical problem. It hinted at the idea that graphs are mathematical structures with intrinsic properties that are worthy of exploration, ultimately laying the foundations of graph theory.

1.1.2 From recreational puzzles to formal definitions

The visual representation we commonly use today for graphs (Fig. 1.1b) did not appear until the late 1800s, when mathematicians started using Euler's diagrams to solve the various recreational puzzles that were popular at the time[1]. In the twentieth century, the formal study of graphs finally gained momentum and evolved into an important branch of mathematics. Polish mathematician Dénes Kőnig wrote the first textbook in graph theory laying the groundwork for further exploration (König, 1936). Early analysis of tree structures by Cayley (1890) inspired many papers in combinatorics, and specifically in enumerating graphs with particular properties (Pólya, 1937; Gilbert, 1956). Meanwhile, the so-called *network flow* problems were popular, providing simplified models of real-life traffic flows, applicable to railway and road networks among others. The *shortest path* problems were attracting growing interest, with algorithms proposed by

[1]For an extensive overview of the nineteenth century recreational puzzles that led to graph theory, we refer the reader to the *Mathematical Recreations and Essays* by Ball (1893).

Dijkstra and Bellman-Ford becoming highly influential for both graph theory and computer science (Dijkstra, 1959; Ford, 1956; Bellman, 1958).

Random graph models. During the 1950s, a significant new branch of graph theory started to emerge: Advancements in combinatorics and graph enumeration drove the exploration of *random graphs*, a class of probabilistic or, equivalently, random processes capable of generating graphs with different properties. Two concurrent works stand out for establishing this field: (i) the seminal 1959 paper by Paul Erdős and Alfréd Rényi that proposed a random graph generation process where all graphs with a fixed number of vertices and a fixed number of edges are equally likely (Erdős and Rényi, 1959); and (ii) the model by Edgar Gilbert that was published independently in the same year, where each edge in a random graph has a fixed probability of being present or absent, independently of each other (Gilbert, 1959). As we will see in Section 1.1.4, while these works were highly influential within their discipline, their widespread adoption came decades later, as an essential tool for analyzing large real-world systems (Barabási, 2013).

Interdisciplinary research. In the 1960s, Frank Harary published his *Graph Theory* book where he emphasized how diverse the applications of graphs can be (Harary, 1969). In his own words:

> *"It has become fashionable to mention that there are applications of graph theory to some areas of physics, chemistry, communication science, computer technology, electrical and civil engineering, architecture, operational research, genetics, psychology, sociology, economics, anthropology, and linguistics."*

Indeed, an increasing number of researchers at the time were making the connections between their field and graph theory. A notable example is that of sociologist M. S. Granovetter and his highly cited paper *The strength of weak ties* (1973), where he compared the properties of the distance between individuals in a real-life social network with the mathematical properties of the distance between nodes in a graph.

1.1.3 The introduction of computers

Certainly, many of the early graph problems remain pertinent to this day; for example, the problem of determining in polynomial time whether two graphs are isomorphic (Whitney, 1932) remains unsolved. However, the *way* we solve these problems and any other kind of mathematical problem in general has changed

drastically since. The year 1977 specifically marked a historical turning point, when Appel, Haken, and Koch proved the *Four Color Theorem* by using assembly language (1977). Their work made computer-assisted proofs a real possibility, reshaping radically the scientific landscape (Tymoczko, 1979). Nevertheless, due to the persisting limitations of computational resources at that time, handling graphs on a large scale remained impossible; we lacked efficient mechanisms to process larger structures, thereby missing many opportunities in treating real-world examples that include millions, or even billions, of interactions at a time (Barabási, 2013).

1.1.4 Network science: graphs as complex systems

It was thanks to the advent of the *Internet* in the 1990s, and numerous innovations in the way we collect, store, and exchange information that we were finally able to treat a wide range of large interconnected networks, also known as *complex systems*. This term is used to reflect the complex interactions that may appear between components of any real-world system and that are difficult to model (Waldrop, 1993). Examples can be found nearly everywhere in real life, from the neuron-level interactions in the human brain and the biological structures of a living organism, to the broader scale systems of transportation, communication, and finance (Gell-Mann, 1994). The study of complex systems became the goal of *network science*, a field that integrates and combines theories from diverse domains like physics, computer science, graph theory and sociology (Newman, 2018). It eventually brought interdisciplinary attention to random graph models, proposed much earlier by Erdős-Rényi and Gilbert, as simplified means to simulate the behavior of many of these structures. Over time, new models were introduced to incorporate more realistic elements during the network generation process, such as the Barabási-Albert model (1999)[2].

1.1.5 Machine learning: a modern tool for graphs

The technological advancements of the twenty-first century in computational hardware and the increased accessibility of large-scale data, allowed treating graphs with a new set of tools, derived from *Deep Learning*, a specialized field of *Machine Learning (ML)* (LeCun et al., 2015). Central to the functioning of these methods is the use of *artificial neural networks*, a set of models inspired by the connections of neurons in the human brain (Rosenblatt, 1957). To address

[2]We will explore these models in further detail in Chapter 2.

irregularities and intricate patterns inherent in graph-structured data, a task also known as *graph representation learning*, a distinct class of artificial neural networks was introduced, also known as *Graph Neural Networks (GNNs)* (Gori et al., 2005; Scarselli et al., 2008). Particularly in recent years, GNNs are considered an extremely efficient architecture in settings where traditional models fail to perform, due to the interconnectivity of the input systems (T. N. Kipf and Welling, 2016).

In the present thesis

This thesis views graphs from two different perspectives: network science and machine learning. We will use the term *networks* when referring to real-world examples — such as social networks or the Internet — and the term *graphs* when discussing from a mathematical perspective.

1.2 The network is not the data

As in any scientific endeavor, when dealing with the study of graphs and networks, access to data providing genuine interactions is necessary (Butts, 2003). This allows for making informed decisions based on insights from real-world insights as well as developing more reliable graph models. For example, data from real-world networks can validate estimations made by theoretical models, and ensure that they accurately capture reality (Newman, Watts, et al., 2002). The data revolution that came about with the Internet democratized our access to this kind of information (Kitchin, 2014). Although not always explicitly structured as graphs, these public datasets detail diverse types of interactions, e.g., connections between users on social media platforms, collaborations between researchers in academic databases, links between web pages, etc. (Hu et al., 2020).

As suggested by Brugere et al. (2018), in many cases this data reflects reality in a straightforward way: Are two students enrolled in the same class? Are two researchers frequently citing each other? Are two products often purchased together? To respond, it suffices to observe the data in question. However, in other, more complicated examples, relationships between entities are only *indirectly* observable and must be *inferred* in a non-trivial manner: Do two individuals who frequently communicate online share a close friendship? During a global virus pandemic, who has infected whom? Are two proteins observed in the laboratory

interacting with each other? Arguably, creating a network representation that accurately responds to these questions is not as simple. As recently highlighted by Newman (2018):

> *"Some of the data may have no bearing at all on the network structure. Others maybe related only obliquely to it. And we may not know in advance which data are relevant and which are not, or how accurate any of the measurements are. Remarkably, under these seemingly daunting circumstances we can nonetheless make progress."*

In another recent work (Peel et al., 2022), the authors emphasize the fact that "whatever measurement we make is not necessarily "the" network we want to study, but includes at least some amount of distortion". These works bring forward a clear distinction between the *observed data* that we have available, and the *real network* underlying the observations. In many cases, these two can differ significantly due to errors and discrepancies in the experiments or missing information (Newman, 2018). Although early works in network science had already recognized this distinction (Goldberg and Roth, 2003), taking it explicitly into consideration remains extremely rare among practitioners (Peel et al., 2022). Naturally, an important question comes forward: *How can we access the real network that underlies a set of measurements or observations in a collection of data?* The response can be found in the context of *network inference (NI)*.

1.3 Network inference

In this thesis, we join the stance of research arguing that a clear distinction between the data and the underlying network should become common practice. We suggest that this could be addressed under a framework of *network inference*. It is a term that has not been given a precise definition; rather, it serves as a way to describe all methods employed when we wish to infer networks that are not directly provided by the data (Butts, 2003).

1.3.1 Why is it needed?

Network inference approaches are widely used in a variety of domains, combining tools from different fields of study such as statistics (Butts, 2003; Mukherjee and Speed, 2008), graph theory (Albert, 2007), network science (Goldberg and Roth, 2003), and physics (Newman, 2018b; T. Peixoto, 2018). One of its earliest uses is found in molecular biology, where it was proposed as a tool to recreate and explain complex interactions between proteins or genes (Friedman et al., 2000).

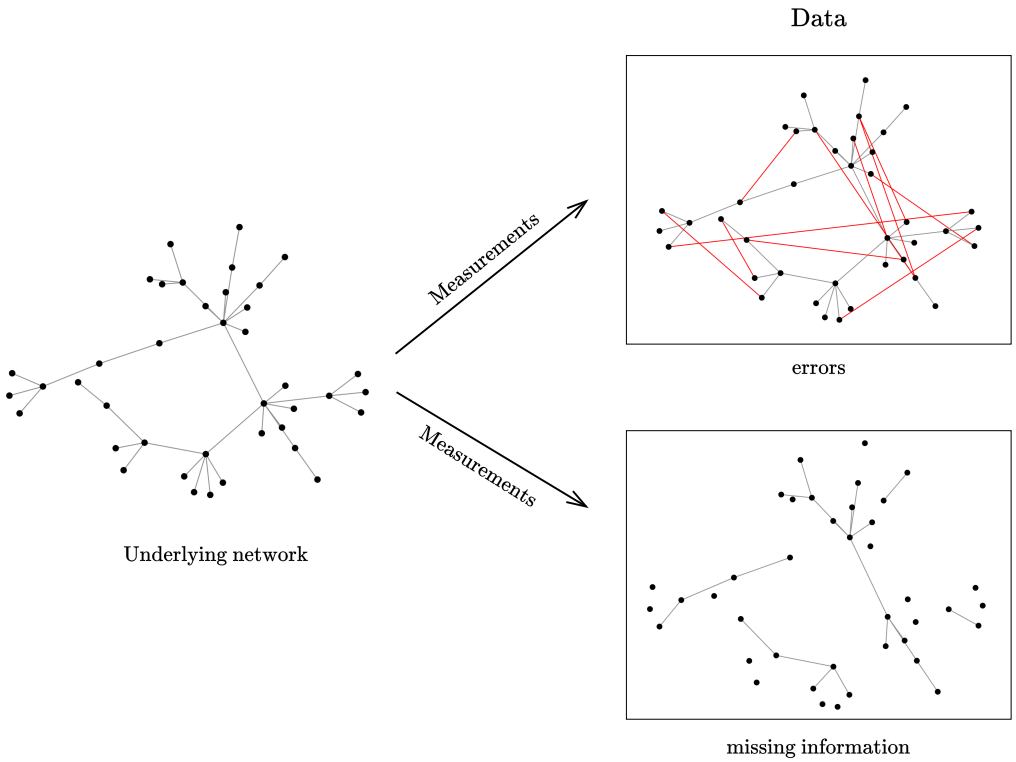


Fig. 1.2. – *A hypothetical underlying network (left) and examples of two different kinds of unreliable measurements that we might get: erroneous (right-top) and/or missing edges (right-bottom). Both kinds of measurements can distort significantly the structure of the underlying network.*

Many network inference methods have since been applied in other fields as well. Examples include social network analysis (Saito et al., 2008), epidemiology (X. Zhang et al., 2021; Firestone et al., 2020), finance (Giesecke et al., 2020), physics (Brugere et al., 2018), and telecommunications (J. Wu et al., 2022). Generalizing to any kind of setting, we can summarize some of the most common reasons why network inference might be needed into the following points:

– *Unreliability of measurements:* Many seminal works have emphasized the fact that experimental data, including data on networks, is often unreliable (Von Mering et al., 2002; Deane et al., 2002; Clauset et al., 2008). We consider two main sources of data unreliability that have been highlighted by several studies: the presence of errors in the measurements (Goldberg and Roth, 2003; Sprinzak et al., 2003; Butts, 2003), and missing information (Guimerà and Sales-Pardo, 2009). Both aspects might result in data measurements that are significantly different than the underlying network.

As we show in Fig. 1.2, confusing these experimental measurements of a network for its true underlying structure, without acknowledging the unreliability present, involves the risk of ignoring some of its unique structural properties. Unfortunately, ignoring this fact is still very common among network scientists and robust methods that take it into account are still understudied (T. Peixoto, 2018).

– *Representation of networks:* An interesting case in point is made by Butts (2009) who warns that a network representation is only an *approximation* of the underlying system and its construction involves assumptions that should always be made clear and explicit. A concrete example of this can be found in sociology: Social networks can be constructed via surveys and questionnaires (Marsden, 1990). However, the participants may often provide contradictory responses, posing a challenge for researchers to accurately represent these relationships in a network (Butts, 2003; Kossinets, 2006). Consequently, constructing the final network structure will necessarily require some assumptions or simplifications, potentially distorting the (hidden) reality. On top of that, experts sharing their network datasets are often omitting the assumptions made during their construction, making it even more challenging for the community to decipher underlying information (Pereira-Kohatsu et al., 2019).

– *Data is rich and multimodal:* Many datasets include measurements that extend beyond mere interactions between nodes. Data can be rich and multimodal, involving different sources of information such as text, video, images, etc (Newman, 2018b). Take social networks for instance, which can be measured by taking surveys of humans, collecting social media data, or gathering observations of face-to-face interactions with satellite signals (Stopczynski et al., 2014). However, not all interactions in a dataset are equally informative about the target structure. Some interactions may be irrelevant, making it challenging to choose among the most meaningful connections. Network inference methods offer a valuable alternative by indirectly deriving these connections, eliminating the need for manual selection (Newman, 2018b).

– *Collecting data is expensive:* Even when one is an expert of a domain, collecting data that accurately captures a target structure can be costly, time-consuming, or even impossible in some scenarios. For example, gaining

direct access to the topology of the human brain is still not experimentally possible, and therefore inference methods might be needed to map important brain regions in a non-invasive way (Hagmann et al., 2008). Even the design and implementation of more straightforward data collection strategies such as questionnaires and surveys can be incredibly time-consuming and expensive, making the need for indirect methods of measuring information such as network inference an important tool (Gao and J. Liu, 2016).

1.3.2 The challenges

Although crucial, implementing network inference methods is not an easy task (Mukherjee and Speed, 2008). Firstly, taking into account data unreliability during inference is not as straightforward, and requires consideration of different challenges, including quantifying the errors present, detecting the mechanisms that produced them, and addressing inherent uncertainties (2003). Another significant challenge that we have to confront is the vast space of possible networks that can be inferred even for moderately sized data (Mukherjee and Speed, 2008). Given that real-world networks usually involve a large number of nodes that interact with each other, sophisticated optimization strategies might be needed to infer networks in a time-efficient way (De Smet and Marchal, 2010).

Additionally, it is important to acknowledge that the nature of the data itself might introduce additional challenges, as different types of networks may exhibit distinct characteristics in terms of quality and structure. For example, it has been suggested that missing edges might be easier to infer in social networks compared to biological and technological networks due to properties such as the local clustering of nodes and assortativity (Ghasemian et al., 2020). Consequently, the development of a universal network inference method that effectively addresses all types of data is challenging. Tailor-made solutions are often necessary to tackle specific challenges (De Smet and Marchal, 2010), which may involve adapting existing network inference techniques or devising entirely new methods to accommodate the unique features of the data being analyzed (Ghasemian et al., 2020).

In the present thesis

Given the interdisciplinary nature of network inference applications and the domain expertise often needed to evaluate them in real-world scenarios, we narrow down our focus to a single domain: online social networks. The motivations behind this decision will become apparent in the next section.

1.4 Network inference for Online Social Networks

1.4.1 An introduction to Online Social Networks

Together with the Internet and the data revolution came what may be regarded as one of the most impactful innovations of the twenty-first century: *social media*. Central to their functioning are the interactions between users, who engage with each other's content through different mechanisms, such as posts, reposts, likes, comments, and follows, giving rise to a vast variety of *Online Social Networks (OSNs)*. As a result, social media provides a rich source of networks for all sorts of domains: In politics, for example, we can illuminate the dissemination of opinions or detect key influencers within political discourse (Cointet et al., 2021). Similarly, in marketing, we can identify potential brand advocates or predict consumer behavior based on social ties and interactions (Gomez-Rodriguez et al., 2012). Given that malicious users are very prevalent on these platforms, governments are engaging network experts to point out those responsible for spreading misinformation or encouraging harmful behaviors like hate speech and harassment (Shu, H. R. Bernard, et al., 2019; Pereira-Kohatsu et al., 2019; Pitsilis et al., 2018).

The ecosystem of social media *Box 1.4.1*

The first social media platform is considered to be SixDegrees.com, launched in 1997, where people could create their personal profiles under their real names and connect with their friends. Signaling an exciting new era of communication and information exchange, many similar ventures followed, with the likes of Friendster and Myspace gaining widespread popularity on an international level. Many of these early platforms have ceased to exist, but a handful of them remain extremely popular to this day: Facebook, YouTube, Twitter (today known as X), and Reddit are some of

the most notable examples, followed by later ventures, such as Weibo, Pinterest, Instagram, Snapchat and, more recently, TikTok. Each social media platform presents its distinct features that affect a network's characteristics and dynamics (see Box 1.4.1). For instance, while on some platforms like Facebook, the friendship is reciprocated, on others, like Instagram or Twitter/X, we have asymmetric follower-followee connections. As a result, each platform serves a different purpose with a direct impact on all aspects of modern life, from politics, science, and art, to professional and interpersonal relationships. Perhaps not surprisingly, their popularity is still ever-increasing: As of January 2024, there is an estimated 5.04 billion social media users — equating to more than 62% of the world's total population — and an average of more than 8 new users per second (DataReportal, 2024).

1.4.2 The need for network inference

Certainly, OSNs can provide numerous opportunities for exploration and analysis with significant implications for diverse domains. Nonetheless, the insights we can get, are only as good as the datasets that we have at hand or that we can potentially collect. In addition to errors or missing information as any kind of experimentally derived datasets, OSNs present some additional characteristics, unique to the way they operate. In this section we present the main challenges of dealing with this kind of data, highlighting the need for network inference methods:

– *Challenge 1: Access to social media data is limited.* What we can quickly realize when analyzing OSNs, is how challenging it is for researchers to collect data from social media platforms. Typically, via a public API, social media platforms provide researchers with some select information. For example, Twitter used to provide researchers with data such as tweets, retweets, friendships, etc. However, more informative details like who retweets whom are not included[3]. On top of that, in the latest years, we have been noticing a social media policy shift, where most major platforms are constraining access to their data. For example, since last year, access to the users' *friendship networks*, i.e., the networks that show who-follows-whom, has been prohibited on Twitter/X. A recent study is even suggesting that

[3]According to the Twitter API documentation of a Tweet Object, the "retweets of retweets do not show representations of the intermediary retweet, but only the original Tweet".

we are witnessing a "post-API" era with radical changes in data scraping strategies risking to alter research practices (Trezza, 2023). Even if we move towards platforms that provide data more openly, it can still get prohibitively time-consuming and labor-intensive to collect it (Failla and Rossetti, 2024). Consequently, in such cases, network inference methods can be incredibly valuable. They allow researchers to infer missing or inaccessible network structures based on available data in a cost-effective manner, thereby facilitating analysis and exploration even when direct data collection is challenging or restricted (Brugere et al., 2018).

– *Challenge 2: Uncovering hidden information is non-trivial.* Let us stay in the example of Twitter/X and think about what kind of information we might leverage. As demonstrated, the platform's API gives us access to tweets, their retweets and their respective timestamps, without revealing who-retweets-whom and who-follows-whom. Given this dataset, one may pose the following question: *How does a tweet posted on the platform reach the users that retweeted it?* We can start by trying to infer the (hidden) path that each tweet has taken, starting from its original author to the initial user that retweeted it, then to the second one, and so forth. If we make a simplified *closed-world assumption* that users retweet *only* from the users they follow, the inferred edges can also tell us who-follows-whom. For instance, for the first user that retweets some tweet, we can easily infer that they saw it directly from the author who created it, thus drawing an edge between the two. However, as we progress in time and onto the next users, it becomes exponentially harder to infer which edges are present or absent. Given that a dataset may include thousands, even millions of tweets and retweets, we are dealing with a challenging combinatorial problem. This is where a network inference method might be needed.

– *Challenge 3: Evaluating existing network inference methods is difficult.* Many methods have been proposed throughout the years that can provide solutions for scenarios like the one above, inferring the optimal networks that most accurately explain a set of timestamped interactions (Saito et al., 2008; Gomez-Rodriguez et al., 2012; Newman, 2018b; T. Peixoto, 2019). Of course, the need for these methods stems from the absence of information such as the intermediary retweet paths or the users' friendship networks. Therefore, we have no ground truth available to evaluate each method's results and an important question arises: What is considered a *useful* network inference result? How can we compare the many possible representations given by each method and what makes us prefer

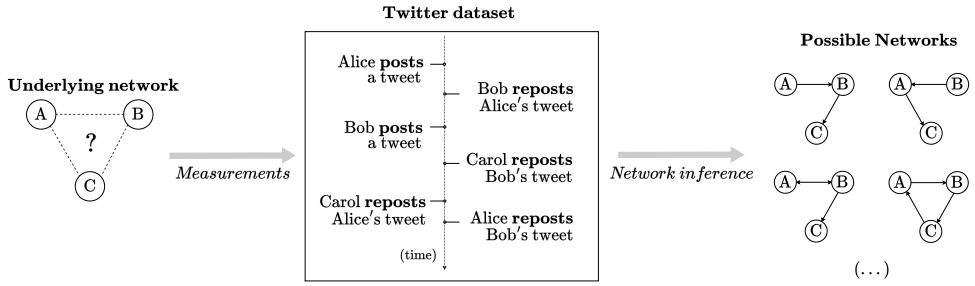


Fig. 1.3. – *A toy example of a network inference setting assuming data from the platform Twitter/X. An (unknown) underlying network gives a dataset or posting/reposting interactions from users Alice (A), Bob (B), and Carol (C). We can infer various combinations of connections between the users.*

the one to the other (Brugere et al., 2018)? An intuitive thing we can do is to interpret the results (i.e., the inferred edges) with respect to the input dataset. We may try to explain for example, why we observed a specific retweet in the input dataset, by looking into the inferred user-user connections. In the next section, we make this concept more concrete, by using a real-life inspired example.

1.4.3 A toy example

It might be easier to understand the above challenges by looking into an example of a toy Twitter/X dataset, as shown in Figure 1.3. We assume that an underlying network between the users of the platform gives rise to a set of measurements/observations, which makes the final dataset that we have at hand. As we observe in the figure, two tweets have been posted, one by user Alice and one by user Bob. Alice's tweet has been retweeted/reposted by Bob and then by Carol. Bob's tweet has been retweeted by Carol and then by Alice. The first question to ask is: *How are these three users connected to each other?* Looking at Alice's tweet we can non-trivially infer that Bob has retweeted her directly. Under our closed-world assumption, this means that Bob follows Alice. What about Carol? She could have retweeted Alice, Bob, or both. However, this information is hidden from us. Generally, we have 64 possible combinations of networks that could be inferred between the three users. Some of these are shown on the right side of the figure. They are *directed networks*, connecting the users according to the way that information diffuses. For example, if the directed edge (A, B) exists, it means that Alice can pass on information to Bob, or equivalently, that Bob can receive updates from Alice, i.e., he follows her. A second important question to ask then is: *Which of these networks satisfy us the most as a result?* To give a response, the

most intuitive way to proceed is to give an interpretation of the edges inferred by each network. For example, the upper left network tells us the following: (i) that Alice can diffuse information (e.g., a tweet) to user Bob; and (ii) that Bob can diffuse information to Carol. Given these connections, we can explain the retweets of Alice's tweet. However, we cannot do the same for Bob's tweet, since there is no path for the information to diffuse from its author Bob to the user Alice. As a result, the retweet by Alice remains unexplained, and we can therefore claim that we are not satisfied by the edges inferred in the selected network.

Towards feasible results. Generalizing this idea, we may wish to accept *only* the networks that provide sufficient explanations for all tweets and should therefore satisfy the two following requirements:

– *Requirement 1:* They provide at least one path from the authors to all the users that retweeted their tweets.

– *Requirement 2:* These paths abide by the temporal order of the observed retweets.

If we now examine the upper right network inferred in Figure 1.3, we can easily see that it satisfies the first requirement but violates the second: Bob's tweet cannot be explained by the inferred path $B \rightarrow A \rightarrow C$, since, by looking at the retweets' ordering, Alice could not have possibly diffused the information to Carol who retweeted it first. Therefore, for both of the two upper networks, we say that *the dataset is not feasible given the inferred network*. In the opposite case, if a network inference result satisfies both requirements, we say that *the dataset is feasible given the inferred network*, or, equivalently, that *the inferred network explains the dataset*. This is the case for the two lower networks. Of course, if we trivially inferred a complete graph, where all users were connected to each other, the dataset would also be feasible. However, this would not reflect the real properties of OSNs, which, as we will see in Chapter 2, present some unique characteristics that should be taken into account during inference.

1.5 Thesis goals

Given the above challenges, we are motivated to explore the following key points within this thesis:

– *Defining feasibility on OSNs:* One of the primary goals of this thesis is to propose the metric of *feasibility* as a requirement that must be met by a network

inference method applied to OSNs. This way we guarantee that the network can reproduce and explain all interactions observed in a dataset. The intuition behind this suggestion is that a network that can explain all the interactions and their chronological order inside the dataset is closer to the real one. Previously, we outlined the procedure for assessing feasibility using a specific example. Our goal now is to devise a more general definition that can be applied to any similar dataset derived from OSNs.

– *Exploring the feasibility of current network inference methods and its relation to the ground truth:* An accompanying goal of this thesis is to examine experimentally: (i) to what extent the current network inference methods provide feasible results; and (ii) whether feasibility does indeed guide to results that are closer to the underlying network, thus serving as an indicative metric of inference quality. To achieve this, we gather the friendship network that underlies a real-world Twitter/X interactions dataset. This is an important step since it will stand as a meaningful ground truth against which we can compare the precision of the inferred results and its relation to feasibility. Interestingly, this evaluation approach has been largely overlooked in existing literature, primarily due to the scarcity of comprehensive datasets containing both the network measurements and the underlying connections to compare with. This is an important step of the thesis towards bridging this gap.

– *Developing a network inference method that guarantees feasibility:* At first glance, inferring networks that can explain the given dataset may seem as simple requirement. However, as we will demonstrate in this thesis, it is a non-trivial task to achieve, and current network inference methods do not take it into consideration. As a result, another aim of this thesis is to propose a network inference approach that can infer networks while guaranteeing the feasibility of the results. Of course, in reality, a portion of the observed interactions can originate from indirect sources. For example, users might encounter suggested content from individuals they do not directly follow, as determined by the platform's recommendation algorithm. To address these scenarios, we also intend to explore ways to relax the strictness of the feasibility requirement.

By exploring these aspects, we contribute to the literature of network inference, particularly in the context of OSNs. We aim to provide a network inference framework that holds particular value in scenarios characterized by unreliable and inadequate data. This involves not only proposing a novel network inference

framework with which we can infer networks that can explain the input dataset but also the introduction of a novel evaluation method based on the metric of feasibility. It is important to highlight that although our data comes from online social networks, the methodologies, and mathematical solutions presented in this thesis hold potential for application in other domains as well.

1.6 Thesis structure and contributions

We will now provide an overview of the thesis' overall structure, summarizing the contributions of each chapter:

Chapter 2: Fundamentals of network inference

This chapter provides the network inference fundamentals designed to draw inferences from unreliable datasets originating from OSNs. We begin by introducing the concept of network data unreliability, primarily defined by the presence of errors and missing values in the data. We explore how this unreliability impacts information coming from OSNs, emphasizing the distinction between the raw data and the network we aim to infer. We then demonstrate how OSNs can be represented as graphs within a network inference framework and discuss various metrics that characterize their unique properties, which must be considered during inference. Additionally, we examine random graph models used to infer synthetic graphs with well-defined properties, highlighting their theoretical significance as benchmarks for comparing with real-world networks. Lastly, we present standard metrics and best practices for assessing the inferred edges. These concepts are crucial for this thesis, forming the foundation of the network inference framework we aim to develop.

Chapter 3: Network inference approaches for Online Social Networks

This chapter offers an overview of the existing literature concerning network inference from unreliable data. We present various network inference approaches, with the principal one being Bayesian modeling. Given that our particular case of OSNs data comes with timestamps, we also look into the inference methods tailored specifically to temporal data. These may pose additional challenges, particularly in the way that the time aspect is represented. Finally, we highlight the limitations of these existing works, especially regarding their evaluation process. This discussion will provide additional motivation for proposing a new network

inference evaluation metric, as well as developing a novel network inference method that offers a better alternative to current techniques.

Chapter 4: Constrained Expectation Maximization for feasible network inference

This chapter presents the main contribution of this thesis: a novel network inference evaluation metric called feasibility and a network inference method, namely Constrained Expectation Maximization (CEM), to guarantee it. Given that our algorithm relies on real-world datasets from user interactions on OSNs, we first show how to model the information stemming from such datasets. We then formally define the concept of feasibility in the context of OSNs. To ensure feasible results, we develop a novel network inference algorithm that incorporates a set of *feasibility constraints* to account for all possible hidden paths given timestamps of user interactions. We formulate the feasibility guarantee as a *linear* constrained optimization problem, which simplifies the computation. It eventually infers a posterior distribution of feasible underlying networks that explain the provided dataset, while respecting the chronological order of the interactions observed. We provide extensive experimental evaluations on both a synthetic and a real-world Twitter dataset consisting of nearly 300,000 tweets and over 1,600,000 retweets related to the 2017 French presidential elections. We compare with other network inference methods in terms of feasibility and two additional aspects: (i) their real-world properties and (ii) their proximity to the actual underlying friendship network. To achieve the latter for the real-world Twitter dataset, we had to manually collect the friendship network, which was still allowed by the platform's API at the time. As mentioned in Section 1.5, this evaluation step is an important contribution since current network inference methods rarely compare their results with the actual underlying network, relying instead on heuristics or artificially generated data. We report the results suggesting that our proposed method can ensure feasibility while providing better approximations for the ground truth, compared to existing network inference methods.

The contributions of this chapter have led to the following publications:

□ Effrosyni Papanastasiou, and Anastasios Giovanidis. Constrained expectation-maximisation for inference of social graphs explaining online user–user inter-

actions." *Journal of Social Network Analysis and Mining 13.1, Springer 2023: 41.* (Papanastasiou and Giovanidis, 2023)

☐ Effrosyni Papanastasiou, and Anastasios Giovanidis. Bayesian inference of a social graph with trace feasibility guarantees. *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), ACM, 2021.*

Additionally, the Twitter friendship network that we collected for the purposes of our study has also contributed to a different domain, beyond the scope of this thesis, specifically in the context of opinion dynamics: ☐ Antoine Vendeville, Anastasios Giovanidis, Effrosyni Papanastasiou, and Benjamin Guedj. Opening up echo chambers via optimal content recommendation. *International Conference on Complex Networks and Their Applications (pp. 74-85). Cham: Springer International Publishing.* (Vendeville et al., 2022a)

☐ Antoine Vendeville, Anastasios Giovanidis, Effrosyni Papanastasiou, and Benjamin Guedj. Recommendation of content to mitigate the echo chamber effect. Extended abstract. *Conference on Complex Systems, 2022.* (Vendeville et al., 2022b)

Chapter 5: A contrastive approach using Graph Neural Networks

Motivated by the poor scalability of constrained optimization problems for applications with large-scale data, this chapter investigates an alternative feasible network inference framework that relies on the field of Artificial Intelligence. We begin by briefly introducing the subfield of machine learning, then focusing on techniques adapted to graph-shaped data. A comprehensive review of literature follows, presenting diverse modeling concepts that could be relevant to the domain of network inference given data from OSNs. Their main difference to the previous framework is that they learn latent node embeddings instead of probability distributions for the entire structure. We discuss the weaknesses of these methods, primarily focusing on their evaluation processes and their omission to account for unreliability. This motivates us to propose a novel alternative: We first suggest representing the general network inference problem in a self-supervised learning framework. It leverages an encoder model often employed for link prediction tasks, based on the use of Graph Neural Networks (GNNs). To incorporate the concept of feasibility, we propose a simple contrastive loss that can learn node embeddings incorporating the notion of time via a simple sampling approach. The basic idea behind this is that the node embeddings of the

users who could potentially be connected in the underlying graph in a feasible fashion should also be encoded closer in the latent space. We run preliminary experiments to compare against CEM and a simple deep learning baseline. The results are promising, and promote an exciting new direction which could potentially improve the interpretability of the learning process along with the scalability of the network inference problem.

Chapter 6: Conclusion

In this final chapter, we summarize the contributions of this thesis for the problem of feasible network inference in the domain of OSNs and reflect on our findings. We also address the main limitations stemming from our modeling decisions and discuss potential avenues for future research.

* * *

2 Fundamentals of network inference

Contents

As demonstrated in the introduction, graphs can be found everywhere in real life, offering a mathematical framework to represent the intricate relationships found in biological, physical, and social networks, among other domains (Newman, 2018c). Our ability to access these graphs is made possible by a vast repository of publicly available datasets, accessible to everyone for exploration. However, these datasets are unreliable, and may differ substantially from the actual networks they represent, making necessary the use of network inference (NI) methods. In this thesis, we are interested in the data coming from Online Social Networks (OSNs), which, as we explained earlier present some additional challenges (Section 1.4). As a result, this chapter presents some of the background needed to deal with this problem. In Sections 2.1 and 2.2 we make the distinction between the data and the network more concrete with several studies to support this claim. Then, we show how we can formalize this problem on OSNs by representing

them as graphs, illustrating it with an artificial dataset example. Then, Section 2.3 presents different metrics from graph theory to characterize some of the characteristics and real-world properties of OSNs that we would ideally wish for a network inference method to capture. In Section 2.4, we compare the real-world properties of OSNs to these of random networks, an important class of networks often used as prior knowledge when the exact structure is not yet known. Finally, Section 2.5 presents the most common metrics for experimentally evaluating the efficacy of a network inference method, underlining the potential challenges involved.

2.1 Network data unreliability

In the Introduction, we established that OSNs datasets providing measurements or observations may lack reliability, a quality that is inherent to any kind of empirical data collection process (Clauset et al., 2008). Especially in recent years, with a wealth of network data at our disposal, there is a growing consensus that *all* empirical networks include some kind of unreliability which should be taken into account (Newman, 2018; T. Peixoto, 2018). However, incorporating it systematically is still far from common practice among network scientists, and the development of methods to address it is deemed as understudied (Peel et al., 2022). The first important step towards that goal is to make a distinction between the *observed* network, as given by the data, and the *real* network that exists and which we are trying to capture. The difference between the two has been extensively documented in the literature and focuses primarily on two aspects: error and incompleteness. Though distinguishing between them might not always be straightforward, errors typically manifest as false positive edges (interactions observed but not actually present in the real network), whereas incompleteness results in false negative edges (interactions missing from the data but existing in the real network) (Guimerà and Sales-Pardo, 2009).

Networks contain errors. The assessment of the errors in the collected data emerged as a topic of research as early as the 1970s (Killworth and H. Bernard, 1976; H. R. Bernard et al., 1984; Freeman et al., 1987; Marsden, 1990). Sociologists, for instance, began questioning the viability of self-report data due to concerns about its error levels and its reliability in inferring genuine interaction patterns (Butts, 2003). These concerns extend to other domains, most notably biological networks such as protein-protein interaction networks, which are experimentally derived and prone to significant false positive errors (Von Mering et al., 2002;

Deane et al., 2002; Goldberg and Roth, 2003, Sprinzak et al., 2003). Errors may not only occur at the level of edges but also affect nodes. For instance, in collaboration networks linking researchers who co-author papers, erroneous nodes might appear due to individuals sharing identical names or appearing under different spellings, formats, or affiliations (Peel et al., 2022).

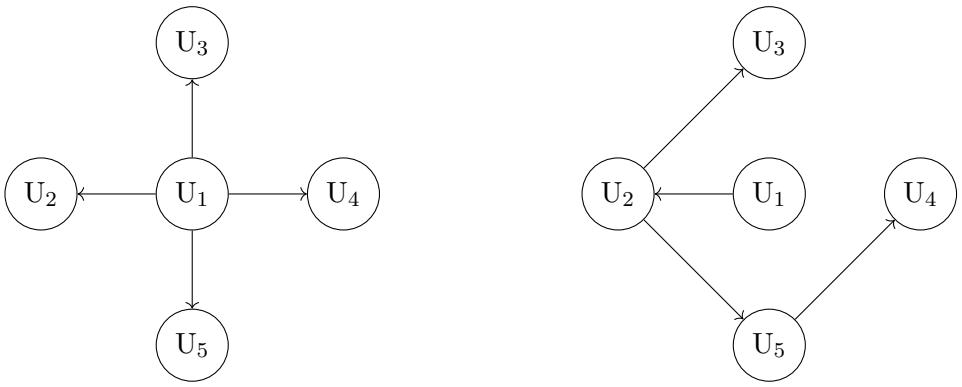
Networks are incomplete. In many experimental settings, discovering network interactions is an extremely challenging task (Clauset et al., 2008). Consequently, the networks we gather often prove to be incomplete, failing to capture numerous interactions critical to the task at hand (Guimerà and Sales-Pardo, 2009). For example, studies have found that protein-protein interaction networks exhibit a substantial number of false negatives alongside false positives: collected data may overlook up to 90% of the interactions in the yeast interactome (Ito et al., 2001; Yu et al., 2008), and in the case of the human interactome, this figure can rise as high as 99.7% (Stumpf et al., 2008; Amaral, 2008). This is especially prevalent in social sciences, where missing values may arise due to several reasons, such as survey participants omitting responses, dropping out, or being influenced by biases inherent in a study's design (Kossinets, 2006; Schafer and Graham, 2002).

2.2 Modeling Online Social Networks with graphs

To proceed, we first need to define OSNs mathematically. Usually, OSNs are defined similarly as a graph:

Graph. A dataset coming from an online social media platform can be represented by a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of user nodes (N in total) and $\mathcal{E}$ is the set of edges connecting them (E in total). The edges can be *directed* or *undirected*. A directed edge (i, j) is an edge that begins from a user node i and points to user node j.

Directed graphs are used to model asymmetric relationships or flows of information; an example is Instagram or Twitter/X, where one user can follow another without being followed back. Undirected graphs are used instead to model symmetric interactions or relationships, such as the friendships between users on platforms where connections are mutual, like Facebook or LinkedIn. Online social media platforms offer various types of directed or undirected OSNs, each capturing unique aspects of user interactions within their ecosystem. One of the most common information that is provided by public OSNs datasets is the reposting behavior of users, forming what we call as *interaction networks* (Brugere



(a) *The interaction network that is given* (b) *The interaction network that is hidden*

Fig. 2.1. – *A toy example of what kind of information could be hidden from us. User* U_1 *is assumed to be the author of the post and users* $\mathrm{U}_2, \mathrm{U}_3, \mathrm{U}_4, \mathrm{U}_5$ *the ones that reposted it. The interaction network we get is a star-shaped structure. On the right, we see one possible way that the post could have diffused in reality, but that is hidden from us in practice. If we assume that the users repost only the users they follow, this network could also be showing the friendship connections between the users.*

et al., 2018; Lin et al., 2016). On different platforms these may take different names — on Twitter/X, for instance, they are commonly referred to as *retweet networks* (Peng et al., 2011). By reposting other users' *posts* (*tweets*) one can express acknowledgment, endorsement, or even disagreement (Malhotra et al., 2012).

Let us assume that we have a reposting dataset that shows that a post by a user U_1 has been reposted by four other users denoted as $\mathrm{U}_2, \mathrm{U}_3, \mathrm{U}_4,$ and U_5. An example of an interaction network that can be derived from such a dataset is shown in Fig. 2.1a. It connects the user nodes directly to the author, in a star-shaped structure. The edges are *directed*, depicting the flow of information from one user to another. For each new post that is created, we can form separate interaction networks connecting the author and those who reposted it. To combine several posts and reposts into a single network, we can follow a threshold policy of our choice, drawing for example an edge if a user reposts frequently from a specific user (S. Myers and Leskovec, 2010). The question therefore that comes up naturally is the following: *Is the network of Fig. 2.1a created by such dataset the real interaction network?* Or, equivalently: *Did users* $\mathrm{U}_2 - \mathrm{U}_5$ *all really repost author* U_1 *directly?*

Responding to this question is not as simple as it seems. As we motivated in the introduction, a known limitation of OSNs is that they only provide partial infor-

mation about the behavior of the users, hiding the intermediate paths involved in the process of reposting. As a result, while the provided network of Fig. 2.1a accurately captures the endpoints of each repost interaction (i.e., the users who reposted the content by the original author U_1), it inherently lacks information about the intermediary users that were reposted. In reality, the are many ways through which the post could have been diffused — Fig. 2.1b shows one such possible scenario out of the many combinations.

How can we infer what is happening in reality? And what kind of network is the one that we should be looking for? As highlighted by Newman (2018) we lack the ability to specify the exact nature of the connections underlying a dataset. It could be anything between friendships, professional ties, proximity, or a combination of these relationships that work as a driving force behind the observed data. In the case of the interaction networks, one type of underlying information that could be of help is the so-called *friendship networks*, depicting the connections that can be formed between users by following each other. Formally, we assume that if a user U_1 follows another user U_2, the final friendship network will have a directed edge from user U_2 to U_1; we say that user U_1 is a *follower* of U_2, and user U_2 their *followee*.

A simplified assumption. If we make the simplification that users only repost the people they follow, one could simply look into the friendship connections, and give the potential paths that a post has taken through the users' reposts. However, this approach is very often restrained by a platform's API. This is the case of Twitter/X for instance, which no longer provides access to users' following and follower networks (Trezza, 2023). Even before this policy change, public datasets rarely were comprehensive enough to include both reposting and friendship networks, probably because of the strenuous and labor-intensive nature of data collection (Failla and Rossetti, 2024). As a result, we might need network inference methods that can infer non-trivially the real network underlying the observed sets of users' interactions/reposts.

In the present thesis

In this thesis, we assume that the target structure to be inferred given a dataset of reposts is the friendship network. Of course, this is not always the case, since one can also repost content from users they do not follow, e.g., after searching for

specific content via the platform's search bar or from accounts suggested by the platform's recommendation algorithm. As Kossinets et al. (2008) emphasized:"just because two individuals are acquainted does not imply that they have communicated within some particular time interval, in which case no information could have passed directly between them. Correspondingly, the indirect flow of information between individuals requires a sequence of communication events along a path of intermediaries linking them." These observations introduce additional challenges for the analysis of social networks, and can have important consequences for the relation between network structure and information flow (Kossinets et al., 2008; Holme, 2005; Onnela et al., 2007). For this reason, we are aiming to propose a network inference framework that addresses these challenges while maintaining the flexibility to adjust assumptions as necessary.

2.3 Understanding the properties of Online Social Networks

To infer the hidden network underlying our observations, we first need to understand better the structures that we have at hand. To do that, we can borrow tools from Social Network Analysis *Social Network Analysis (SNA)*, borrowing different measures from network science and graph theory to understand and analyze the complex relationships within social networks (Otte and Rousseau, 2002). Given the above definition, OSNs possess different properties that can be quantified with several local (node-level) and global (graph-level) measures. In this section, we present some of the most common measures and properties that are been highlighted in this context[1].

2.3.1 Local and global graph measures

Subgraph. In practice, it is very often the case that we may wish to look only into the connections of users with a specific characteristic, e.g., in terms of gender, nationality, or political affiliation. A common way to do this is to define a *subgraph* of the main graph $\mathcal{G}$, by selecting a subset of nodes that interest us the most and keeping only the edges that connect them. Formally, such subgraph can be defined as $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$, with a subset of nodes $\mathcal{V}_s \subseteq \mathcal{V}$ and a subset of edges $\mathcal{E}_s \subseteq \mathcal{E}$.

Degree. On a local level, each node in a graph $\mathcal{G}$ has a *degree*, denoted as $\deg(j)$, equal to the number of other nodes directly connected to it via an edge. In

[1]For a more extensive overview of all the useful network metrics we recommend the book *Networks* by Newman (2018).

directed graphs, the *in-degree*, is the number of incoming edges to the node, while the *out-degree* is the number of its outgoing edges.

Degree distribution. On a global level, we can define a *degree distribution* of the graph, denoted as $P(k)$, showing the fraction of nodes in the graph that have a degree k. The degree distribution can offer important insights into the network's global structure and connectivity patterns.

Neighborhood. The *neighborhood* $\mathcal{N}_i$ of a node i is defined as the set of its directly connected neighbors, both in-coming and out-coming:

$$\mathcal{N}_i = \{j : (i,j) \in \mathcal{E} \vee (j,i) \in \mathcal{E}\},$$

We denote with N_i the number of nodes in each neighborhood $\mathcal{N}_i$.

Local clustering coefficient. For a directed graph, there are $N_i(N_i - 1)$ possible edges that could exist among the nodes in the same neighborhood N_i. An important metric exists by making this observation that can quantify the extent to which the nodes of $\mathcal{G}$ tend to cluster tightly together. It is called the *local clustering coefficient* and is defined as (Watts and Strogatz, 1998):

$$C_i = \frac{|\{(i,j) : i,j \in \mathcal{N}_i, (i,j) \in \mathcal{E}\}|}{N_i(N_i - 1)}.$$

Its values range from 0 to 1, where a value of 1 indicates that all neighbors of a node are also neighbors of each other, forming a complete subgraph also known as *clique*. Nodes with high clustering coefficients in OSNs are often pivotal in spreading information or influencing the behavior of other nodes within their neighborhood.

Density. It is a measure that quantifies how many edges are present in $\mathcal{G}$ relative to the number of possible ones. For example, if no self-loops are allowed in $\mathcal{G}$ (i.e, a node cannot be connected to itself):

$$D = \frac{E}{N \times (N-1)}$$

Network density can range from 0 to 1, with a density of 0 indicating no edges and a density of 1 indicating that all edges are present.

Distance and average shortest path. The distance $d(i, j)$ between any two nodes i, j in G is the number of edges in $\mathcal{E}$ that need to be traversed in the shortest path that connects them. This *shortest path* is the path that connects the two nodes while also minimizing the total number of edges in $\mathcal{E}$ traversed (without allowing to pass from the same node twice). This path is not unique, and there may be multiple shortest paths between two nodes with the same length. To find the shortest path between two nodes, various efficient algorithms have been proposed, with the most classic ones being Dijkstra's and Bellman-Ford's (Bellman, 1958; Dijkstra, 1959). The *average shortest path* is a global measure that averages over all shortest paths in the graph.

Strongly and weakly connected component. A *strongly connected component* is a subgraph of $\mathcal{G}$ where every node can be reachable from any other node via a directed path. A *weakly connected component* instead is a subgraph of $\mathcal{G}$ where all nodes are connected to each other by any path, without taking into account the direction of the edges.

Diameter. It is a global measure of the graph's structure which shows the maximum distance required to travel between any two nodes:

$$d = \max_{i \in \mathcal{V}} \max_{j \in \mathcal{V}} d(i, j).$$

If $\mathcal{G}$ has a small diameter it may indicate that the nodes in the graph are closely connected, while a large diameter may indicate that the nodes in the graph are more spread out.

Other kinds of graphs. On top of being directed or undirected, a graph $\mathcal{G}$ can be classified under different definitions based on some of its measures. For example, $\mathcal{G}$ is called:

- *Disconnected*, if there are at least two nodes that are not connected by a path. A disconnected graph has an infinite diameter.
- *Directed Acyclic* (also called *DAG*), if it is directed and no node has a path that returns back to itself.
- *Complete*, if it has a density equal to one, meaning that all possible edges between nodes are present.

– *Strongly* or *weakly connected*, if the entire graph forms one single strongly or weakly connected component respectively. In this case, all nodes are connected to each other by at least one path.

In the present thesis

Of course, numerous other important metrics exist that could characterize the components of a graph structure in greater detail. For instance, with metrics like *PageRank*, *closeness*, and *betweenness* we could quantify the importance or centrality of individual nodes within a graph (Page et al., 1999; Bavelas, 1950; Freeman, 1977). In this thesis, we focus on measures that are able to characterize a graph as whole, and that can be used to describe the real-world properties of graphs coming from OSNs. As we will demonstrate shortly, evaluating these measures in relation to each other instead of in isolation, can give us new important insights into the properties of the observed graphs (Watts and Strogatz, 1998).

A TRIVIAL APPROACH TO NETWORK INFERENCE *BOX 2.3.1*

A trivial approach to deciding whether to infer an edge between a pair of nodes (i, j) is to employ different statistical measures that focus on the local or global structure. On a local level, for example, we can count the number of neighbors that i and j share in common:

$$\mathbf{S}_{ij} = |\mathcal{N}(i) \cap \mathcal{N}(j)|,$$

where $\mathcal{N}_i$ denotes the set of neighbors of a node i. If we want to minimize any biases present due to their degrees we can employ the *Jaccard index*:

$$\mathbf{S}_{ij}^{Jaccard} = \frac{|\mathcal{N}(i) \cap \mathcal{N}(j)|}{|\mathcal{N}(i) \cup \mathcal{N}(j)|}.$$

In order to give different levels of importance to each neighbor, we can use measures such as the *Adamic-Adar index* that sum over the inverse degrees of the common neighbors. Instead of using measures that focus only on the local neighbors of users, we can use *global overlap* measures that consider that two nodes may be of importance to each other even if they do not have many direct neighbors in common — for example, they belong to the

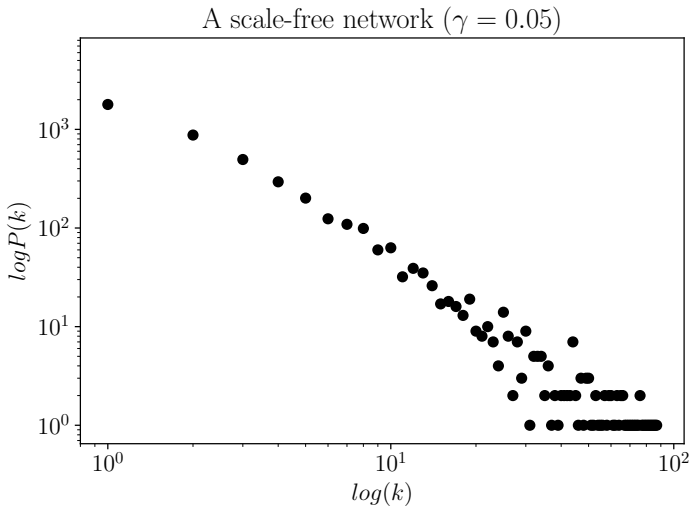


Fig. 2.2. – *The power-law distribution of a scale-free network (here, consisting of 10,000 nodes) is typically depicted on a log-log scale. This representation reveals a heavy tail, indicating that as the degree k increases, the probability for a node to possess such a degree decreases.*

same community. Given these heuristics, we can simply assume that the likelihood of the edge (i, j) is proportional to $\mathbf{S}_{ij}$, and set a threshold to determine whether to infer it or not.

2.3.2 Real-world properties of Online Social Networks

Generally, Online Social Networks (OSNs) are considered to have properties of real-world networks. This classification indicates that, when examining their specific measures as outlined above, we find that OSNs possess significant statistical properties that differentiate them from randomly generated networks (Mislove et al., 2007). In this section, we highlight some of the most significant of these properties.

The scale-free property. On OSNs we often notice an interesting phenomenon: while in principle, users have the option to follow everyone, only a select few accumulate a significant number of followers. These individuals may already be popular in real life (e.g., celebrities, politicians, etc) or they may have become viral through the platform's recommendation algorithm, which promotes their content. This is also called *preferential attachment*: users who are already popular

are more likely to receive new followers (Barabási and Albert, 1999). These types of networks are called *scale-free* due to having a degree distribution that follows a *power law*: the probability of observing a user with a degree k decreases rapidly as the value of k increases, formally:

$$P(k) \propto k^{-\gamma},$$

where γ is a parameter that determines the shape of the distribution. This leads to many small values having a very high probability, while a few very high values have a very low probability. The few user nodes with a high degree are called *hubs* and play a crucial role in the network for disseminating information. An example of a power-law distribution of a scale-free directed graph with 10,000 nodes is shown in Fig. 2.2. It is generated with the scale_free_graph generator function of the networkX Python package which returns a scale-free directed graph according to the definition given by Bollobás et al. (2003).

Presence of communities. In real life, humans naturally gravitate towards those who share similar traits or interests to them, a phenomenon known in sociology as *homophily* (McPherson et al., 2001). This tendency is also evident in OSNs: users tend to engage with content from individuals who resemble them in terms of interests, beliefs, or demographics (Musiał and Kazienko, 2013). This leads to the formation of *communities*, groups of users who share common traits and interact frequently with each other's content (Ganley and Lampe, 2009). From a SNA perspective, nodes within a community are more densely connected to each other than to nodes outside (Marin and Wellman, 2011). Communities can be identified using various clustering algorithms, such as the Louvain method (Blondel et al., 2008). It maximizes an important function known as *modularity*, that measures the density of edges inside the communities of a network compared to the edges between the communities (Newman and Girvan, Feb. 2004). On OSNs this modularity is positive, indicating a strong presence of community structure. This is also reflected in the high local clustering coefficient that is often observed (S. A. Myers et al., 2014).

Small-world property. The *small-world* property refers to the idea that a node can be reached from any other node by a relatively small number of steps, even in a large network (Barrat and Weigt, 2000). Many real-world networks possess this property, including OSNs (Poblete et al., 2011; Chao Yang et al., 2012; Sadri et al., 2017; Grandjean, 2016). During SNA, this is often reflected in the values of

the average shortest path and the diameter, both of which are relatively small. In sociology, this phenomenon is also known as *six degrees of separation*, suggesting that any two people in the world can be connected through a chain of six people on average (see Box. 2.3.2). In the context of OSNs, studies have shown that the average distance between users can be even smaller than four in some cases (S. A. Myers et al., 2014). Certainly, the average shortest path length depends on the system size but it does not vary significantly with it. More specifically, it has been found that the average shortest path length increases proportionally to the logarithm of the number of nodes in a network (Newman, 2000).

We Live In a Small World *Box 2.3.2*

The surprising phenomenon of six degrees of separation has a rich history that traces back to a 1929 short story by Hungarian author Frigyes Karinthy: a group of people conducts an experiment to prove that they can connect to any person in the world through a chain of at most five others. This narrative held a great influence on discussions around social networks, and eventually became the subject of actual scientific research. The most notable example is Stanley Milgram's *Small World Experiment*, which showed that any individual in the United States can be reached by a maximum of five connections on average (Milgram, 1967). Playwright John Guare further popularized the small world property, linking it to a term that he coined as *six degrees of separation*:

> *"I read somewhere that everybody on this planet is separated by only six other people. Six degrees of separation between us and everyone else on this planet. The President of the United States, a gondolier in Venice, just fill in the names. I find it extremely comforting that we're so close."*

In 1998, Watts and Strogatz found that the small-world property extends to other real-world networks, such as biological and electrical networks (Watts and Strogatz, 1998). These observations have been extended to OSNs as well: despite its immense size, Facebook users have an average distance of 4.74 between them (Backstrom et al., 2012), while on Twitter, this number is slightly smaller, between 3.78 and 4.37, depending on the country of the users (S. A. Myers et al., 2014).

2.3.3 Modeling the aspect of time

Often, online social media give rise to special cases of networks that can evolve and change over time. These are also known as *temporal graphs*, where the edges and nodes of the graph can be used to represent interactions or connections occurring at different time points (Kostakos, 2009; Holme and Saramäki, 2012). Both interaction networks and their underlying friendship networks can be treated as temporal graphs, and their analysis may require dynamic modeling approaches (Palla et al., 2007; T. P. Peixoto and Rosvall, 2017). However, this thesis deviates from the literature that considers OSNs as dynamically evolving structures. Instead, we treat both reposting and friendship networks as static graphs, disregarding the temporal evolution of the observed connections and interactions. To do so, we focus on a snapshot of the network structure within a relatively short time frame, during which connections can be considered unchanged. For example, on Twitter, a span of one week could be seen as a brief period during which friendship connections typically undergo minimal change (Gavilanes et al., 2013).

In the present thesis

Although we are treating networks as static, we are still interested in the aspect of time. Rather than dealing with distinct time IDs, which can complicate the modeling process, we will concentrate on the temporal ordering of the user-user interactions. This approach introduces the concept of time in a more intuitive manner, allowing us to capture temporal dynamics with greater simplicity.

2.3.4 Modeling the diffusion of information

The interaction networks that are usually provided by datasets show what is called the *diffusion* of a piece of information such as posts/tweets/retweets in the network (Gruhl et al., 2004; Bourigault, Lagnier, et al., 2014). This process of information diffusion from user to user via reposts/retweets is usually called a *cascade*, referring to the sequential fashion by which any "infectious" entity, such as information, influence, or a disease diffuses in time through the nodes of a network (Shannon, 1948; Granovetter, 1973; Anderson and May, 1991).

Modeling these cascades mathematically provides valuable simplifications and abstractions, when lacking full knowledge of the exact interactions underlying them. In epidemiology, for example, compartmental models are very effective in capturing the spread of an infectious disease between individuals, despite our incomplete understanding of reality (Kermack and McKendrick, 1927). In its simplest form, an epidemiological model assigns to each individual a state, based on the stage of the disease that they are currently in. It assumes that at any time, an individual can be found in one of the three following compartments or states:

- Susceptible ($\mathcal{S}$): Healthy individuals who have not yet been infected with the disease.
- Infectious ($\mathcal{I}$): Contagious individuals who have contracted the disease and can potentially infect others.
- Recovered ($\mathcal{R}$): Individuals who have been infected before, but have recovered from the disease and, therefore no longer infectious.

The modeling of some diseases may include additional states, like immune individuals, who cannot be infected, or unknown individuals, who have been exposed to the disease, but are not yet contagious (Hethcote, 2000).

Applying epidemiological models on interaction networks. Interestingly, in many cases, this simple yet effective model can be used with small variations to describe how information diffuses between individuals online (Saito et al., 2008). Accordingly, if we take the simplest SI approach, and assume a reposting cascade, a user can be found in one of the two following states at any point in time:

- Susceptible ($\mathcal{S}_p$): Users who have not yet reposted the post p.
- Infectious ($\mathcal{I}_p$): Users who have reposted p and therefore can transmit it to others (e.g., their followers).

All users in a network begin from the $\mathcal{S}_p$ state. Once they become Infectious with a specific post p, they are associated with a specific probability of transmitting it to others at any time step ahead. A user that is Infectious cannot change their state back. They can still, however, be Susceptible to other tweets, different than p, following the same logic. We will take advantage of this simple modeling

approach later, when formalizing our problem of network inference in more detail (Chapter 4).

2.4 Generating networks with random graph models

When confronted with the challenge of network inference, where the goal is to infer the structure of a real-world network from observed data, a randomly generated network might be used first. This type of synthetic network can be constructed using *random graph models* (Erdős and Rényi, 1959), serving as an initial or *prior* approximation of the underlying structure (Allahverdyan et al., 2010). Different random graph models can be used to generate networks with specific properties and characteristics, based on random processes. Of course, most of the random graph models that exist do not fully represent reality, having many properties that are different from those of real-world networks like OSNs (Daudin et al., 2008). However, these random graph models do provide insights into how any of these properties can influence the global behavior of networks, including real-world ones (Van Der Hofstad, 2013). Apart from their utility as priors during network inference, random networks have also been extensively used as benchmarks against which we can evaluate whether observed network characteristics are statistically significant or arise merely by chance (Watts and Strogatz, 1998). In this section, we present two of the most important random graph models that we can use to generate priors within a network inference framework.

2.4.1 The Erdős-Rényi model

The *Erdős–Rényi (ER) model* is a fundamental random graph model used to generate graphs with a specified number of nodes and edges (Erdős and Rényi, 1959). There are two main variations of this model, but we will focus on the $\mathcal{G}(N, p)$ model, also known as the Erdős-Rényi-Gilbert model (Gilbert, 1959). In this variation, N represents the number of nodes in the graph, and p is the probability of an edge existing between any two nodes. For each pair of nodes, an edge is added with probability p, independently of other edges. Assuming that a node cannot be connected to itself, we create an undirected graph as follows (Barabási, 2013):

1. Begin with N disconnected nodes.

2. Select two different nodes and generate a random number between 0 and 1. If the number exceeds p, connect them with an edge, otherwise leave them disconnected.

3. Repeat 2 for each of the $N(N-1)/2$ possible node pairs[2].

We can think of the creation of such a random graph as a *binomial experiment*, where the existence of an edge is a *Bernoulli random variable*. Therefore, the degree distribution of such a graph is binomial, and is described by the following probability mass function that gives the probability of a node j being connected to exactly k other nodes:

$$P(k) = \binom{N-1}{k} p^k (1-p)^{N-1-k}.$$

As we see, it is a product of three quantities: (i) the number of possible ways for selecting k nodes out of the $N-1$ potential ones; (ii) the probability that k nodes are indeed connected to j; and (iii) the probability that the remaining $N-1-k$ nodes are not connected to j.

Limitations. While the ER offers a principled and simple way to generate graphs, it does not always succeed in capturing many of the properties of real-world graphs. Although it does give graphs with a small diameter and average shortest path, which capture well the small-world property observed in OSNs, it falls short in capturing other important properties because of its following characteristics:

– **Binomial degree distribution and absence of hubs.** As the desired number of nodes N becomes larger, the distribution of an ER graph gets closer to a Poisson degree distribution with an exponential decay: the probability of encountering nodes with high degrees diminishes rapidly as the degree increases (Bollobás, 1998). An example of such distribution is shown in Fig. 2.3, on an ER network of $N = 10,000$ nodes[3]. This contrasts with the power-law distributions of OSNs that decay slowly, presenting heavy tails and hubs.

[2]Here, we assumed the simplest case of an undirected graph. In the case of a directed graph, we follow the same process but the number of possible node pairs will be twice the size, i.e., $N(N-1)$.

[3]Generated with the erdos_renyi_graph generator function of Python.

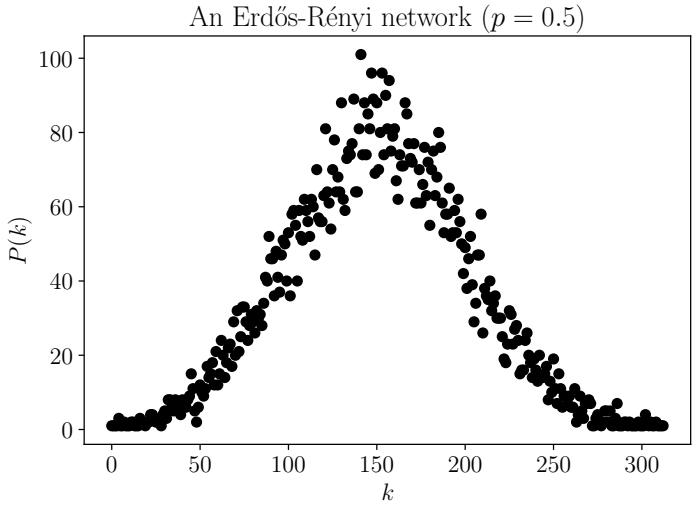


Fig. 2.3. – *The binomial degree distribution of an Erdős-Rényi network with 10,000 nodes.*

– **Low clustering coefficient.** The local clustering coefficients of an ER graph tend to be relatively small. They are independent of each node's degree and are inversely proportional to the number of nodes N. As a result, the graph exhibits a *homogeneous* structure with a uniform clustering coefficient across nodes. Instead, real-world OSNs display a more *heterogeneous* structure where the clustering coefficient is relatively higher and decreases with a node's degree, while remaining largely independent of the total size of the network N.

2.4.2 Stochastic Block Model

The *Stochastic Block Model (SBM)* is a generative model for random graphs first proposed in the 1980s by Holland et al. (1983). It offers a more nuanced approach to graph generation by incorporating community structure. Nodes are partitioned into distinct blocks or communities, with edges between nodes generated according to the probabilities that depend on their block memberships. For the standard SBM, we can generate a graph according to the following process:

1. Begin with N disconnected nodes.

2. Assign to each node i a unique community label $C_i \in [1, C]$ where C is the total number of communities present.

3. Select a pair of two different nodes (i, j) and connect them with probability p if $C_i = C_j$ (*intra-community* connections) and q otherwise (*inter-community* connections).

4. Repeat step 4 for each of the $N(N-1)/2$ possible node pairs.

To capture the presence of communities in real-world OSNs, we assume that nodes within the same community have a higher probability of being connected compared to nodes from different ones, i.e. $p > q$. This is also known as *assortativity* (Newman, 2002).

Limitations. While the traditional SBM can generate community structure with denser connections inside than outside, it is still incapable of handling the scale-free property of OSNs. This is due to its block creation process, which treats nodes within the same community equally, without taking the individuality of nodes into consideration (Karrer and Newman, 2011). As a result, it does not take into account node degree heterogeneity that is observed in real-world OSNs, where inside communities themselves we may notice nodes with higher degrees than average.

2.5 Measuring performance

Certainly, evaluating the efficacy of a network inference method in accurately deducing the underlying structure is a very crucial step. To explore this aspect experimentally, access to a comprehensive dataset containing both a set of interactions and their corresponding underlying structure, also referred to as the *ground truth*, is essential. With such a dataset at hand, at the end of a network inference algorithm, we can look into the edges for which the model has made inferences or predictions regarding their existence. Generally, a prediction that a model can make about an edge falls under two categories: *positive* and *negative*. In comparison to the actual state of an edge (the ground truth) the prediction can be classified as *true positive (TP)* if the model predicts the existence of an edge that exists in the ground truth, *false positive (FP)*, if the model incorrectly predicts the existence of an edge that does not exist in the ground truth, *true negative (TN)* if the model correctly predicts the absence of an edge that does not exist in the ground truth, and *false negative (FN)* if the model incorrectly predicts the absence of an edge that exists in the ground truth. Based on these, several metrics are used to assess the overall performance of the model:

– *Precision* measures the fraction of correctly predicted positive edges out of all cases predicted as positive by the model:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

– *Recall* measures the proportion of correctly predicted positive edges out of all actual positive edges in the ground truth:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

– *Accuracy* measures the proportion of correctly predicted edges (both positive and negative) out of all edges:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

– F_1 *score* is the harmonic mean of Precision and Recall, providing a balance between the two metrics:

$$\text{F}_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Usually, we cannot rely on these metrics in isolation. For example, we can achieve misleadingly perfect Recall ($= 1$) if we trivially infer every possible edge as positive. In this case, however, we will have many false positives, making the Precision score very low. In general, deciding on the exact metrics that we rely on for designing an algorithm is critical and requires consideration of different factors, such as:

□ *Is our data imbalanced?* In cases with imbalanced classes, where one class dominates the dataset, a model might achieve high Accuracy by simply predicting the majority class for all instances. For example, in the context of link prediction for OSNs, where users are very sparsely connected, the negative class (absence of edges) vastly outweighs the positive class (presence of edges). Consequently, a trivial approach assuming no edges exist could yield a seemingly high Accuracy metric. However, such a model would overlook the minority positive class, resulting in low performance for the other metrics.

□ *Do we prioritize false negatives or false positives?* Depending on the application, the significance of each prediction class may vary. For instance, in medical diagnosis, a false negative (missing a positive case) could have more severe consequences than a false positive. On the contrary, in the domain of OSNs, falsely identifying a non-existent connection between users (false positive) could lead to unnecessary recommendations by the social media platform and have a negative impact on the overall user experience. Hence, determining the priority between classes is vital during model evaluation, and depends on the specific application and its objectives.

In the context of OSNs, where our data is heavily imbalanced (many negative edges and few positives), two additional metrics can be useful: (i) the *Area Under the Receiver Operating Characteristic Curve (AUC-ROC)* which assesses the model's ability to distinguish between positive and negative edges across various thresholds; and (ii) the *Area Under the Precision-Recall Curve (AUC-PR)* which evaluates the trade-off between Precision and Recall across different threshold values.

2.6 Conclusion

In this chapter, we set the stage for further exploration into network inference given data from OSNs. We show the potential inadequacies of data (e.g., interaction networks) obtained from social media platforms and elaborate on a strategy to uncover the true underlying structure (e.g. friendship networks) which might be non-trivial to obtain. Additionally, we look into the distinctive properties of OSNs using methodologies from graph theory and SNA that can describe and verify their real-world structure. This is an important aspect to take into account during network inference, as the inferred structure should ideally mirror real-world properties. Therefore, in the experimental phase of this thesis, we will employ these metrics and methods to assess the effectiveness of inference methods in producing realistic structures. It is worth noting that while seemingly straightforward, many papers proposing inference methods overlook evaluating their inferred structures in terms of this aspect.

* * *

3 Network inference approaches for Online Social Networks

Contents

Network inference is a topic that became prevalent at the beginning of the 21st century, as a response to the increasing availability of data, which revealed a common theme: the information that is given directly from datasets is often insufficient or unreliable (Sprinzak et al., 2003). Consequently, network inference has found application in various domains, most notably social networks, and biology, as a way to infer hidden structures that cannot be directly observed experimentally (Goldberg and Roth, 2003; Butts, 2003). However, the process of inferring inaccessible network structures is challenging, demanding strategies capable of navigating noise, incomplete information, and data complexity in an efficient manner (Mukherjee and Speed, 2008). Given the variety of information sources and the unique challenges of different data domains, many different techniques have been proposed over the years. This chapter provides an overview of these works; our objective is to assess the suitability of existing approaches for

inferring networks in the context of OSNs, identifying potential weaknesses and areas of improvement.

3.1 Network inference with Bayesian models

One of the early formulations of network inference was given by Goldberg and Roth (2003). They addressed the task of inferring missing edges given unreliable biological networks and proposed a probabilistic framework that leverages statistical tools such as *Bayes' rule*. In the same year, but in a different domain, motivated by "the enduring problem of error in social network analysis", Butts (2003) proposed a family of *Bayesian models* which allows for the inference of posterior social structures in the presence of errors and missing data. As we quickly realize when overviewing the network inference literature, a significant portion of studies like the above opt for *Bayesian modeling* to infer hidden structures (Gelman, 2004). This approach incorporates *prior knowledge* or beliefs about different networks. In light of new observations, or data, it allows for the use of statistical tools that can update this prior knowledge and infer unobserved latent structures. To this day, Bayesian modeling is an important tool for network inference due to its flexibility and suitability to a wide range of data types and structures (Mukherjee and Speed, 2008). In this section, we present the fundamentals needed to understand this approach along with an overview of the works that have used it as a model of inference.

3.1.1 The Bayesian modeling approach

In this kind of approach, we assume that the observed data $X = \{x_1, x_2, \ldots, x_n\}$ has been generated from an unknown probability distribution with parameters θ that we want to estimate. This is captured by a data model expressed through a likelihood function $L(\theta|X) = P(X|\theta)$, which quantifies the probability of observing the current data given the estimated parameters. Typically, we possess some prior estimates or knowledge concerning the parameters and the distribution of the data. However, these beliefs are liable to change in light of new evidence. To update them, we can employ *Bayes' theorem*, which allows to obtain *posterior* beliefs, according to the following equation:

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}, \tag{3.1}$$

where:

– $P(\theta|X)$ is the posterior distribution of the parameters θ given the observed data X.

– $P(X|\theta)$ is the likelihood function $L(\theta|X)$, representing the probability of observing the data given the parameter values θ.

– $P(\theta)$ is the prior distribution, representing the initial beliefs or knowledge about the parameters before observing any data.

– $P(X)$ is the marginal likelihood representing the probability of observing the data under all possible parameter values.

In order to estimate the above parameters, we can use a classical statistical method known as *Maximum Likelihood Estimation (MLE)*, which can provide estimates for a model's parameters by maximizing the likelihood function (Aldrich, 1997). The primary objective is to find the parameter values $\hat{\theta}$ that maximize the likelihood function:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, L(\theta|X).$$

For independent and identically distributed (i.i.d.) observed data X, this likelihood function is often expressed as the product of the individual likelihoods:

$$L(\theta|X) = P(X|\theta) = \prod_{i=1}^{n} P(x_i|\theta).$$

In practice, it is often more convenient to work with the log-likelihood function:

$$\log L(\theta|X) = \log P(X|\theta) = \log \prod_{i=1}^{n} P(x_i|\theta) = \sum_{i=1}^{n} \log P(x_i|\theta).$$

Since the logarithm is a monotonic function, maximizing the log-likelihood function is equivalent to maximizing the likelihood function. Since the denominator of Bayes' theorem in Eq. 3.1 is independent of θ, and if we assume that the prior $P(\theta)$ is a uniform distribution, the task can also become equivalent to maximizing $P(\theta|X)$, also known as the *Maximum a posteriori estimate.*

As we have already mentioned, it is common to assume that a dataset has missing or unobserved values that should be taken into account when estimating the

parameters. Mathematically, this is expressed by assuming that the data involves some latent variables Z and the likelihood function will now become:

$$L(\theta|X, Z) = P(X, Z|\theta).$$

If we consider the MLE approach for parameter estimation we will get the marginal likelihood:

$$L(\theta|X) = \sum_{Z} P(X, Z|\theta).$$

Unfortunately, directly maximizing this expression or its logarithm is often computationally intractable due to the summation over all possible configurations of the latent variables (Beal and Ghahramani, 2006).

Dealing with the intractability of the inference. The *Expectation Maximization (EM) algorithm* is an iterative method that can provide a solution to the above challenge of maximizing the log-likelihood function in the presence of latent variables (Dawid and Skene, 1979). It was initially proposed by Dempster et al. (1977), as a way to leverage the relationship between the unobserved data and the latent variables of a model: If we were aware of the missing data the estimation of the model's variables would be straightforward. Similarly, if the model's variables were known, it would be possible to make unbiased estimations for the missing data. This mutual reliance between model variables and missing data brings forward an iterative approach, which alternates between two steps: first, estimate the missing data based on some assumed values for the variables, and then use these estimations to update the values of the variables (Nelwamondo et al., 2007). From an algorithmic perspective, it consists of the following steps:

1. **Initialization:** Initialize the parameters θ to some initial values.

2. **E-step**: compute the expected value of the log-likelihood function with respect to the conditional distribution of the latent variables given the observed data and the current estimate of the parameters. This involves calculating the following:

   $$Q(\theta, \theta_t) = \mathbb{E}_{Z|X,\theta_t}[\log p(X, Z|\theta)],$$

   that is the posterior distribution of the latent variables given the observed data and the current parameters.

3. **M-step**: update the parameter estimates by maximizing the expected log-likelihood $Q(\theta, \theta_t)$ obtained in the E-step with respect to parameters θ. This step typically involves solving an optimization problem that finds the parameter values that maximize the expected log-likelihood:

$$\theta^{t+1} = \underset{\theta}{\operatorname{argmax}} Q(\theta, \theta_t).$$

4. **Iterate updates:** Continue repeating steps 2 and 3 until the parameters' updates or the log-likelihood function change by an insignificantly small amount, indicating convergence.

In the present thesis

The EM algorithm provides a principled and effective approach to parameter estimation, in cases where direct optimization of the likelihood function is challenging. However, the EM algorithm cannot be easily adapted to NI settings with graph-structured data. In the next chapter, we will explore how the EM algorithm can be used to handle network data effectively.

3.1.2 Related works

The MLE approach for network inference is prevalent in a large body of work (Brugere et al., 2018). It generally comes together with a set of prior assumptions about the network underlying a set of observations (Mukherjee and Speed, 2008). The choice of the model representing these assumptions can be informed by our theoretical knowledge of the network's nature. For example, works like the ones by Redner (2008), Sales-Pardo et al. (2007) and Clauset et al. (2008) suggest that the intrinsic hierarchy in biological and social networks can be modeled into an MLE framework to reveal missing information in the data. Considering the prevalent community structure in many real-world networks (Girvan and Newman, 2002), the choice of the Stochastic Block Model (SBM) has been proposed as well (see Section 2.4.2). In this context, Abbe et al. (2015) provided an exact efficient MLE for inferring communities when using an SBM. Hayashi et al. (2016) also provided a tractable EM-like approach to infer the clusters that nodes belong to. However, these approaches usually focus more on accurately assigning nodes to clusters rather than addressing the unreliability directly on the level of edges.

More recent works have began explicitly estimating the levels of errors during an EM-type inference. For example, Le et al. (2018), consider a false discovery rate that can be estimated during a set of EM-like parameter updates that assume an SBM prior. Their work is motivated by the problem of neuroimaging applications where edges of the original brain networks are recorded with both false positives and false negatives. A more generalized approach in terms of parameterizing errors was given in the same year by Newman (2018): He proposes that the probability of observing an edge between two nodes can be parameterized by two quantities: the true-positive rate which is the probability of observing an edge where one truly exists, and the false-positive rate, which is the probability of observing an edge where none exists. He incorporates these into an EM-based inference framework which can be combined with different priors, such as the ER model, the SBM and others. The significance of his approach lies in its flexibility with regard to modeling the underlying structures and its ability to estimate different kinds of important data errors in a joint inference framework. This work will stand as an important benchmark for this thesis.

3.2 Recent approaches

In addition to the MLE approach discussed above, other methods have also been proposed, suggesting a range of techniques that differ in their modeling assumptions and complexity. Choosing the most appropriate one often depends on the specific characteristics of the data and the goals of the inference task. For instance, in a preliminary survey, Lü and Zhou (2011) divided the different network inference approaches into statistical estimation methods and heuristic methods, that rely on diverse similarity measures. A later survey by Brugere et al. (2018) categorized different network inference techniques based on their diverse applications in computational biology, climate science, neuroscience, epidemiology, ecology, and mobile networks. They further distinguished between *parametric* approaches, which utilize a probability distribution to model the edges in the network (like the MLE approach) and *non-parametric* ones, which directly assess the existence of each edge through statistical tests without using a model for the entire structure.

In this section, we categorize these methods based on the inference models they employ. Besides the MLE methods discussed above, we classify them into three additional categories: similarity-based, variational, and Monte Carlo Markov Chain (MCMC) based algorithms.

3.2.1 Similarity-based approaches

An early example of similarity-based approaches can be found in a paper by Newman (2001) who looked into collaboration networks and found that the probability of scientists collaborating (and therefore an edge existing between them) increases with the number of neighbors they have in common (namely, the Common Neighbours metric). In Liben-Nowell and Kleinberg (2003), the authors compare several node-to-node similarity metrics on large collaboration networks, and found that information about future interactions can be extracted from network topology alone. Likewise, Zhout et al. (2009), found that the simplest measure of Common Neighbours has the best overall performance for link prediction, and the Adamic-Adar index (Adamic and Adar, 2003) the second best. Kossinets and Watts (2006) have also analyzed different network statistics to investigate the effects of missing data on the properties of large-scale social networks.

However, the approaches based on this kind of metrics are considered quite inflexible, since they require a careful, hand-engineered choice of statistics which can neither be updated nor optimized (Hamilton, 2020). Nevertheless, they can still serve as a useful baseline to compare with more sophisticated methods (Hamilton, 2020).

3.2.2 Variational inference

Due to the intractability of the Bayesian approach in cases with many hidden variables, a different class of models has been proposed, called *variational* (Beal and Ghahramani, 2006). The main idea is to first construct a parameterized/-variational distribution over the latent variables of interest. The goal is to find the parameters of this distribution that bring it as close as possible to the true, posterior distribution of the latent variables (Jordan et al., 1999). For example, Beal et al. (2006) presented a variational Bayesian EM algorithm for directed graph models, which aims to optimize the divergence of the approximate to the true posterior distribution. Its main difference to the standard EM algorithm for MLE is that the maximization step computes a *distribution* over parameters, rather than their point estimates. Other methods have combined variational inference with an SBM model for the sampled data (Daudin et al., 2008; Latouche et al., 2012). More recent works are proposing in a variational context the consideration of missing data (Aicher et al., 2015; Tabouy et al., 2019) and errors (Priebe et al., 2015; Balach et al., 2017).

Although fast (Airoldi et al., 2008), variational approximators used may suffer from severe loss of posterior accuracy, since, they do not explore the parameter space as thoroughly (Bürkner et al., 2023). This is supported by studies showing that they provide fewer guarantees for correct inference than other approaches, like the Monte Carlo Markov Chain algorithm (Blei et al., 2017; Bürkner et al., 2023).

3.2.3 Monte Carlo Markov Chain algorithms

Another approach to inference is the use of Monte Carlo Markov Chain (MCMC) algorithms (Hoff et al., 2002; Kemp et al., 2004; Griffiths and Steyvers, 2004; Handcock et al., 2007). In these processes, instead of directly using a probability distribution, a sequence of data samples simulating a Markov chain is generated to approximate the target distribution (Hastings, 1970). Guimerà and Sales-Pardo (2009) applied this method to calculate the reliability of individual edges based on the groups to which their respective nodes belong. They used a Metropolis-Hastings algorithm (Hastings, 1970) for obtaining a sequence of random samples from a probability distribution from which direct sampling is difficult. More recently, Peixoto (2018) developed a similar approach, incorporating explicit error considerations, including true and false positive rates as motivated by Newman's work (2018). Peixoto's method utilizes a Metropolis-Hastings probability to accept each inference solution. Its main advantage is that it adopts the Stochastic Block Model (SBM) as the fundamental generative process, capable of inferring hierarchical community structure simultaneously.

Despite their attractive theoretical properties, MCMC methods face practical challenges when dealing with a large number of variables (Airoldi et al., 2008). They are considered significantly slow, making the estimation of complex models or their application to very large datasets practically not possible (Blei et al., 2017; Bürkner et al., 2023).

3.3 Time-aware network inference

Up to this point, we have not discussed the methods that incorporate the temporal information that is very commonly present in datasets coming from OSNs (see Section 2.3.3). Early works on network inference, along with the majority of those mentioned above, have largely overlooked this aspect. However, the increasing availability of data on human interactions motivated researchers to acknowledge the risks of neglecting the time dimension and started incorporating it more

rigorously (John Tang et al., 2009). The invention of OSNs provided a rich source of information for this task as well, providing many large-scale datasets with temporal interactions stemming from email (Leskovec, Adamic, et al., 2007), blog (Leskovec, McGlohon, et al., 2007), instant messaging communications (Onnela et al., 2007; Leskovec and Horvitz, 2007), and soon after social media (Dooms et al., 2013; Baumgartner et al., 2020; Naseem et al., 2021; Failla and Rossetti, 2024).

3.3.1 Temporal networks as cascades

In most works, it is assumed that the timestamped information observed in datasets has been triggered by specific events or activities with which users interact over short time scales (Kossinets et al., 2008). Notably, Kempe et al. (2000) drew an analogy between the well-studied field of epidemics — where diseases are triggering infection *cascades* (see Section 2.3.4) — and the study of temporal networks, where paths also emerge in a cascade-like manner. They define a *temporal network* as an undirected graph, where edges are labeled with timestamps that indicate when the connected nodes interacted. It is one of the first works that concentrated on inferring *time-respecting* paths, whose time labels abide by the ordering of time. However, at the time, we were missing the necessary theoretical tools to deal with the complexity of the problem, focusing instead only small-scale theoretical models. Shortly, a vast body of literature emerged, offering different models to capture the temporal dynamics of various real-world networks in a more efficient manner. In this section, we provide an overview of the different ways that time can be represented, most notably by assuming models of information diffusion.

3.3.2 Information diffusion models

Which are the models used for the specific case of social networks coming from OSNs? Network inference in this context is usually framed as a problem of recovering influence probabilities between users, given their interaction timestamps in the available cascades. If these interactions are incomplete, some type of *diffusion model* is chosen along with the learning method to formalize the principles that dictate the evolution of the observed cascades within a mathematical framework (Loossens et al., 2021).

Discrete Time Models

In some types of representations, the information diffuses in discrete time steps and the focus is on targeting the best set of users in a social network under a certain goal, such as influence maximization, or predicting the number of users being infected (Domingos and Matt Richardson, 2001; Matthew Richardson and Domingos, 2002). One of the earliest works approaching the problem in this way is by Kempe et al. (2003) who put forward two basic diffusion models, the *Independent Cascade Model* (a generalized SIR model, see section 2.3.4) and the *General Threshold Model*, on top of which we can build algorithms for selecting the most influential nodes in a network. In other, more pertinent to network inference tasks, the goal is predicting if and when an individual will be influenced or infected. Saito et al. (2008) proposed such an approach, assuming an Independent Cascade model of diffusion and employing an EM-based algorithm to infer influence probabilities between users. Here, an infected user is associated with a unique probability of infecting their graph neighbors, but if they do not succeed in the immediate next step, no attempts are made in subsequent steps. This is, of course, a less realistic representation of real-world information diffusion, where influence or information can propagate over multiple steps.

Continuous Time models

Instead of dealing with discrete-like structures, Goyal et al. (2010) proposed Continuous Time Models, where the influence probabilities are represented as continuous functions of time. They apply this as a way to predict the activation state of a user, and the time at which they are most likely to perform the action. In another example, Yang and Leskovec (2010) propose to work in continuous spaces where the temporal diffusion dynamics can be learned from the observations. They formulated this as a problem of predicting which node will infect which other node, developing a Linear Influence Model to estimate non-parametric influence functions of users in a least squares-like formulation. However, it has been shown that these methods suffer from expensive computation times (Goyal et al., 2010). Another work that is very relevant to the problem we are looking to solve, is the one by Rodriguez et al. (2011). Their model aims to infer the continuous temporal dynamics of the underlying diffusion network, modeling transmission happening at different rates across different edges, as is usually the case in real-world cases. However, this model does not explicitly account for

missing data or errors, and its evaluation is based on artificially created ground truth with assumptions not considered during inference.

Incorporating missing information

A common limitation that we notice among the above approaches is that they do not consider the inherent unreliability of data. In response, Wu et al. (2013) created an EM method that can tolerate missing observations in a diffusion process that follows the Continuous Independent Cascade model. Similarly, Lokhov (2016) introduced an approximate approach that reconstructs the parameters of an SI diffusion model, given only partial information on node activation times. Despite these efforts, there remains a general tendency for information diffusion models to be unable to handle both errors and missing data, often ignoring the concept of unreliability entirely. Additionally, many of these models make unrealistic and simplified assumptions about node-to-node interactions, such as restricting diffusion to a single time step, as seen in the commonly used Independent Cascade model.

Other terminologies *Box 3.3.2*

Network inference can also be traced in literature under different definitions, notably as a problem of (i) *network reconstruction* and (ii) *link prediction*. The problem of network reconstruction is framed as rebuilding a network from incomplete and erroneous data, as seen in some of the works that we already explored, by Guimerà and Sales-Pardo (2009) and others (Lokhov, 2016; T. Peixoto, 2018; Firestone et al., 2020; Young et al., 2021; J. Wu et al., 2022). In link prediction, on the other hand, one may have to consider two different types of edges that are missing and should be inferred: *missing* edges or *future* edges. According to Lü and Zou (2011), given that both represent previously unknown connections, distinguishing between the two types is challenging without additional context. Yet, in practice, the task of link prediction is more often associated with predicting edges that are likely to form in the future based on the existing structure of the network, whereas the task of inference is associated with inferring connections that had already been missing from the observed data.

3.4 Conclusion and limitations of existing works

In this chapter, we explored various techniques for inferring networks from unreliable data, also applicable to the domain of OSNs. Each method has its own advantages and disadvantages, which vary depending on the application and the size of the data. Notably, we observed a growing need for more unified methodologies that account for both errors and missing information while allowing for flexible, data-guided inference, as seen in the works by Newman (2018) and Peixoto (2018). Strengthening this argument, it has been recently advocated in a general perspective by Peel et al. (2022) that there is a lack of standardized approaches that address errors and incompleteness in network data. This is further challenged by the fact that prior unreliability estimations around the way that the data has been calculated prior to publication are often omitted (T. Peixoto, 2018). To ensure robustness and reproducibility, it is strongly suggested to interpret uncertain network data using statistical generative models, like the ER model and SBM, which can naturally handle uncertainties.

On top of that, as already mentioned in Chapter 1, assessing the efficacy of an inference method and determining the most suitable one for our specific dataset, still presents a significant challenge. This is further exacerbated by the empirical realization that the datasets we test these algorithms on rarely contain both the diffusion cascades *and* the underlying graph for comparison. As a result, for evaluation to be possible, one may have to artificially conceal a portion of the dataset and treat it as missing information. However, this approach may be problematic as it fails to consider the inherent unreliability of the data, which already contains errors and missing interactions. Towards a more reliable evaluation, two key points should be taken into consideration:

– **Are the inferred graphs feasible?** In cases where a reliable underlying graph is unavailable, we suggest a potentially valuable metric for the evaluation of our results (i.e., the inferred graph), that we call *feasibility*. Informally, an inferred graph that is *feasible* should explain all the interactions and their chronological order inside the dataset. The main intuition behind this is that a feasible graph may be closer to the real underlying graph that is unavailable. This concept may become more clear when considering the implications of non-feasibility in the context of OSNs: Suppose that in the dataset there is an interaction by a user (e.g., a repost) that cannot be explained by the inferred graph. Under our definition, this means that there

is no path (one or more hops away) in the inferred graph from the author of a post to the user who reposted it, or that the path is temporally not feasible. Then, either the latter user found this post from some other source (e.g., platform recommendation), or there is an error in the inference because the two users appear disconnected or connected in the wrong (temporally non-feasible) direction. It is trivial to confirm that current NI methods suffer from a lack of explicit consideration of feasibility during inference or evaluation — our goal is to verify this experimentally by looking closer into the results of different inference methods, eventually motivating the development of an inference process that can guarantee feasibility.

– **Are the inferred graphs realistic to the ground truth?** Newman (2018) highlights a key observation: inference algorithms lack the ability to specify the exact nature of the connections we are inferring. Whether it is friendships, professional ties, proximity, or another relationship, remains unclear from the algorithm's output. It merely serves as an estimate regarding the driving force behind the observed data. In our case, when inferring the connections underlying an interaction network from OSNs, we could probably get a combination of friendships, influence, shared opinions, and affiliations. Although time-consuming, it is more straightforward to collect the friendships between users when a platform's API allows it — hence, we are motivated to examine to what extent an inferred network can reveal the underlying friendship networks of the users.

Based on these points, our motivation is to investigate whether existing methods in the literature, when applied to real-world datasets, can infer graphs that are both feasible and close to the ground truth. To evaluate the latter aspect, we aim to collect the real-world friendship network that connects the users that retweeted the same set of tweets in a Twitter dataset, serving as the *ground truth graph*. In contrast, the feasibility metric does not require a ground truth for measurement; the dataset itself is sufficient. However, collecting a ground truth is important since it enables us to explore the relationship between feasibility and the connections that actually exist, while also closing the gap that exists in current network inference evaluation methods that do not have a ground truth to compare with. We hope that this investigation will provide valuable insights into the reliability of the feasibility metric for future applications, where such ground

truth cannot be easily obtained. The next chapters of this thesis will delve deeper into this exploration.

* * *

4

Constrained Expectation Maximization for feasible network inference

Contents

In this chapter, we introduce a novel approach to network inference, which we call *Constrained Expectation Maximization (CEM)*. It infers a posterior distribution of feasible underlying graphs that explain the provided dataset while respecting the chronological order of the interactions observed. Since the structure of the underlying network is not known, the definition of a prior that enforces a structure to the posterior inferred network is necessary. We will introduce two special cases of CEM. The first case assumes an Erdős–Rényi (ER) prior and is called CEM-ER. According to this prior, the underlying network that we are trying to infer has been created under a uniform probability ρ that is the same for all edges. However, this does not accurately reflect the structure of social media graphs, which, as we showed in Section 2.3.2 are less random and have some important properties, such as the existence of hubs. For that reason, we propose an additional case that incorporates a more realistic model for the underlying graph, the Stochastic Block Model (SBM). We call this extended method CEM-SBM.

4.1 Problem formulation

4.1.1 The input dataset: a reposting network

This thesis focuses on inferring networks from online social media datasets that may contain missing information. Since the most common data given by the platforms is traditionally the content of their users, our goal is to leverage this information in order to infer the friendship graphs between users that are usually hidden. More specifically, we will be looking into the *posts* and the *reposts* that the users generate given by their reposting networks. On X, for example, this corresponds to the tweets and retweets that users exchange, but we keep to a more general notation to consider other platforms as well.

The information that is provided. The input dataset that includes the posts and reposts is denoted by $\mathcal{D}$ and it includes P posts/reposts in total. For each instance in the dataset, we keep only four types of information: its unique post

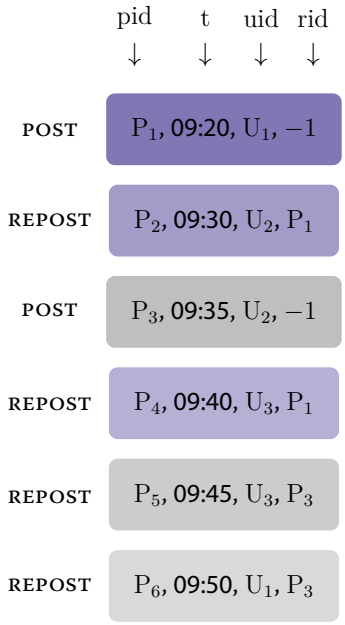


Fig. 4.1. – *Example of a reposting networks dataset. Each data entry has the form* $(\mathrm{pid}, \mathrm{t}, \mathrm{uid}, \mathrm{rid})$ *and reflects a post if* $\mathrm{rid} = -1$ *and a repost otherwise, where the value of* rid *gives the* pid *of the post that has been reposted.*

Tab. 4.1. Notations and definitions for the input dataset.

Notation	Definition
$\mathcal{D}$	Dataset of P posts and reposts of the type $(\mathrm{pid}, \mathrm{t}, \mathrm{uid}, \mathrm{rid})$.
$\mathcal{U}$	Set of users that are included in $\mathcal{D}$ ($\lvert\mathcal{U}\rvert = N$).
$\mathcal{D}_s \in \mathcal{D}$	Diffusion episode of post s.
author^s	The uid of the author of s.
$t^s(i) < t^s(j)$	User i reposted or posted s before user j.
$\mathbf{M}_{ij}$	Number of diffusion episodes for which it holds true that $t^s(i) < t^s(j)$.

id (pid), the time that the user posted it (t), the unique user id (uid), and the repost id (rid) that equals -1 if the post is original, or, if it is a repost, it is equal to some $\mathrm{pid} \in \mathcal{D}$ which points to the post instance in the dataset. If a user is the author of a pid we mark them as $\mathrm{author}_{\mathrm{pid}}$. The set that includes all the users that participate in the dataset is denoted by $\mathcal{U}$ and is of size $\lvert\mathcal{U}\rvert = N$. Figure 4.1 shows an example of a dataset $\mathcal{D}$ like the one described above. It includes $P = 6$ posts/reposts instances and $N = 3$ users in total. The first instance in $\mathcal{D}$ is an post with $\mathrm{pid} = \mathrm{P}_1$, and is posted at $\mathrm{t} =$ 09:20 by author U_1; the second instance with $\mathrm{pid} = \mathrm{P}_2$, tells us that user U_2 reposted at $\mathrm{t} =$ 09:30 the post with $\mathrm{pid} = \mathrm{P}_1$ (mapped to the author U_1), and so on.

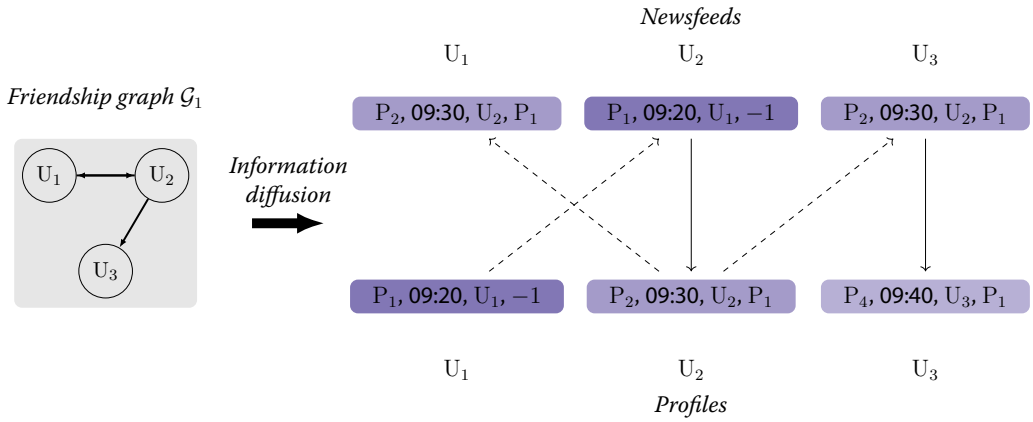


Fig. 4.2. – *The hidden way that a specific post* P_1 *authored by* U_1 *at* t_0 *=09:20 diffuses through the friendship network of users* $\mathcal{G}_1$*: At timestamp* t_0*, post* P_1 *appears on the Newsfeeds of* U_1*'s followers, in this case user* U_2*. At a later timestamp,* t_1 *=09:30,* U_2 *reposts* P_1 *on their Profile. Their repost takes the* $\mathrm{pid} = P_2$ *and appears on the Newsfeeds of* U_2*'s followers,* U_1 *and* U_3*. Finally,* U_3 *reposts it to their own Profile.*

4.1.2 Modeling the hidden way information diffuses

Generally, a social media platform provides a *Newsfeed* and a *Profile* for each user. The *Profile* includes their personal posts and reposts whereas the *Newsfeed* includes the posts and reposts created by their followees. *Newsfeeds* are formed based on the friendships in the network. Accordingly, Fig. 4.2 shows the possible *Newsfeeds* and *Profiles* of the users $\mathcal{U} = \{\mathrm{U}_1, \mathrm{U}_2, \mathrm{U}_3\}$ that created the dataset $\mathcal{D}$. As we see, the *Newsfeeds* are a result of the way that users are connected, i.e., their friendship graph $\mathcal{G}_1$, which is what we are seeking to infer. The *Profiles* are filled with individual posts from users and their interaction with *Newsfeeds*. If we assume that we have access to the unknown *Newsfeeds* and the corresponding friendship network of Fig. 4.2, we can infer directly that P_1 diffused from user U_1 to U_2 and then to U_3 (assuming that users only repost the users that they follow). Inferring this path was trivial since we assumed that we had access to the *Newsfeeds* which show the intermediary $\mathrm{pid}s$ of the reposts. However, until today, social media platforms keep other users' *Newsfeeds* private. Therefore, in the final dataset $\mathcal{D}$ this information is hidden. Instead, we only have access to the timestamps of the reposts of P_1 and the author it is mapped to (user U_1). For user U_2 it is trivial to infer that they reposted P_1 directly from U_1 (and thus follow them) since they are the first in the dataset to repost it. However, it is non-trivial to infer through whom U_3 reposted P_1; it could be through any of the users U_1 or U_2.

Towards the inference of hidden diffusion paths

Of course, the above example is quite simplistic; we can still come up with some trivial guesses about how the three users are connected that are not very far from the ground truth. In reality, though, we will have to deal with datasets that include millions of users, which makes our task much more challenging. Since social media datasets hide the *Newsfeeds* and the intermediary repost IDs, we do not know the real paths through which posts diffuse: a repost made by each user uid points only to the author of the initial post and not to the real user that uid reposted. Therefore, due to the dataset being only a partial view of each user's *Profile* and their interactions with their hidden *Newsfeed*, we cannot infer the friendship connections between the users directly on a large scale, and provides the motivation for a method that infers them indirectly through the intermediary diffusion paths.

4.1.3 Formulating network inference for Online Social Networks

We formulate the problem of NI for OSNs as follows: Given a dataset $\mathcal{D}$ of the activity of a set of users $\mathcal{U}$, we assume that there is an underlying friendship graph $\mathcal{G}$ connecting all users in $\mathcal{U}$ that is unknown and is what we are trying to infer. More formally, it is a directed friendship graph $\mathcal{G}$ where the nodes are the N users in $\mathcal{U}$ and each edge (i, j) translates to user j following user i. The graph $\mathcal{G}$ is represented by an adjacency matrix $\mathbf{A}$, of dimensions $N \times N$, where an element $\mathbf{A}_{ij}$ equals 1 if user j follows user i. The goal of NI is to infer the hidden adjacency matrix $\mathbf{A}$. Our intuition is that it is more likely that user j is following user i ($\mathbf{A}_{ij} = 1$) if a post reaches often user j through user i, through the edge (i, j). With this information not being directly available, we aim to infer it indirectly, via the intermediary diffusion paths that are hidden in the dataset. This will generate the unknown friendship graph $\mathcal{G}$ in question.

Users create diffusion episodes. Each original post with $\mathrm{rid} = -1$ in the dataset $\mathcal{D}$ is denoted by s and initiates what we call a *diffusion episode* of users reposting it sequentially in time, independently from other original posts. We say that the whole dataset $\mathcal{D}$ is a set of *diffusion episodes* $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_S\}$. Formally, we define each $\mathcal{D}_s \in \mathcal{D}$ as follows:

Definition 4.1.1 (Reposting diffusion episode): A reposting diffusion episode, or simply, a diffusion episode of a post s is defined as a set of timestamped user

posts and reposts $\mathcal{D}_s = (\text{author}^s, t_0^s) \cup (u, t^s) | u \in \mathcal{U} \bigwedge \exists(\text{pid}, t^s) : (\text{pid}, t^s, u, s) \in \mathcal{D}\}$, where author^s is the user that first posted s at timestamp t_0^s, and t^s is the timestamp for a user later reposting s. ⊠

In other words, each diffusion episode $\mathcal{D}_s \in \mathcal{D}$ includes the author of s, followed by the users who reposted it, in chronological order. To indicate that a user i appears in a diffusion episode $\mathcal{D}_s$ before j we use the notation $t^s(i) < t^s(j)$. Although a diffusion episode specifies who and when reposted a post in a network, it does not give the underlying mechanics of how the diffusion process is created.

An intuition. Out of all the diffusion episodes in $\mathcal{D}$, we count $\mathbf{M}_{ij}$ where it holds that $t^s(i) < t^s(j)$. If $\mathbf{M}_{ij} > 0$, it is probable that j has reposted an s from i. In this case, the pair is referred to as an *active pair*. Our intuition is that we become more certain about the existence of a diffusion path from i to j as $\mathbf{M}_{ij}$ becomes larger. As a result, $\mathbf{M}_{ij}$ is a quantity that can determine the hidden diffusion paths and we will use it extensively in the chapter that follows.

4.1.4 Assumptions on the diffusion of posts

To infer the hidden diffusion paths that resulted in a diffusion episode, we first need to decide on a diffusion model. In this case, we opt for a simple model, the SI diffusion model, which has been extensively used in epidemiological models (Daley and Gani, 1999) and apply it to social media users: when a new post s arrives on a user's *Newsfeed*, they become Susceptible. If they repost it they become Infected with the specific post s and remain so for the rest of the diffusion. Most existing works using the SI model consider that infection can happen only one time step ahead, after a user becomes Susceptible. We assume, however, that when a user posts something, they can diffuse it to their still uninfected followers (those in the Susceptible state) during any consecutive timestamp. However, given that at the time of the inference we are not aware of the exact followers of users, we need to make some additional assumptions:

- *Assumption 1:* The author of every post s that has been reposted is included in the dataset $\mathcal{D}$.
- *Assumption 2:* Users repost only from the users they follow, i.e., their set of followees. We assume that the latter are always present in $\mathcal{D}$.
- *Assumption 3:* A post can diffuse from user i to user j only if user i has reposted s chronologically earlier in the diffusion episode $\mathcal{D}_s$ than user j.

Although the second assumption does not always hold in practice, it simplifies our task. As we will see later, our approach can be expanded accordingly to take into account instances in which someone reposts content from users who are not inside the dataset or even from users outside their list of followees (e.g., when Twitter users repost something from the trending hashtags or via the search function, etc). We should also note that we can only obtain friendships between users who have interacted with one another at least once in the available dataset $\mathcal{D}$.

4.2 Definition of feasibility

A probabilistic approach to inference. For every diffusion episode $\mathcal{D}_s$ in the dataset $\mathcal{D}$ and every user i that reposted s at a timestamp before j, we first define the binary variable $X_{ij}(s) \in \{0, 1\}$ that is equal to 1 if the post s passed from i to j (i.e., j follows i) and 0 otherwise. As underlined in the previous section, the real value of $X_{ij}(s)$ is unknown. Therefore, given the chronological order of reposts in $\mathcal{D}_s$, we may imagine many feasible routes through which the post s might have spread to those who reposted it. These paths create a propagation graph $\mathcal{G}_s = \{\mathcal{V}_s, \mathcal{E}_s\}$ per diffusion episode, with the users in each diffusion episode $\mathcal{D}_s$ as nodes $\mathcal{V}_s$ and the edges set $\mathcal{D}_s$ containing the (unknown) edges that we have to infer for the given post. Every edge follows the propagation's direction; for instance, if an edge (i, j) is inferred in $\mathcal{G}_s$ it means that $X_{ij}(s) = 1$.

Given the above and the motivations for the concept of feasibility given in the Introduction, for each diffusion episode $\mathcal{D}_s$, our goal is not only to infer an underlying propagation graph $\mathcal{G}_s$, but also one that is feasible. Formally, we wish to infer a directed acyclic graph (DAG) $\mathcal{G}_s$ that is *feasible* and explains the whole $\mathcal{D}_s$ sequence, according to the following definition:

Definition 4.2.1 (Feasible propagation DAG $\mathcal{G}_s$ per diffusion episode $\mathcal{D}_s$): Given a diffusion episode $\mathcal{D}_s$ from $\mathcal{D}$, we say that a propagation DAG $\mathcal{G}_s$ is feasible, or, equivalently, that it explains $\mathcal{D}_s$, if (i) there exists (at least) one directed path from the author author^s to every other user $j \in \mathcal{D}_s \backslash \mathrm{author}^s$ and (ii), for each edge (i, j) of the path it holds that $t^s(i) < t^s(j)$, i.e., all of its edges follow the time-ordering of the reposts. ⊠

If we take the union of every feasible propagation graph $\mathcal{G}_s$ inferred per diffusion episode $\mathcal{D}_s$, we get the full friendship graph $\mathcal{G}$ and we can build its adjacency

matrix $\mathbf{A}$ as follows: we set $\mathbf{A}_{ij} = 1$ if there exists at least one $\mathcal{G}_s$ where the edge (i, j) exists, and $\mathbf{A}_{ij} = 0$ otherwise.

Definition 4.2.2 (Feasible friendship graph $\mathcal{G}$): An inferred graph $\mathcal{G}$ is called feasible, if, for every diffusion episode $\mathcal{D}_s \in \mathcal{D}$, there exists a subgraph which is a feasible propagation DAG $\mathcal{G}_s$ as we defined it above[1]. ⊠

Why enforcing feasibility makes sense?

Enforcing feasibility makes sense in OSNs because information (e.g., tweets) is propagated in a diffusion-like manner through the underlying friendship network that we are trying to infer. If we assume that users rarely repost information outside the users they follow, enforcing the feasibility constraint could help the inference process generate the true diffusion network more accurately. Alternatively, as we will show later, we can lower the percentage of feasibility that we expect from the inference, if we assume that the source of information in the dataset comes outside of the users' friendship connections (e.g. guided by the platform's recommendation algorithm or self-guided search).

4.3 Enforcing feasibility with a set of feasibility constraints

The main challenge of network inference in OSNs arises from the fact that the binary value $X_{ij}(s)$ defined for the different user pairs is unknown. However, we can restrict the number of solutions by imposing a set of feasibility constraints on all the values. These constraints should ensure that all the diffusion episodes in the dataset are feasible given the inferred network according to Definition 1. Specifically, they should guarantee that if a user j appears in a diffusion episode $\mathcal{D}_s$ they should be connected with at least one user i that appears in $\mathcal{D}_s$ before them, including the author of s (i.e., it should hold that $t^s(i) < t^s(j)$). As a result, the constraints have the following format:

$$\sum_{i \in \mathcal{D}_s \text{ s.t. } t^s(i) < t^s(j)} X_{ij}(s) \geq 1, \forall j \in \mathcal{D}_s \backslash \{\text{author}^s\}, \tag{4.1}$$

$$X_{ij}(s) \in \{0, 1\},\ \forall i, j \in \mathcal{U},\ \forall \mathcal{D}_s \in \mathcal{D}. \tag{4.2}$$

[1]Keep in mind that the full graph $\mathcal{G}$ is not a DAG.

We can again understand this expression by the simple example of Fig. 4.3, which shows the constraints on $X_{ij}(s)$, given the set of diffusion episodes $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2\}$. The role of these constraints is to guide the process toward solutions that belong to the feasible group of graphs. To do so, the constraints should be defined for each diffusion episode $\mathcal{D}_s \in \mathcal{D}$, and each user that reposted s, according to Eq. 4.1. For example, as we see in Fig. 4.3, given the first constraint for diffusion episode $\mathcal{D}_1$, we can derive easily that the user U_2 reposted post $s = 1$ directly from its author U_1 ($X_{12}(1) = 1$). The second constraint tells us that user U_3 has reposted $s = 1$ either from U_1, or from U_2 (or, from both). If we look closer, the possible graphs that we marked as non-feasible earlier violate these constraints. For example, $\mathcal{G}_A$ violates the first constraint for $\mathcal{D}_1$, since $X_{12}(1) = 0$. Likewise, $\mathcal{G}_B$ violates the last constraint for $\mathcal{D}_1$, since $X_{21}(2) + X_{31}(2) = 0$. As we saw in the figure and the equations above, the X_{ij} value of a pair (i, j) is different for each diffusion episode that it appears in. For example, X_{23} appeared two times, one time for $\mathcal{D}_1$ and another one for $\mathcal{D}_2$.

Overcoming the intractability of the solution. With all the possible combinations that each X_{ij} value can take for all active pairs and diffusion episodes observed, we soon realize that the problem is intractable when dealing with large datasets. The only direct knowledge we have for each pair is the constant value $\mathbf{M}_{ij}$, i.e., the total number of times that a user i appears before j in every diffusion episode $\mathcal{D}_s \in \mathcal{D}$. What we are interested in, is the number of times that a post diffused through the edge (i, j), out of the $\mathbf{M}_{ij}$ times that it could be possible. We model this with the unknown quantity Y_{ij} which is equal to the total number of times that j reposts from i. More formally:

$$Y_{ij} = \sum_{\mathcal{D}_s \in \mathcal{D},\ \text{s.t. } t^s(i) < t^s(j)} X_{ij}(s). \tag{4.3}$$

As we can see above, to find Y_{ij}, we sum over all diffusion episodes where it holds that $t^s(i) < t^s(j)$. This happens $\mathbf{M}_{ij}$ times in total. In Table 4.2, we summarize all the different parameters defined for the edges along with whether this information is directly available from the dataset $\mathcal{D}$ (column "Hidden").

4.4 Defining probabilities of diffusion

To solve the above problem we make the following important assumption: for every active pair (i, j) in any diffusion episode $\mathcal{D}_s \in \mathcal{D}$, a user j reposts an s from i independently from other diffusion episodes with an unknown diffusion

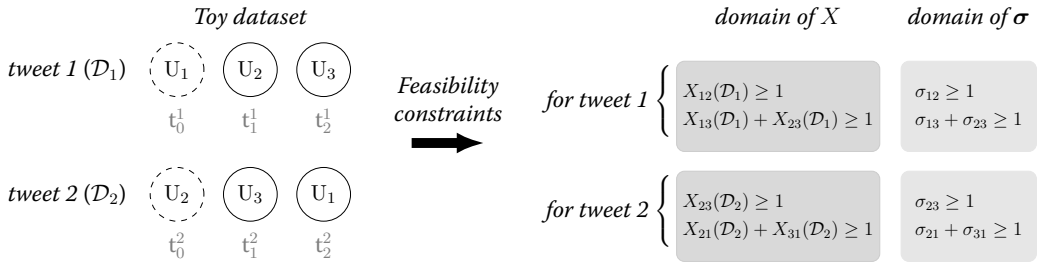


Fig. 4.3. – *Constructing the feasibility constraints on the domain of X and σ on the toy dataset*

Tab. 4.2. Edge parameters*

Parameter	Type	Range	Description	Hidden
$\mathbf{A}_{ij}$	(i,j) instance of adjacency matrix $\mathbf{A}$	$\{0,1\}$	1 if j follows i in the hidden $\mathbf{A}$	✓
$\mathbf{M}_{ij}$	Constant, counted directly from the data	$\mathbb{N}_0$	#times that i appears before j, in any $\mathcal{D}_s \in \mathcal{D}$	✗
$X_{ij}(s)$	Independent Bernoulli r.v. with mean σ_{ij}	$\{0,1\}$	1 if j reposted s directly from i	✓
Y_{ij}	Independent Binomial r.v. with mean $\mathbf{M}_{ij}\sigma_{ij}$	$\mathbb{N}_0$	#times that j reposts directly from i, any s	✓
σ_{ij}	Mean of $X_{ij}(s)$	$[0,1]$	Probability for j reposting directly from i, any s	✓

** r.v.: random variable*

probability $\sigma_{ij} \in [0,1]$. Therefore, $X_{ij}(s)$ is an independent Bernoulli random variable with a mean parameter σ_{ij} which does not depend on s. In other words, the diffusion probability σ_{ij} of a user pair is the same across all diffusion episodes, which means that there is no preference in terms of content when someone

chooses to repost. Of course, this does not accurately reflect reality but it serves as a useful simplification. Therefore, for a user pair (i, j):

$$\sigma_{ij} = \mathbb{E}\left[X_{ij}(s)\right]. \tag{4.4}$$

We can now transfer our problem from searching over the binary domain of $X_{ij}(s)$ to solving over the real domain of the σ_{ij} values. By taking the expectation in (4.1) and given Eq. 4.4, we get the following set of constraints:

$$\sum_{i \in \mathcal{D}_s \text{ s.t. } t^s(i) < t^s(j)} \sigma_{ij} \geq 1, \forall j \in \mathcal{D}_s \backslash \{\text{author}^s\}, \tag{4.5}$$

$$\sigma_{ij} \in [0, 1], \ \forall i, j \in \mathcal{U}. \tag{4.6}$$

From Eq. 4.3 and Eq. 4.4, Y_{ij} is the sum of $\mathbf{M}_{ij}$ independent Bernoulli random variables that have a mean value σ_{ij}. In other words, Y_{ij} is an independent Binomial random variable with mean value $\mathbf{M}_{ij}\sigma_{ij}$:

$$\mathbb{E}[Y_{ij}] = \sum_{\mathcal{D}_s \in \mathcal{D}, \text{ s.t. } t^s(i) < t^s(j)} \mathbb{E}[X_{ij}(s)] = \sum_{\mathcal{D}_s \in \mathcal{D}, \text{ s.t. } t^s(i) < t^s(j)} \sigma_{ij} = \mathbf{M}_{ij}\sigma_{ij}. \tag{4.7}$$

4.5 Problem modeling and learning method

4.5.1 Erdős–Rényi prior (CEM-ER)

As mentioned above, the prior structure of the network $\mathbf{A}$ is not known, and therefore a uniform prior ρ is assumed for all edges. Hence, the prior takes the form of a probability distribution $P(\mathbf{A} \mid \theta)$, where θ is a set of hidden parameters that give us more details on the underlying network. Given a dataset $\mathcal{D}$ of posts and reposts, $P(\mathbf{A}, \theta \mid \mathcal{D})$ is the probability that the inferred network is $\mathbf{A}$ and the parameters get the value θ. The parameters θ should account for a wider range of potential network types and data generation methods. Therefore, they are chosen as follows:

- The probability that a user j shares content through a user i, represented by the set of σ_{ij} values that we presented in Section 4.3.

- To model the uncertainty about the structure of the graph's adjacency matrix $\mathbf{A}$, we assumed that there is a prior probability ρ of an edge drawn independently between any two nodes i, j (Erdős–Rényi prior).

– The *true positive utilization rate* α: the probability of a post propagating through an edge that we inferred to exist in the underlying network $\mathcal{G}$. Given the (hidden) number of interactions between users Y_{ij}, we consider that when an edge exists in $\mathcal{G}$ ($\mathbf{A}_{ij} = 1$) the Y_{ij} out of the $\mathbf{M}_{ij}$ experiments are successful (we get Y_{ij} true positive edges in total), each with probability α.

– The *false positive utilization rate* β: the probability of inferring that a post propagated through edges that do not exist in $\mathcal{G}$. Likewise to above, when $\mathbf{A}_{ij} = 0$, we consider that the Y_{ij} out of $\mathbf{M}_{ij}$ experiments are successful (we get Y_{ij} false positive edges), each with probability β.

We can see that the global parameters α and β depend on whether an edge exists in the ground truth network $\mathcal{G}$. To find the most probable value of the parameters θ given the observed data and infer $\mathbf{A}$ with maximum likelihood, we will employ an Expectation–Maximization (EM) algorithm which is a standard inference tool when some data is unknown or hidden. As suggested by its name, an EM iteration involves two consecutive steps: an expectation (E) step, which computes the expected log-likelihood under the most recent estimation of the parameters in θ; then, a maximization (M) step, which determines the parameters that maximize the expectation. Then, the computed parameters are used in the following iteration, and so on, until we satisfy a convergence criterion.

We start constructing the EM iterations, following the method proposed by Newman (2018) and employ the Bayes' theorem:

$$P(\mathbf{A}, \theta \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \mathbf{A}, \theta) P(\mathbf{A} \mid \theta) P(\theta)}{P(\mathcal{D})}. \tag{4.8}$$

The probability that we get the specific set of posts and reposts $\mathcal{D}$ given $\mathbf{A}$ and the parameters $\theta = \{\alpha, \beta, \rho, \boldsymbol{\sigma}\}$, found in the numerator of the above expression, will differ here from Newman since we have introduced the hidden number of interactions between users, Y_{ij}. Given the ordered nodes of a diffusion episode, each repost path is chosen independently per diffusion episode. We also assumed as prior knowledge that between any two nodes in $\mathbf{A}$ an edge has been drawn with probability ρ. Therefore, if we consider that the data $\mathcal{D}$ is composed of the

known constant values $\mathbf{M}_{ij}$ and the unknown binomial random values Y_{ij}, then $\mathcal{D} = (\mathbf{M}, Y)$ and we get:

$$P(\mathcal{D} \mid \mathbf{A}, \theta)P(\mathbf{A} \mid \theta) = \prod_{i \neq j} \left[\alpha^{Y_{ij}}(1-\alpha)^{\mathbf{M}_{ij}-Y_{ij}}\rho\right]^{\mathbf{A}_{ij}} \left[\beta^{Y_{ij}}(1-\beta)^{\mathbf{M}_{ij}-Y_{ij}}(1-\rho)\right]^{1-\mathbf{A}_{ij}}. \tag{4.9}$$

What this model tells us is that when $\mathbf{A}_{ij} = 1$, the Y_{ij} out of the $\mathbf{M}_{ij}$ experiments are successful, each with probability α. When $\mathbf{A}_{ij} = 0$, the Y_{ij} out of $\mathbf{M}_{ij}$ experiments are successful, each with probability β. This gives a simple way to describe the different scenarios of users interacting with each other depending on whether they are following each other or not in the underlying network. For the whole set of parameters θ, we assume a uniform prior probability $P(\theta)$. If we sum in (4.8) over all possible networks $\mathbf{A}$, we find that $P(\theta \mid \mathcal{D}) = \sum_{\mathbf{A}} P(\mathbf{A}, \theta \mid \mathcal{D})$. Then, as suggested by Newman (2018), we can apply the well-known Jensen's inequality on the $\log$ of $P(\theta \mid \mathcal{D})$:

$$\log P(\theta \mid \mathcal{D}) = \log \sum_{\mathbf{A}} P(\mathbf{A}, \theta \mid \mathcal{D}) \geq \sum_{\mathbf{A}} q(\mathbf{A}) \log \frac{P(\mathbf{A}, \theta \mid \mathcal{D})}{q(\mathbf{A})}, \tag{4.10}$$

where $q(\mathbf{A})$ is any probability distribution over networks $\mathbf{A}$ satisfying $\sum_{\mathbf{A}} q(\mathbf{A}) = 1$. We also define the posterior probability of an edge existing between i and j by $\mathbf{Q}_{ij} = P(\mathbf{A}_{ij} = 1 | \mathcal{D}, \theta) = \sum_{\mathbf{A}} q(\mathbf{A})\mathbf{A}_{ij}$. If we take the expectation of Eq. (4.10) we find that:

$$\mathbb{E}[\log P(\theta \mid \mathcal{D})] \geq \sum_{\mathbf{A}} q(\mathbf{A}) \log \frac{D_{ij}}{q(\mathbf{A})}, \tag{4.11}$$

where[2],

$$D_{ij} = \Gamma \prod_{i \neq j} \left[\rho\alpha^{\mathbf{M}_{ij}\sigma_{ij}}(1-\alpha)^{\mathbf{M}_{ij}(1-\sigma_{ij})}\right]^{\mathbf{A}_{ij}} \left[(1-\rho)\beta^{\mathbf{M}_{ij}\sigma_{ij}}(1-\beta)^{\mathbf{M}_{ij}(1-\sigma_{ij})}\right]^{1-\mathbf{A}_{ij}}. \tag{4.12}$$

We find that the choice of q that achieves equality of (4.11) and hence, maximizes the right-hand side with respect to q is:

$$q(\mathbf{A}) = \prod_{i \neq j} \mathbf{Q}_{ij}^{\mathbf{A}_{ij}}(1-\mathbf{Q}_{ij})^{1-\mathbf{A}_{ij}}, \tag{4.13}$$

[2]The full derivations can be found in Appendix A.

where, $\mathbf{Q}_{ij}$ is the posterior probability that the edge (i, j) exists, and we find that it equals:

$$\mathbf{Q}_{ij} = \frac{\rho\alpha^{\mathbf{M}_{ij}\sigma_{ij}}(1-\alpha)^{\mathbf{M}_{ij}(1-\sigma_{ij})}}{\rho\alpha^{\mathbf{M}_{ij}\sigma_{ij}}(1-\alpha)^{\mathbf{M}_{ij}(1-\sigma_{ij})} + (1-\rho)\beta^{\mathbf{M}_{ij}\sigma_{ij}}(1-\beta)^{\mathbf{M}_{ij}(1-\sigma_{ij})}}. \tag{4.14}$$

The details of the above derivation are shown in Appendix A. To find the maximizing posterior distribution $q(\mathbf{A})$ it suffices to find the individual maximizing posterior probabilities $\mathbf{Q}_{ij}$ according to Eq. (4.14). Given these values, if we further maximize with respect to the parameters $\theta = \{\alpha, \beta, \rho, \boldsymbol{\sigma}\}$ we can get the maximum-likelihood value we seek. The updates for the first three parameters are thus calculated to be the following:

$$\alpha = \frac{\sum_{i\neq j}\mathbf{M}_{ij}\sigma_{ij}\mathbf{Q}_{ij}}{\sum_{i\neq j}\mathbf{M}_{ij}\mathbf{Q}_{ij}},\ \beta = \frac{\sum_{i\neq j}\mathbf{M}_{ij}\sigma_{ij}(1-\mathbf{Q}_{ij})}{\sum_{i\neq j}\mathbf{M}_{ij}(1-\mathbf{Q}_{ij})}, \tag{4.15}$$

$$\rho = \frac{1}{N(N-1)}\sum_{i\neq j}\mathbf{Q}_{ij}, \tag{4.16}$$

where N is the number of users in the dataset. Finally, to find the whole vector $\boldsymbol{\sigma}$ that includes all the σ_{ij} unknown diffusion parameters, we must solve a linear optimization problem as follows (for derivation refer to Appendix A):

$$\max_{\sigma}\sum_{i\neq j}\sigma_{ij}(\mathbf{W}_{ij} - \lambda W) \tag{4.17}$$

$$\text{s.t. } \boldsymbol{\sigma} \in F_{\boldsymbol{\sigma}},$$

$$\text{where } \mathbf{W}_{ij} = \mathbf{M}_{ij}\left(\mathbf{Q}_{ij}\log\frac{\alpha}{1-\alpha} + (1-\mathbf{Q}_{ij})\log\frac{\beta}{1-\beta}\right),$$

$$\lambda > 0 \text{ some given penalty for regularisation, and } W = \max_{i\neq j}\mathbf{W}_{ij}.$$

We added the value λ into the optimization objective as a penalty per iteration, since our initial goal is to infer a network that is feasible with the minimum possible number of edges. Without it, all (i, j) pairs with $\mathbf{W}_{ij} > 0$ would immediately get their $\sigma_{ij} = 1$, leading to the inference of more edges than we initially wanted. As λ moves closer to 1, it forces the optimization goal to be negative and thus, to be guided only by the provided constraints. It is equivalent to penalizing the total expected number of inferred edges. As λ approaches 0, the optimization infers the largest number of edges possible. We will explore in detail the effect

of the hyperparameter λ with values that vary from 0 to 1 in the Experiments section. The final CEM-ER algorithm is shown in Algorithm 1.

4.5.2 Stochastic block model prior (CEM-SBM)

Since we are working with social media data, where there is usually a strong presence of communities, we believe it is more realistic to assume that the network is derived from an SBM, a generative model of community structure that was first proposed by Holland et al. (1983). In the standard SBM, each node i participates in a different block (community) which we indicate by $\mathbf{g}_i$, which may take values in $[1, G]$ where G is the number of hidden communities. The number of edges between nodes i and j follows a Bernoulli distribution with mean $\omega_{\mathbf{g}_i,\mathbf{g}_j}$, that is the relative probability of intra-community (if $\mathbf{g}_i = \mathbf{g}_j$) or inter-community (if $\mathbf{g}_i \neq \mathbf{g}_j$) connection.

As we can see, in the case of CEM-ER, the prior structure of the network $\mathbf{A}$ was the only kind of unobserved data, but in this case, we have two unknowns: the network $\mathbf{A}$ and the vector of the community assignments of the users $\mathbf{g}$. Hence, the prior takes the form of a probability distribution $P(\mathbf{A}, \mathbf{g} \mid \theta)$, where θ denotes the unknown parameters of the distribution, which gives additionally the details of the community structure. This approach, therefore, allows us to infer both the unknown network structure and the community structure simultaneously. Given a dataset $\mathcal{D}$, $P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{D})$ is the probability that we get $\mathbf{A}$, the users' community participation vector $\mathbf{g}$ and a set of chosen parameters θ. The parameters set θ that we select here includes two newly added parameters that replace the prior ρ that we had in the CEM-ER case:

- Following the SBM for $\mathbf{A}$ and the users' community participation vector $\mathbf{g}$, we suppose that there is a prior probability p of an edge existing between any two nodes i, j that belong in the same community, i.e., $\mathbf{g}_i = \mathbf{g}_j$.

- The nodes that belong in different communities are connected with a probability q.

We construct the EM iterations as we did before, following Bayes' theorem:

$$P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \mathbf{A}, \mathbf{g}, \theta) P(\mathbf{A}, \mathbf{g} \mid \theta) P(\theta)}{P(\mathcal{D})}. \tag{4.18}$$

Taking into consideration the definition of the parameters above, the probability that we get the specific dataset $\mathcal{D}$, given $\mathbf{A}$, $\mathbf{g}$ and $\theta = \{\alpha, \beta, p, q, \boldsymbol{\sigma}\}$ is driven by the probabilities α and β, whereas the probability that we get $\mathbf{A}$ and $\mathbf{g}$ given θ depends on the probabilities p and q. Therefore, assuming that each user reposts independently from others:

$$P(\mathcal{D} \mid \mathbf{A}, \mathbf{g}, \theta) P(\mathbf{A}, \mathbf{g} \mid \theta) = \prod_{\substack{i \neq j \\ \mathbf{g}_i = gj}} \left[\alpha^{Y_{ij}} (1-\alpha)^{\mathbf{M}_{ij} - Y_{ij}} p\right]^{\mathbf{A}_{ij}}$$

$$\left[\beta^{Y_{ij}} (1-\beta)^{\mathbf{M}_{ij} - Y_{ij}} (1-p)\right]^{1-\mathbf{A}_{ij}} \prod_{\substack{i \neq j \\ \mathbf{g}_i \neq gj}} \left[\alpha^{Y_{ij}} (1-\alpha)^{\mathbf{M}_{ij} - Y_{ij}} q\right]^{\mathbf{A}_{ij}}$$

$$\left[\beta^{Y_{ij}} (1-\beta)^{\mathbf{M}_{ij} - Y_{ij}} (1-q)\right]^{1-\mathbf{A}_{ij}}. \tag{4.19}$$

For the whole set of parameters θ, we assume again a uniform prior probability $P(\theta)$. We sum (4.18) over all possible networks $\mathbf{A}$ and we find that $P(\theta \mid \mathcal{D}) = \sum_{\mathbf{A}} P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{D})$. Then, we can apply the well-known Jensen's inequality on the log of $P(\theta \mid \mathcal{D})$:

$$\log P(\theta \mid \mathcal{D}) = \log \sum_{\mathbf{A}} P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{D}) \geq \sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \log \frac{P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{D})}{q(\mathbf{A}, \mathbf{g})}, \tag{4.20}$$

where $q(\mathbf{A}, \mathbf{g})$ is any joint probability distribution over networks $\mathbf{A}$ and community assignments $\mathbf{g}$ satisfying $\sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) = 1$. We also define the posterior probability of an edge existing between i and j that belong to communities $\mathbf{g}_i, \mathbf{g}_j$ by $\mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j) = P(\mathbf{A}_{ij} = 1 \mid \mathcal{D}, \theta) = \sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \mathbf{A}_{ij}$. For the E-step of the EM algorithm, following the same derivation logic as in the CEM-ER variation (in detail in Appendix B), we find that $\mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j)$ is the posterior probability that the edge (i, j) exists and is different depending on whether users i, j belong in the same community($\mathbf{g}_i = \mathbf{g}_j = r$) or not ($\mathbf{g}_i = r, \mathbf{g}_j = s, r \neq s$):

$$\mathbf{Q}_{ij}(r, r) = \frac{p\alpha^{\mathbf{M}_{ij}\sigma_{ij}} (1-\alpha)^{\mathbf{M}_{ij}(1-\sigma_{ij})}}{p\alpha^{\mathbf{M}_{ij}\sigma_{ij}} (1-\alpha)^{\mathbf{M}_{ij}(1-\sigma_{ij})} + (1-p)\beta^{\mathbf{M}_{ij}\sigma_{ij}} (1-\beta)^{\mathbf{M}_{ij}(1-\sigma_{ij})}}, \tag{4.21}$$

$$\mathbf{Q}_{ij}(r, s) = \frac{q\alpha^{\mathbf{M}_{ij}\sigma_{ij}} (1-\alpha)^{\mathbf{M}_{ij}(1-\sigma_{ij})}}{q\alpha^{\mathbf{M}_{ij}\sigma_{ij}} (1-\alpha)^{\mathbf{M}_{ij}(1-\sigma_{ij})} + (1-q)\beta^{\mathbf{M}_{ij}\sigma_{ij}} (1-\beta)^{\mathbf{M}_{ij}(1-\sigma_{ij})}}. \tag{4.22}$$

Notice that for $\mathbf{M}_{ij} = 0$, $\mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j)$ becomes equal to the prior probability p if $\mathbf{g}_i = \mathbf{g}_j$ and equal to q if $\mathbf{g}_i \neq \mathbf{g}_j$. Next, to maximize the likelihood in terms of the parameters we find:

$$\alpha = \frac{\sum_{i \neq j} \mathbf{M}_{ij} \sigma_{ij} \mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j)}{\sum_{i \neq j} \mathbf{M}_{ij} \mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j)}, \tag{4.23}$$

$$\beta = \frac{\sum_{i \neq j} \mathbf{M}_{ij} \sigma_{ij} (1 - \mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j))}{\sum_{i \neq j} \mathbf{M}_{ij} (1 - \mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j))}, \tag{4.24}$$

$$p = \frac{1}{\sum_{i \neq j} \mathbf{1}(\mathbf{g}_i = \mathbf{g}_j)} \sum_{i \neq j, \mathbf{g}_i = \mathbf{g}_j} \mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j), \tag{4.25}$$

$$q = \frac{1}{\sum_{i \neq j} \mathbf{1}(\mathbf{g}_i \neq \mathbf{g}_j)} \sum_{i \neq j, \mathbf{g}_i \neq \mathbf{g}_j} \mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j). \tag{4.26}$$

To find the diffusion probabilities σ_{ij} we must solve the following linear optimization problem:

$$\max_{\boldsymbol{\sigma}} \sum_{i \neq j} \sigma_{ij} (\mathbf{W}_{ij} - \lambda W) \tag{4.27}$$

$$\text{s.t. } \boldsymbol{\sigma} \in F_{\boldsymbol{\sigma}},$$

$$\text{where } \mathbf{W}_{ij} = \mathbf{M}_{ij} \left(\mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j) \log \frac{\alpha}{1 - \alpha} + (1 - \mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j)) \log \frac{\beta}{1 - \beta} \right),$$

$$\lambda > 0 \text{ some given penalty for regularisation, and } W = \max_{i \neq j} \mathbf{W}_{ij}.$$

The final CEM algorithm, when we choose the SBM prior is shown in Algorithm 1. It iterates between finding an optimal value for q, via the $\mathbf{Q}_{ij}$ values, and then holding it constant to maximize the likelihood (the right-hand side of (A.10) in Appendix B) with respect to $\theta = \{\alpha, \beta, p, q, \boldsymbol{\sigma}\}$ (M-step). We underline that the updates of the $\mathbf{Q}_{ij}$ values that are essential for the E-step require the knowledge of the communities participation vector g. It is updated in each iteration as follows: we generate first a network from the current $\mathbf{Q}_{ij}$ estimations, by drawing an edge whenever $\mathbf{Q}_{ij} > 0.5$. To get the updated vector g, we apply to the generated network the Louvain method, which returns a single community label for each user node (Blondel et al., 2008). Our algorithm converges when the L2 norm

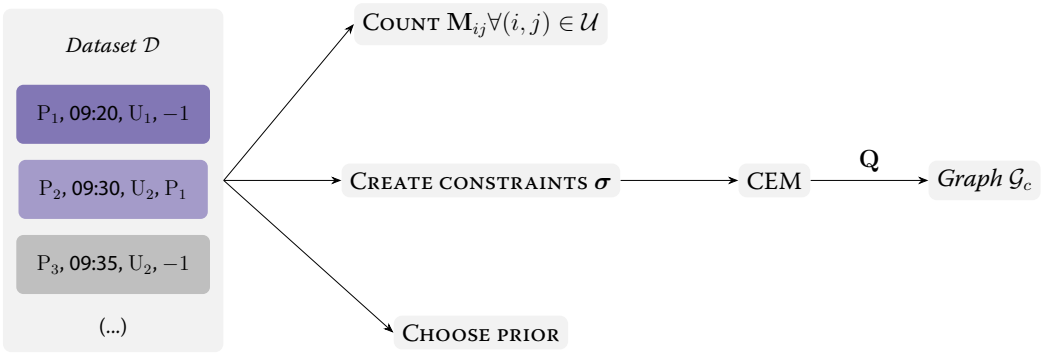


Fig. 4.4. – *Framework of Constrained Expectation Maximization.*

of improvement $||\mathbf{Q}_{new} - \mathbf{Q}_{old}||$ falls under some threshold ϵ that we choose in advance, where $\mathbf{Q}$ is the matrix with the $\mathbf{Q}_{ij}$ values.

Our intuition is that the addition of the community detection parameters p and q, which help the method group the users in communities, can guide the inference towards better results. This is reflected on the updated parameter $\mathbf{Q}_{ij}(\mathbf{g}_i, \mathbf{g}_j)$ - the posterior probability that the edge (i, j) exists - and is different depending on whether users i, j belong in the same community ($\mathbf{g}_i = \mathbf{g}_j = r$) or not ($\mathbf{g}_i = r, \mathbf{g}_j = s, r \neq s$). Before, in the CEM-ER case, we did not have this option, as the probability for a link to exist ($\mathbf{Q}_{ij}$ in Eq. (4.14)) is not influenced by users' community participation. Given that the reposting activity of users is heavily influenced on whether they participate in the same community or not, we believe that the addition of this parameter is the determining factor for the improvement of CEM.

4.6 Methodology

4.6.1 Datasets

The general framework of CEM for both priors is shown in Fig. 4.4. To evaluate our two methods CEM-ER and CEM-SBM against the ground truth and compare them with existing methods, we will use two different datasets: a synthetic and a real-world one. The synthetic dataset that we create aims to illustrate our method's efficiency when the dataset includes sufficient information about the interactions between users. As we will show later, this is not always the case with real-world datasets coming from OSNs, which can make the inference task even more challenging.

Algorithm 1 Iterative process of CEM

Input: PRIOR, $\mathcal{D}, \mathcal{U}, \mathbf{M}$
Output: $\mathbf{Q}, \alpha, \beta, \rho, \boldsymbol{\sigma}$
$t = 0$
Random Initialisation: $\alpha_t, \beta_t, \rho_t, \boldsymbol{\sigma}_t$
if PRIOR = ER **then**
  repeat
    $t \mathrel{+}= 1$
    $\mathbf{Q}_t$ = UPDATE $\mathbf{Q}(\alpha_{t-1}, \beta_{t-1}, \rho_{t-1}, \boldsymbol{\sigma}_{t-1})$ using (4.14)
    α_t = UPDATE $\alpha(\mathbf{Q}_t, \boldsymbol{\sigma}_{t-1})$ using (4.15)
    β_t = UPDATE $\beta(\mathbf{Q}_t, \boldsymbol{\sigma}_{t-1})$ using (4.15)
    ρ_t = UPDATE $\rho(\mathbf{Q}_t)$ using (4.16)
    $\boldsymbol{\sigma}_t$ = UPDATE $\boldsymbol{\sigma}(\mathbf{Q}_t, \alpha_{t-1}, \beta_{t-1})$ using (4.17)
  until CONVERGENCE
else if PRIOR = SBM **then**
  Random Initialisation: $\mathbf{g}_t$
  repeat
    $t \mathrel{+}= 1$
    $\mathbf{Q}_t$ = UPDATE $\mathbf{Q}(\alpha_{t-1}, \beta_{t-1}, p_{t-1}, q_{t-1}, \boldsymbol{\sigma}_{t-1}, \mathbf{g}_{t-1})$ using (4.21), (4.22)
    α_t = UPDATE $\alpha(\mathbf{Q}_t, \boldsymbol{\sigma}_{t-1})$ using (4.23)
    β_t = UPDATE $\beta(\mathbf{Q}_t, \boldsymbol{\sigma}_{t-1})$ using (4.24)
    P_t = UPDATE $\rho(\mathbf{Q}_t)$ using (4.25)
    q_t = UPDATE $\rho(\mathbf{Q}_t)$ using (4.26)
    $\boldsymbol{\sigma}_t$ = UPDATE $\boldsymbol{\sigma}(\mathbf{Q}_t, \alpha_{t-1}, \beta_{t-1})$ using (4.27)
    $\mathbf{g}_t$ = LOUVAIN$(\mathbf{Q}_t)$
  until CONVERGENCE
end if

Synthetic dataset

For the generation of synthetic social media data, we follow the code found in Giovanidis et al. (2021). We first create a set of $N = 100$ users each of which has two random activity (posting and reposting) rates. Then, we create an SBM network between the users, with 7 different partitions of varying sizes, that represents the friendship network of the network. Users in the same community are connected with probability $p = 0.06$ and users of different communities are connected with probability $q = 0.007$. Each subgraph corresponding to a community is a random ER with connection probability p. For each user, we generate a set of random timestamps, that increase according to an exponential distribution that depends on their activity rates. These timestamps represent the times they posted or reposted something. We generate a set of 100,000 timestamp-user-activity instances in total that we call the *Events* set. Additionally,

Tab. 4.3. Dataset statistics for the synthetic and real-world data.

Dataset statistics	Synthetic	#Élysée2017fr
Time span	17,459 time-steps	6 months
# tweets	1,709	293,405
# retweets	24,347	1,605,059
# users	100	11,521
% users with # tweets > 0	27.00	70.74
% users with # retweets > 0	87.00	96.45
% user pairs with $\mathbf{M}_{ij} > 0$	78.10	5.21

Tab. 4.4. Ground truth graph statistics for the synthetic and real-world data.

Ground truth graph	Synthetic	#Élysée2017fr
# edges	158	1,555,718
% intra-edges(labeled)	63.92 (101)	84.29 (1,311,463)
% inter-edges(labeled)	36.08 (57)	15.71 (244,255)
% edges with $\mathbf{M}_{ij} > 0$	99.36 (155)	45.23 (703,682)
% intra-edges with $\mathbf{M}_{ij} > 0$	100.00 (101)	50.13 (657,389)
% inter-edges with $\mathbf{M}_{ij} > 0$	98.25 (56)	18.95 (46,293)

we assume that each user has a Newsfeed that can hold up to 10 posts and reposts from their followees. Based on the friendship network and the Events set, we simulate a set of interactions between the users according to the following scheme: when a user i visits their Newsfeed, they repost randomly one of the 10 entries made by their followees. A new entry on the Newsfeed list will push out an older entry of a random position. The Newsfeeds of the users that follow user i will then be updated accordingly. Of course, in reality, users on a social media platform may show a preference towards a specific account or topic, or even repost something outside of the scope of their followees. The random uniform selection, however, makes the simulation collect sufficient information for all the edges in the friendship graph.

The simulation generates a social media dataset from which we can extract all the quantities that are necessary for our method, as presented in Section 4.1.1. The detailed statistics of the synthetic dataset can be found in Table 4.3. Table 4.4 shows instead the statistics of the ground truth graph. This is the network that we will be trying to infer. The intra-edges refer to the edges inside a community, whereas inter-edges refer to the edges between different communities.

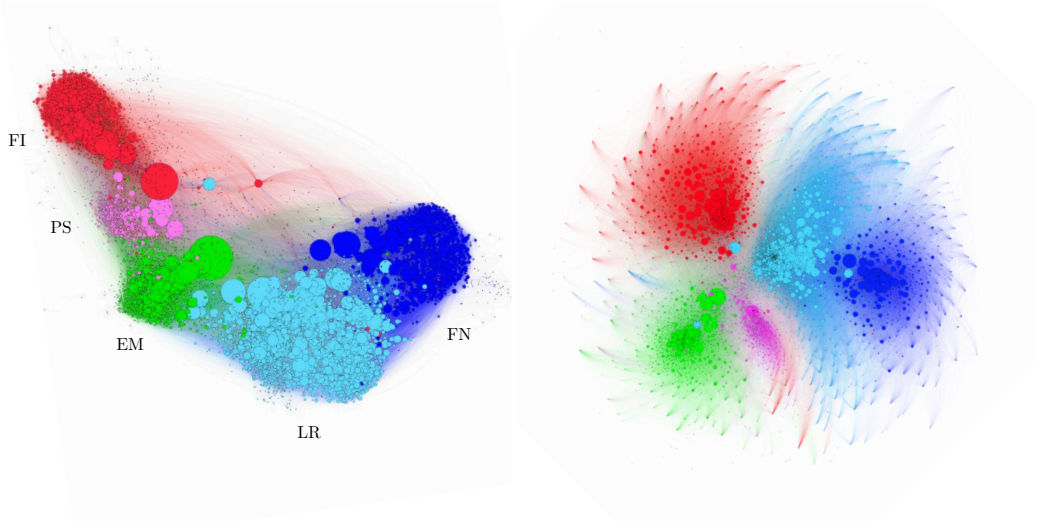
Real-world data: the #Élysée2017fr dataset

For the evaluation of our method on real-world data, we choose #Élysée2017fr, a publicly available dataset related to the 2017 French presidential campaign on Twitter (Fraisier et al., 2018). It features 2,414,584 tweets and 7,763,931 retweets from 22,853 Twitter profiles discussing the election. Users have been manually annotated by experts with political affiliations expressing support for one of the 5 main competing parties in France:

- FI: France Insoumise, far-left (Jean-Luc Mélenchon)
- PS: Parti Socialiste, left-wing (Benoît Hamon)
- EM: En Marche, center (Emmanuel Macron)
- LR: Les Républicains, right-wing (François Fillon)
- FN: Front National, far-right (Marine Le Pen)

The exact timestamps of interactions between users had not been published by the creators of the dataset, therefore we had to crawl them using the Twitter API. On top of that, we collected the follower-followee connections, i.e., the friendship network of the observed user IDs, which had not been provided by the authors. This is a crucial step since most network inference methods do not have a ground truth to compare their results with. It is important to note that at that time, Twitter was still providing access to the friendship networks (January 2021). However, since then, many changes have been made to their API policies and this information has been restricted. A visual representation of the friendship network that we collected along with the community participation of each node is shown in Fig. 4.5a. The network of retweets is shown in Fig. 4.5b. From these figures, we can see that even though users follow people from other communities (e.g., there are many friendships between the two extreme parties FI and FN), they mostly retweet posts from authors that belong inside their community and they do not interact much with users outside.

From this dataset, we keep only the tweets that have been retweeted by at least one user. Additionally, we remove retweets for which we do not know the author and retweets that have been made more than once by the same user. The statistics of the dataset after the above pre-processing along with the statistics of the ground truth network are shown in Tables 4.3 and 4.4.



(a) *Friendship network* (b) *Retweet network*

Fig. 4.5. – *Networks of the user-user connections provided by the #Élysée2017fr dataset, colored according to the party they support. In the case of the friendship graph (a) there is an edge* (i, j) *if* j *follows* i*. In the case of the retweet network (b) there is a weighted edge for every time that a user* j *has reposted a tweet authored by* i*. The size of each node is proportional to its degree.*

Insufficient information in a real-world dataset. From Tables 4.3 and 4.4 and Fig. 4.5a and 4.5b, we notice the main challenge in working with this dataset against the synthetic one: out of the 1,555,718 edges in the underlying friendship graph, only 45.23% of them have a non-zero $\mathbf{M}_{ij}$ value. On the other hand, the synthetic dataset includes information for more than 99% of the 158 existing edges. This can be partly because, in reality, users may repost their followees with some preference, instead of randomly selecting posts from their Newsfeed as is the case in the synthetic dataset. Therefore, many users may not appear to interact with retweets even if there is a connection between them in reality. However, given that the absolute numbers of the real-world dataset are quite high, we believe that there is sufficient information to work with.

4.6.2 Comparison

Compared models. We compare the graphs inferred by our two models, CEM-ER and CEM-SBM, with those generated by the following baseline and state-of-the-art methods:

– **Star**: a heuristic network inference method that draws a directed edge from the author of every tweet s in the dataset to every user that appears in the

corresponding diffusion episode $\mathcal{D}_s$ after them. The network inferred by Star implies that all the users that have retweeted a tweet are following its author.

– **Chain**: another heuristic method that generates a single long path between the users in each diffusion episode $\mathcal{D}_s$, according to the timestamps of their interactions with tweet s: each path first connects the author of s to the user i that retweeted it first in time. Then, it connects i to the user j who retweeted it second in time, j to the user who retweeted it third, and so on.

– **Saito et al.** (2008): a baseline EM-based algorithm that infers the influence probabilities k_{ij} by assuming an Independent Cascade model of diffusion between the users. For comparison, we produce the final network by drawing an edge (i, j) whenever $k_{ij} > 0.5$.

– **Netinf** (Gomez-Rodriguez et al., 2012): in a similar way to our work, Gomez-Rodriguez et al. identify the network that most accurately explains the observed infection times of nodes. However, their formulation of the problem is combinatorial and thus NP-hard to solve exactly. Therefore, they suggest finding near-optimal networks using approximation algorithms, by exploiting the submodularity properties of the objective, which, as we will show in the next sections, introduces computation-time and precision issues. In contrast, we devise a continuous linear expression based on the dataset, which allows us to find efficiently the exact solution to an LP optimization problem. As explained by the authors, when the activity rates are not the same for all users, the performance of the model worsens. Therefore, we expect Netinf to perform worse than CEM in more realistic settings such as these of the synthetic dataset, in which users have different activity rates. It should be noted that Netinf requires that we set in advance the parameter k, which is the number of edges that we want to infer. For comparison, we set k equal to the number of edges of the corresponding ground truth graph.

– **Newman** (2018): As mentioned before, our algorithm is an extension of the EM formulation provided in Newman's work. The algorithm is not designed to consider hidden paths between users, thus it is not guaranteed that the inferred networks will be feasible. For evaluation, we derive a

network by drawing an edge (i, j) whenever the friendship probability $\mathbf{Q}_{ij}$ for a user pair (i, j) estimated by this method is greater than 0.5.

– **Peixoto** (2019): a state-of-the-art non-parametric Bayesian method that infers posterior distributions from dataset observations using a stochastic block model as a prior. As is the case with our CEM-SBM model, it performs community detection together with network reconstruction. Unlike us, however, during the inference process, the model performs sampling using a Markov Chain Monte Carlo procedure and accepts a solution with a Metropolis-Hastings probability. As demonstrated next, this negatively impacts the computation time of the optimization.

Comparison metrics. The directed edges inferred by each inference method translate to the existence of follower-followee relationships between the respective user nodes. To evaluate and compare them against the ground truth, we will look at the following aspects:

1. **Effects of different dataset sizes and values of hyperparameter $\boldsymbol{\lambda}$.** Firstly, we check how different dataset sizes change the corresponding results of our method. For example, by choosing only the first 10,000 lines of the synthetic dataset, we obtain information for around 65% out of the $N\dot{(}N-1) = 9,900$ possible user pairs, whereas the whole dataset of 100,000 lines informs us on about 78% of the pairs. We see therefore that as we choose more dataset lines from the input, we get more information between users in terms of tweets and retweets (with diminishing returns). In general, we expect the performance of our model to improve with the increasing size of the dataset.

2. **Feasibility of the dataset.** We evaluate each method presented in Section 4.6.2 in terms of the terms of the metric of feasibility. Given the ground truth graph, we check how many diffusion episodes are feasible, according to our definition of feasibility provided in Section 4.2.

3. **Prediction performance.** When the ground truth is available, we can treat the output of the inference as a binary classification task between existing and non-existing edges. We, therefore, choose Precision, Recall, and AUC scores as metrics for evaluation and comparison. As shown in more detail in Section 2.5, these metrics are used frequently to measure prediction success in similar classification tasks. The AUC score is the area under the

ROC curve that represents the tradeoff between Recall (true positive rate) and $\mathrm{Specificity}$ (false positive rate), not to be mixed with the true and false positive utilization rates α and β of CEM.

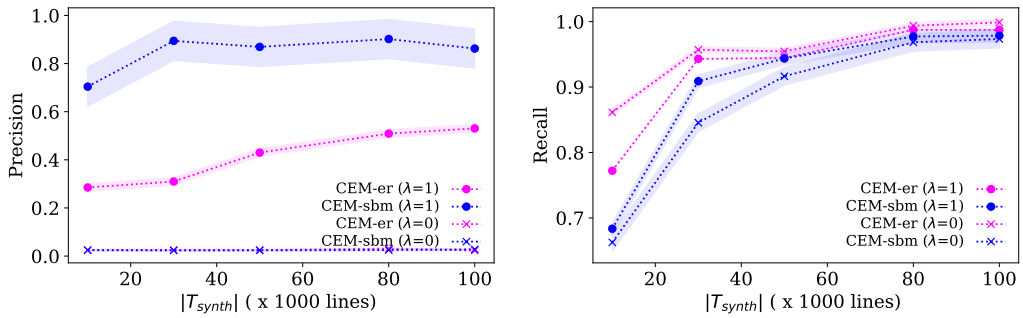
4. **Inferred network metrics.** Additionally, we will look into different statistics of the inferred network (e.g., average degree, diameter, connected components, etc), and compare them to those of the ground truth graph. These measures can be indicative of how much the inferred network resembles the properties of a general real-world network (in cases when the ground truth is not available), as explained in Chapter 2.

5. **Detection of communities.** A useful by-product of our CEM-SBM network inference method is the community detection task. Therefore, we check to what extent the inferred communities resemble the real ones presented in the ground truth. Since a node can only belong to one community, we wish to verify whether the different pairs of users belonging to the same or different communities are the same in the ground truth. The method for the evaluation and comparison is the following: we first generate a network for each model as described in Section 4.6.2 and then apply on it the Louvain method for community detection (Blondel et al., 2008). The detected clusters are then used to calculate the F_1 as follows: we look at each possible user pair and if the users belong to the same community we label the edge with 1 (positive class), otherwise with 0 (negative class). We do the same for the ground truth (with the Louvain labels). From the true/false positive, and true/false negative rates we measure the F_1, which combines Precision and Recall. In addition, we estimate the values of p and q between the communities in the inferred network and compare them to the real ones.

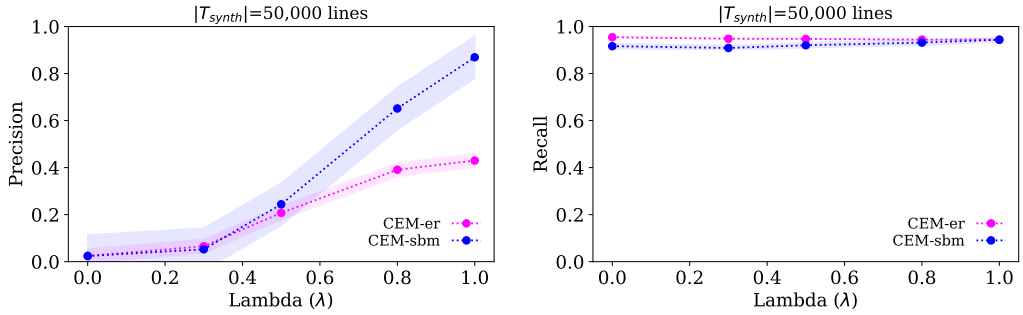
4.6.3 Experimental settings

We run the experiments on a virtual machine with 40 vCPUs and 256 GB RAM. For the solution to the optimization problem, we configure a Gurobi solver through PuLP[3], an open-source linear programming library for Python, using the dual simplex optimization method. The parameters set $\theta_1 = \{\alpha, \beta, r, \boldsymbol{\sigma}\}$ and $\theta_2 = \{\alpha, \beta, p, q, \boldsymbol{\sigma}\}$ for CEM-ER and CEM-SBM respectively are initialized uniformly at random in the range $[0, 1]$. As a convergence criterion for the optimization

[3]https://pypi.org/project/PuLP/



(a) *Results for different sizes of the synthetic dataset* $\mathcal{D}_{synth}$.



(b) *Results for different values of* $\lambda \in (0, 1)$ *given* $|\mathcal{D}_{synth}|$ = *50,000 lines.*

Fig. 4.6. – *The results on Precision (left) and Recall (right) for CEM-ER and CEM-SBM applied on the synthetic dataset.*

we choose the L2 norm of the difference between the values of $\mathbf{Q}$, i.e., $||\mathbf{Q}_{new} - \mathbf{Q}_{old}|| < \epsilon$, where the threshold ϵ is set equal to 0.001. Finally, to generate the unknown friendship network $\mathcal{G}$, we round up all edges with $\mathbf{Q}_{ij} > 0.5$ to 1, and the rest are set to 0. We run the experiments 10 times and report the average results.

4.7 Experiments on synthetic data

We first apply our method CEM on the synthetic dataset that we presented in Section 4.6.1. We examine the differences that we get in performance with respect to different sizes of input for the dataset, different values for the hyperparameter λ (see Eq. 4.17 and 4.27) and the different priors, ER and SBM. Then, we compare CEM to the network inference methods available in the literature, presented in Section 4.6.2.

4.7.1 Different sizes of input

Figure 4.6a illustrates the relationship between Precision and Recall across dataset sizes ranging from 10,000 to 100,000 lines. Larger dataset sizes correspond to higher Recall values, as anticipated due to the increased information availability which helps the identification of more underlying edges. Precision presents relatively stable behavior and is higher ($= 0.869$) when $\lambda = 1$. Overall, we see that CEM-SBM has higher performance than CEM-ER in terms of Precision which reaches up to 0.869 when $\lambda = 1$, and a slightly worse, but still competitive performance in terms of Recall, reaching up to 0.944 for $\lambda = 1$. We conclude therefore that CEM-SBM demonstrates greater Precision than CEM-ER while also effectively retrieving a significant portion of the underlying edges.

4.7.2 Different values of the hyperparameter λ

In Fig. 4.6b we can see more clearly how the choice of the hyperparameter λ inside the optimization objective (Eq. 4.17 and Eq. 4.27) affects the performance of the inference: for $\lambda = 0$ we get very low Precision ($= 0.024$) regardless of the prior since we infer the largest number of edges possible according to the objective, which in turn results to more false positive edges. However, in this case, the Recall value is at its highest (for example 0.954 in the case of CEM-ER). In contrast, for $\lambda = 1$, we infer a network with the smallest number of edges possible given the constraints and thus we get a considerably better Precision ($= 0.869$, CEM-SBM). The Recall value, in this case, is still high ($= 0.944$). This can be linked to the rich information that is provided in the synthetic dataset but can also be indicative of the good prediction probabilities of our method: we manage, with the help of the constraints, to infer the smallest set of edges possible (by setting $\lambda = 1$), that is precise and at the same time retrieves almost the entire ground truth graph.

Regarding the converged true positive optimization rate α, the value we obtain at the end of the optimization is almost equal to 1 for both $\lambda = 0$ and $\lambda = 1$. This means that there is an almost 100% probability that a post propagated through an edge that is present in the network we inferred. On the other hand, the converged value of β, showing the number of false positive utilized edges, is almost zero. This suggests that a post from the dataset almost always propagates through an edge that has been inferred[4].

[4]The term "almost" is used to show that the converged values of α and β are *nearly* equal to 1 and 0 respectively, with a difference much less than 0.001.

Tab. 4.5. Performance of different methods on a synthetic dataset with $|\mathcal{D}_{synth}|$ = 50,000 lines as input.

Performance	Precision	Recall	AUC	runtime (secs)
Star	0.141	**0.956**	0.931	**1.0**
Chain	0.033	0.955	0.752	**1.0**
Saito et al. (2008)	**1.0**	0.051	0.525	3.0
Netinf (2012)	0.159	0.165	0.575	2,199.0
Newman (2018)	0.522	0.450	0.724	2.0
Peixoto (2019)	0.643	0.924	0.958	3,481.0
CEM-ER ($\lambda = 0$)	0.024	0.954	0.668	8.0
CEM-ER ($\lambda = 1$)	0.430	0.944	0.962	9.0
CEM-SBM ($\lambda = 0$)	0.024	0.916	0.650	1.4
CEM-SBM ($\lambda = 1$)	0.869	0.944	**0.970**	1.5

4.7.3 Difference between priors

As λ approaches 1, the differences between the ER and SBM priors become more apparent (see Fig. 4.6b). SBM proves more effective in inferring true positive connections between users while minimizing false positives, achieving a Precision close to 0.9. This suggests that in CEM-SBM, using priors p and q as in Eq. 4.25 and Eq. 4.26, which account for whether a user pair (i, j) belongs to the same community, significantly improves optimization performance compared to a global parameter r (as in Eq. 4.16) unaware of community structure.

Moreover, as detailed in Table 4.7 and further explained later, CEM-SBM is more efficient in detecting underlying communities compared to CEM-ER: For $\lambda = 1$, the estimated p and q values derived by CEM-SBM are much closer to the ground truth ($p = 0.063$ and $q = 0.006$), with small relative errors ($\epsilon_p = 0.05$ and $\epsilon_q = 0.143$), while achieving an almost optimal F_1 score ($= 0.961$). This represents a significant improvement over the F_1 score provided by CEM-ER ($= 0.419$).

4.7.4 Comparison between methods

For a first understanding of the inner workings of each method that we compare with, we can zoom into the propagation network inferred for a random diffusion episode $\mathcal{D}_s = \{22, 17, 18, 81\}$ from the synthetic dataset (Fig. 4.8). Each method receives as an input the first 50,000 lines of the original dataset that contains 859 tweets and 12,236 retweets. The ground truth tells us that users 18 and 81 have

Tab. 4.6. Network statistics of the network inferred by each method compared to the ground truth for $|\mathcal{D}_{synth}|$ = 50,000.

	feasibility(%)	#edges	avg out-degree	max out-degree	max in-degree	diameter	avg shortest path	max scc (% users)
Synthetic graph	100.00	164	1.64	39	15	5	2.57	11
Star	**100.00**	1,072	10.72	78	24	3	1.35	10
Chain	**100.00**	4,545	46.86	75	71	3	1.48	87
Saito et al. (2008)	2.33	8	0.50	1	1	1	1	0
Netinf (2012)	34.80	164*	2.49	9	12	12	4.76	24
Newman (2018)	72.29	138	**1.55**	77	3	1	1	0
Peixoto (2019)	98.02	227	2.34	36	11	10	3.42	19
CEM-ER ($\lambda = 0$)	**100.00**	$6{,}175 \pm 89$	63.66 ± 0.92	94.5 ± 0.17	96	2	1.34	97
CEM-ER ($\lambda = 1$)	**100.00**	349 ± 8	3.59 ± 0.09	45.9 ± 2.03	$\mathbf{15.2 \pm 0.2}$	**5**	2.23	10
CEM-SBM ($\lambda = 0$)	**100.00**	$6{,}141 \pm 244$	63.31 ± 2.52	91.5 ± 2.95	92.6 ± 3.4	2.1	1.34	97
CEM-SBM ($\lambda = 1$)	**100.00**	$\mathbf{177 \pm 11}$	1.83 ± 0.12	$\mathbf{41.3 \pm 1.65}$	11.3 ± 0.15	5.2 ± 0.2	**2.61**	$\mathbf{11.9 \pm 0.6}$

* chosen a priori

reposted user 17, who had previously reposted directly the author user 22. As we see in Fig. 4.8, our method CEM-SBM ($\lambda = 1$) and Peixoto (2019) have inferred the propagation network of the diffusion episode correctly. CEM-ER ($\lambda = 1$) has inferred one more false positive edge from 22 to 18 whereas Star and Chain have inferred two false positive edges. Netinf (2012) has inferred only one false positive edge from 18 to 81 whereas the methods by Newman (2018) and Saito et al. (2008) have inferred no edge at all. Of course, this is only one example of a subgraph inferred by each method. We are going to see next the performance and statistics of the entire friendships graphs inferred.

Performance comparison. Firstly, we are comparing CEM with the other methods by looking into the graphs and the performance of each model. More specifically, we will compare the performance of each method in terms of Precision, Recall, and AUC. The results are shown in Table 4.5 and are combined with observations from each graph's statistics, found in Table 4.6[5].

From there we observe that the two heuristics, Star and Chain, give 100% feasible solutions. However, both methods infer graphs with thousands of edges (1,072 and 4,545 edges respectively) and high average out-degrees (10.72 and 46.86) which is very far from reality: the ground truth features only 164 connections with an average out-degree of 1.64. This may result in high Recall and AUC scores but comes at the cost of a very low Precision rate (0.141 and 0.033 respectively, as seen in Table 4.5). Additionally, both methods infer graphs with very small average shortest paths (< 1.5). In contrast, the ground truth has an average shortest path of 2.57 which is closer to the value that we would expect from a real-world Twitter network to have. Moreover, Chain infers graphs that are too dense, as seen from its maximum strongly connected component (last column, Table 4.6: it includes 87% of the users, whereas the actual value is only 11%). The above suggests that, given the synthetic dataset as input, Star and Chain infer graphs that are feasible but demonstrate properties that are far from these of the actual graph, and also, from these of a real-world network in general.

The method of Saito et al. (2008) is 100% precise but generates only 8 edges, a very low number for it to be considered a sufficient solution to our problem. Consequently, it presents a very low feasibility rate: it can only explain 2.33% of the diffusion episodes presented in the dataset. As a result, its network properties

[5]The highest value is marked with boldface and the second highest value is underlined. max scc: maximum strongly connected component.

are far from those of the real graph. For example, the maximum out and in-degrees of the network are equal to 1, along with the diameter and the average shortest path. Furthermore, the network inferred by Saito has no strongly connected component and has a very low average out-degree of 0.5.

For the Netinf (2012) model, we set in advance $k = 164$ as the number of edges that we want to infer, which is equal to the number of edges of the real network (however such information will not be available in practice and the authors suggest trying different values of k depending on the desired outcome). As we see, the inferred network has feasibility $= 34.8\%$ while also performing poorly on Precision ($= 0.159$), Recall ($= 0.165$), and AUC (=0.575). This is accompanied by weak network statistics: it has a relatively low maximum out-degree ($= 9$ whereas the real value is 39), the largest diameter out of all the methods ($= 12$), and its maximum strongly connected component is more than two times bigger than the real one (it covers 24% of the users).

The method by Newman (2018) returns a Precision $= 0.522$ and Recall $= 0.450$ which are values close to the output of a random classifier. However, it infers a network with 138 edges and an average degree of 1.55 which is close to the real numbers. Still, the diameter, maximum in-degree, and average shortest path values are really small compared to the ground truth. Additionally, it presents no strongly connected component. All in all, the network is neither feasible (feasibility $= 72.29\%$), nor competitive in terms of any performance or statistical metric, which could be due to the fact that it does not consider the hidden paths that exist between users and thus, loses a lot of information that is (indirectly) available in the dataset.

The method by Peixoto (2019) is the most competitive out of all the above methods, with feasibility $= 98\%$, Precision $= 0.643$ and Recall $= 0.924$. Additionally, the network presents some properties that are similar to the ground truth. For example, as we see in Table 4.6, the derived network has a maximum out-degree ($= 36$) whose value is the second closest to the real one ($= 39$). However, it generates almost 40% more edges and therefore the diameter and the maximum strongly connected component of the network is almost two times larger than the true one.

To compare with the above, both our methods, CEM-ER and CEM-SBM achieve feasibility $= 100\%$ across all λ values. In addition, CEM-SBM ($\lambda = 1$) achieves

Tab. 4.7. Performance of community detection for the synthetic network with $|\mathcal{D}_{synth}|$ = 50,000 lines.

Community parameters	$p_{\mathcal{G}_{synth}}$	$\lvert\epsilon_p\rvert$	$q_{\mathcal{G}_{synth}}$	$\lvert\epsilon_q\rvert$	F_1
Synthetic network	0.060	–	0.007	–	
Star	0.148	1.467	0.091	12	0.350
Chain	0.682	10.366	0.351	49.142	0.421
Saito et al. (2008)	0.500	7.333	–	–	–
Netinf (2012)	0.285	3.750	0.002	0.714	0.251
Newman (2018)	<u>0.043</u>	<u>0.283</u>	**0.007**	**0**	0.526
Peixoto (2019)	0.112	0.866	0.006	0.143	<u>0.731</u>
CEM-ER($\lambda = 1$)	0.085	0.416	0.021	2.000	0.419
CEM-SBM($\lambda = 1$)	**0.063**	**0.050**	<u>0.006</u>	<u>0.143</u>	**0.961**

the highest performance out of all the methods in terms of Precision, Recall, and AUC (=$0.869, 0.944, 0.970$ respectively). Furthermore, we see that the network inferred by CEM-SBM for $\lambda = 1$ has network properties almost identical to the ground truth, followed by the one inferred by CEM-ER ($\lambda = 1$).

Optimization runtime. On top of the good prediction and network statistics results, our algorithm is scalable and achieves running times that are close to the times of the heuristics and far lower than other alternatives (last column, Table 4.5). CEM-SBM for example runs in less than 1.5 seconds, which is close to the runtimes of Star and Chain. The methods by Newman (2018) and Saito et al. (2008) may have similar runtime, but they lose in accuracy. In contrast, Netinf (2018) and Peixoto (2019) need more than half an hour to converge and still, as we saw above, their results are not as competitive. This makes our optimization method powerful not only in terms of the accuracy of the prediction but also in terms of the time that is needed to reach a result.

Detection of communities. As shown in Table 4.7, our method CEM-SBM ($\lambda = 1$) achieves the highest F_1 ($= 0.961$) out of all the methods, followed by the method by Peixoto ($= 0.731$). Interestingly, the p, q parameters of CEM-SBM ($\lambda = 1$) are close to these of the ground truth ($p_{\mathcal{G}_{synth}} = 0.063$ with relative error $|\epsilon_p| = 0.05$ and $q_{\mathcal{G}_{synth}} = 0.006$ with relative error $|\epsilon_q| = 0.143$). Among the other methods, regarding p and q, we see that the method by Newman (2018) presents the lowest relative errors regarding the real values (0.283 and 0 respectively).

Tab. 4.8. Converged values for error parameters α, β given $|\mathcal{D}_{elysee}|$ = 5,000,000.

	$(1-\alpha^*)$	β^*
CEM-ER ($\lambda = 0$)	0	1.19e-10
CEM-ER ($\lambda = 1$)	1.32e-11	2.46e-12
CEM-SBM ($\lambda = 0$)	5.55e-16	1.07e-10
CEM-SBM ($\lambda = 1$)	0.004	0.001

4.8 Experiments on the #Élysée2017fr dataset

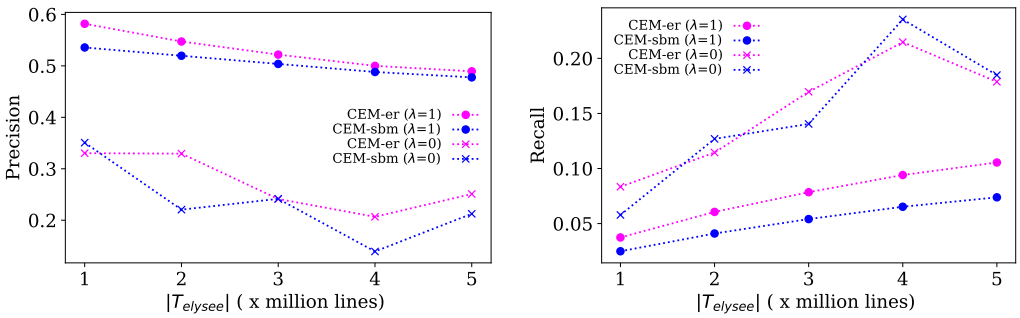
We now apply our method CEM on a real-world dataset that, as shown in Section 4.6.1, has different properties from the synthetic one. As before, we examine the differences that we get in performance with respect to different sizes of input for the dataset, different values for the hyperparameter λ, and the different priors. Then, we compare to the various network inference methods in the literature.

4.8.1 Different sizes of input

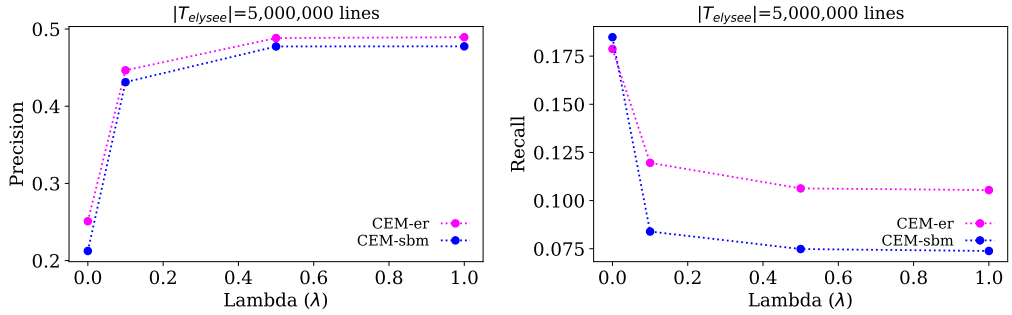
Figure 4.7a shows the relationship between the metrics of Precision and Recall given dataset sizes that range from 1 to 5 million lines. Again, as was the case with the synthetic data, the more information we have available, the higher the value of Recall will be. These values, however, will still stay at relatively low levels, under 0.1. As seen in Table 4.4, this is largely due to the fact that only 45.23% of the positive (i, j) edges in the ground truth appear in the dataset (i.e., they have $\mathbf{M}_{ij} > 0$). The rest of them do not appear in the measurements, therefore it is not possible to infer them given the specific dataset we have at hand. Still, we manage to predict thousands of edges that are mostly true positive (as seen from the Precision value). More specifically about Precision, we notice a slight drop as the size of the dataset increases. This makes sense, since we infer more edges the more data we get, and therefore we are more likely to make errors. The drop is milder when $\lambda = 1$ and more noticeable when $\lambda = 0$.

4.8.2 Different values of the hyperparameter λ

From Fig. 4.7b we notice that high values of λ given a constant dataset size (= 5 million lines) correspond to higher values of Precision. Here, we observe a trade-off between Precision and Recall, which was not evident in the synthetic dataset: in CEM-SBM for example, the lowest Precision ($= 0.213$) corresponds to the highest Recall value ($= 0.185$) when $\lambda = 0$ and a lower Recall value ($= 0.074$) corresponds to a higher Precision ($= 0.478$) when λ is set to 1. Therefore, we see



(a) *For different sizes of the dataset* $\mathcal{D}_{elysee}$ *and* $\lambda \in \{0, 1\}$.



(b) *For different values of* $\lambda \in (0, 1)$ *given* $|\mathcal{D}_{elysee}|$ *= 5 million lines.*

Fig. 4.7. – *Precision and Recall when applied on #Élysée2017fr.*

that depending on our goal, we can choose to prioritize Precision over Recall and vice-versa. This can be controlled by the correct selection of the hyperparameter λ. The converged error parameters α, β of CEM-ER and CEM-SBM on this real-world dataset can be seen in Table 4.8. We see that all the α values are close to 1, meaning that there is an almost 100% probability that a post spread through an edge that we predicted to exist in the inferred networks. For CEM-ER, the smaller value of β, which is almost equal to zero, suggests that there are zero false positive utilized edges. However, in the case of CEM-SBM ($\lambda = 1$), the slightly higher value of $\beta^* = 0.001$ suggests that there is a low, but existing probability, that a tweet passes via an edge that does not appear in the inferred ground truth. As we will see later, this may mean that we have missed some edges and therefore the overall feasibility rate may be (slightly) affected. Likewise, in the same case, the fact that $1 - a^* = 0.004$ means that there is a small probability of false negative utilized edges existing.

Tab. 4.9. Performance of each method for the #Élysée2017fr dataset*.

Performance	Precision	Recall	AUC	runtime(secs)
Star	0.446	0.133	0.565	**1**
Chain	0.262	0.130	0.563	**1**
Saito et al. (2008)	0.199	0.0001	0.500	342.00
Netinf (2012)	N/A	N/A	N/A	N/A
Newman (2018)	0.464 ± 0.031	0.066 ± 0.001	0.533 ± 0.001	25.00
Peixoto (2019)	N/A	N/A	N/A	N/A
CEM-ER ($\lambda = 0$)	0.251	**0.179**	0.586	37,721.00
CEM-ER ($\lambda = 1$)	0.489	0.105	0.552	35,552.00
CEM-SBM ($\lambda = 0$)	0.213	0.185	**0.589**	44,016.00
CEM-SBM ($\lambda = 1$)	**0.478**	0.074	0.537	88,504.00

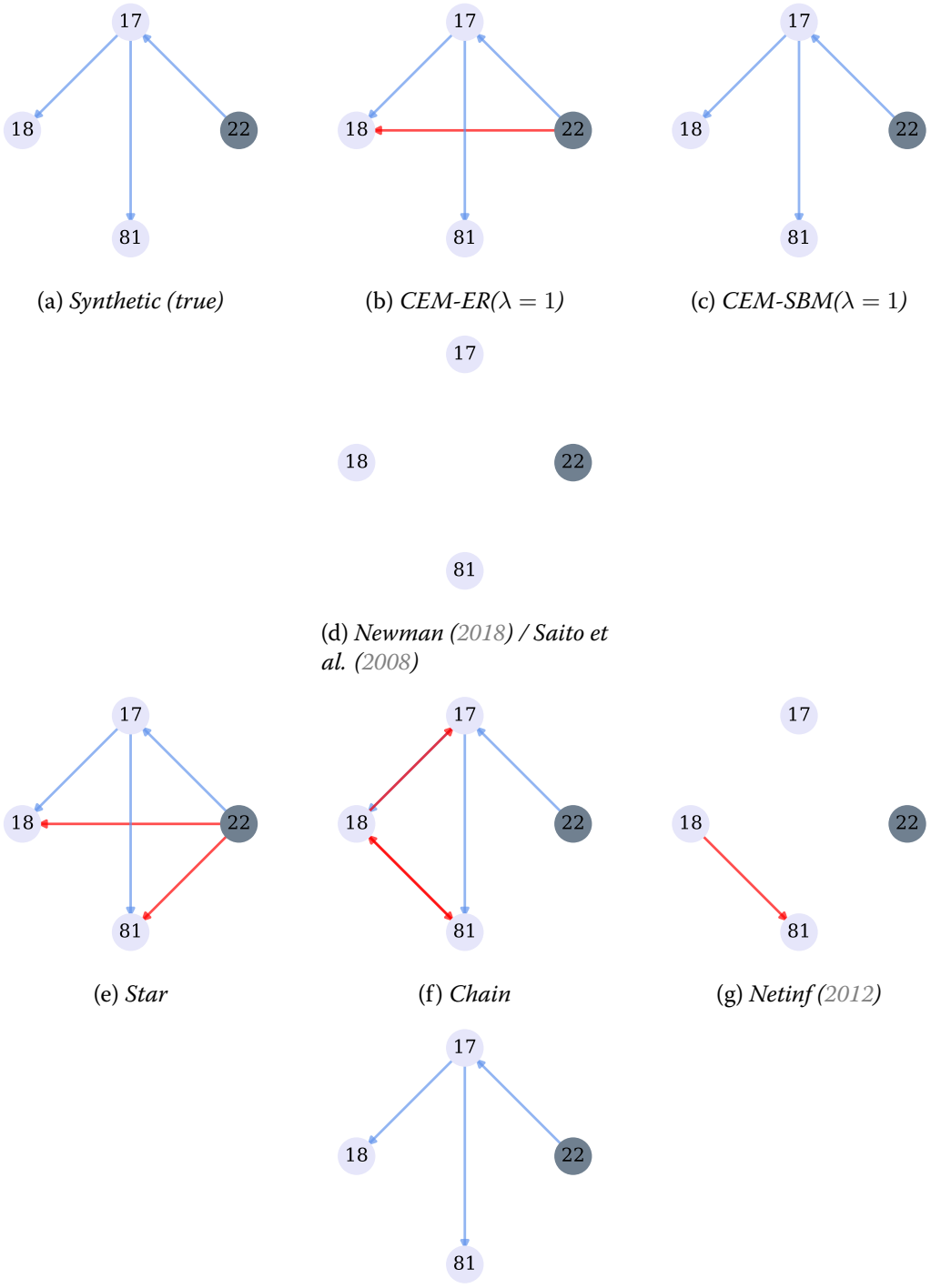
** N/A: no results after 48 hours*

4.8.3 Difference between priors

In contrast to the synthetic dataset case, from the above figures we notice that CEM-ER and CEM-SBM present more similar behavior. This is largely due to the properties of the dataset itself: we have relatively sparse information on the edges between users that belong to different communities (we observe only 18.95% of the existing inter-edges as seen in Table 4.4, in contrast to the 98.25% of the positive inter-edges in the case of the synthetic dataset). This makes sense since, in reality, users between different communities interact less often, so it is less likely that they will appear in a dataset when we collect it. Therefore, the benefit of using the SBM instead of the ER prior cannot be easily made obvious given the specific dataset that we have at hand. Still, the use of the SBM prior provides the highest Recall value ($= 0.185$, for $\lambda = 0$) and AUC value, ($= 0.589$, for $\lambda = 0$), which, as we will show later are also the largest values among all compared methods.

4.8.4 Comparison between methods

We compare the graphs inferred by our two models with the same methods presented before, this time when real-world data is given as input. Given our computational resources, we were not able to run the method by Peixoto (2019) and Netinf (2012) within reasonable timeframes (in < 48 hours), therefore they are left out of the comparison. Table 4.9 shows the Precision, Recall, and AUC performance of each method, and Table 4.10 shows the properties of each corresponding graph.



(a) *Synthetic (true)* (b) *CEM-ER($\lambda = 1$)* (c) *CEM-SBM($\lambda = 1$)*

(d) *Newman (2018) / Saito et al. (2008)*

(e) *Star* (f) *Chain* (g) *Netinf (2012)*

(h) *Peixoto (2019)*

Fig. 4.8. – *Example of a subgraph inferred by each method for a diffusion episode $\mathcal{D} = \{22, 17, 18, 81\}$ from the synthetic dataset, connecting its author (user 22) to every other user that retweeted it. Blue arrows show true positive edges and red arrows the false positive ones.*

Tab. 4.10. Network statistics of the graphs inferred by each method compared to the ground truth network for $|\mathcal{D}_{elysee}|$ = 5,000,000 lines.

Inferred network metrics	feasibility(%)	#edges	avg out-degree	max out-degree	max in-degree	diameter	avg shortest path	max scc (% users)
Ground truth	49.00	1,555,718	136.42	5,004	1,853	11	2.82	93.28 (10,747)
Star	**100.00**	463,290	40.25	**2,524**	1,069	**12**	3.70	66.05 (7,610)
Chain	**100.00**	768,122	66.73	1,122	1,256	8	3.04	**95.34 (10,984)**
Saito et al. (2008)	0.55	786	0.54	2	10	8	1.11	0
Netinf (2012)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Newman (2018)	37.24	237,063	22.43	$1{,}206 \pm 127$	558	12.7	4.32	52.57 (6,057)
Peixoto (2019)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CEM-ER ($\lambda = 0$)	**100.00**	1,108,079	96.26	2,336	1,262	9	3.06	80.31 (9,252)
CEM-ER ($\lambda = 1$)	**100.00**	335,289	29.13	2,291	790	12	3.82	66.07 (7,612)
CEM-SBM ($\lambda = 0$)	**100.00**	**1,353,432**	**117.58**	1,364	**1,609**	8	**2.95**	82.50 (9,505)
CEM-SBM ($\lambda = 1$)	99.37	240,893	20.97	955	775	**11**	3.58	72.81 (8,388)

From Tables 4.9 and 4.10 we observe that Star and Chain give 100% feasible solutions with Precision equal to 0.446 and 0.262 respectively and Recall values equal to 0.133 and 0.130. However, their network statistics resemble less those of the real graph: Star infers 463,290 edges, with max out-degree equal to $2,524$ and max in-degree equal to $1,069$. We consider these values quite high, given the number of edges inferred (they are comparable to the ground truth which has three times the number of edges of Star) and that's why we consider it less trustworthy. This result is expected due to the heuristic method of inferring the edges, which connects directly the author of a post to its reposters.

The network by Chain, as seen in the last column of Table 4.10, has the highest maximum strongly connected component (it includes 95.34% of all users), which is bigger than the corresponding size in the real network ($= 93.28\%$). Given that the inferred network by Chain is half the size of the real graph, this high percentage suggests that it is more densely connected than we would expect from a real graph. What is more, in a real-world graph, most nodes have a relatively small degree, but some of them will have a noticeably larger degree, being connected to many other nodes. However, in Chain, we do not notice this phenomenon.

As was the case in the synthetic dataset evaluation, the method of Saito et al. (2008) generates only a few edges ($= 768$) and is therefore not feasible. The method of Saito et al. (2008) may be again relatively precise, but presents no strongly connected component, has a very low average out-degree ($= 0.5$) and an abnormally high diameter ($= 8$), given the size of the graph. The above shows that the network inferred by this method is very sparse and does not resemble the real-world network in question. Likewise, the model by Newman (2018) is not feasible, but in this case, seems more competitive in terms of the Precision metric ($= 0.464$). However, its large diameter ($= 12.7$) given the size of the inferred network (5 times smaller than the real network which has a diameter $= 11$) prevents us from selecting it as a realistic option.

Compared with the above methods, our algorithm CEM, presents the highest values in terms of every metric: Precision, Recall, or AUC. This can be regulated either by choosing a value close to $\lambda = 0$, that returns the highest number of nodes (>1,100,000) and therefore a high Recall (=0.178), for CEM-SBM ($\lambda = 0$)) but lower Precision, or by choosing a value closer to $\lambda = 1$ that returns less than 340,000 nodes (for both priors) and therefore a lower Recall but a high Precision

Tab. 4.11. Performance of community detection for the real-world network with $|\mathcal{D}_{elysee}|$ = 5,000,000 lines.

	p_G	$\lvert\epsilon_p\rvert$	q_G	$\lvert\epsilon_q\rvert$	F_1
Ground truth	0.0012	N/A	0.0445	N/A	N/A
Star	0.0143	10.92	0.0003	0.99	0.858
Chain	0.0236	18.67	0.0004	0.99	**0.889**
Saito et al. (2008)	0.3834	318.5	N/A	N/A	0.447
Netinf (2012)	N/A	N/A	N/A	N/A	N/A
Newman (2018)	<u>0.0093</u>	<u>6.75</u>	5e-05	1	<u>0.888</u>
Peixoto (2019)	N/A	N/A	N/A	N/A	N/A
CEM-ER ($\lambda = 0$)	0.0346	27.83	<u>0.0005</u>	0.99	<u>0.888</u>
CEM-SBM ($\lambda = 0$)	0.0425	34.42	**0.0006**	0.99	0.880
CEM-ER ($\lambda = 1$)	0.0103	7.58	0.0002	1	0.887
CEM-SBM ($\lambda = 1$)	**0.0074**	**5.17**	0.0001	1	0.878

(=0.489, CEM-ER ($\lambda = 1$)). When it comes to the statistics of the graph, its diameter stays close to the real value (= 11). The same is true for the average shortest path. This illustrates that the two best values from each category are in favor of our CEM method.

Optimization runtime. We verify from the runtime column of Table 4.9 that our model is scalable since we managed to solve an optimization problem with 6,922,990 unknowns and 1,605,059 constraints in only a couple of hours. We achieve this not only by formulating the inference as a linear optimization problem but also by taking advantage of powerful optimization solvers that are publicly available (in our case, the Gurobi solver). On the other hand, the methods by Saito et al. and Newman present fast computation times (342 and 25 seconds) but, as we have seen, they present less competitive results in terms of feasibility and performance.

Detection of communities. As shown in Table 4.11, our methods CEM-ER ($\lambda = 0$) and CEM-ER ($\lambda = 1$) achieve a high F_1 (= 0.888 and 0.887), similarly to Newman's method (= 0.888) and Chain (= 0.889). Chain's high performance does not surprise us in this case since Chain favors the creation of communities all while inferring a very high number of edges compared to other methods. Despite this, all the p parameters estimated on the graphs by each method are far from the real ground truth value. This was expected since we are missing substantial information on how edges interact between different communities and we may

Tab. 4.12. Performance of CEM ($\lambda = 1$) given constant values of parameter β for #Élysée2017fr.

	Precision	Recall	AUC	feasibility (%)
CEM-ER ($\beta = 0$)	0.489	**0.105**	**0.552**	100.0
CEM-ER ($\beta = 0.5$)	0.592	0.060	0.530	66.96
CEM-ER ($\beta = 0.6$)	<u>0.604</u>	0.052	0.526	61.21
CEM-ER ($\beta = 0.7$)	**0.619**	0.040	0.520	**52.78**
CEM-SBM ($\beta = 0$)	0.478	<u>0.074</u>	<u>0.537</u>	99.37
CEM-SBM ($\beta = 0.5$)	0.552	0.054	0.527	71.86
CEM-SBM ($\beta = 0.6$)	0.558	0.048	0.524	65.65
CEM-SBM ($\beta = 0.7$)	0.566	0.041	0.520	<u>58.00</u>

therefore be overestimating the value of p while underestimating q. Still, our method for $\lambda = 1$ has the lowest relative error on the p parameter (7.58 for CEM-ER and 5.17 for CEM-SBM) along with Newman that has an $\epsilon_p = 6.75$.

4.8.5 Controlling feasibility through β

As expected, since 2017 (the year that the dataset was created), some Twitter profiles have been deleted or set to private. In addition, users may have retweeted a tweet/diffusion episode outside the scope of their followees (e.g., through Twitter search, recommendation algorithms, Twitter trends, etc.). As a result, the #Élysée2017fr dataset is not 100% feasible given the ground truth friendship graph. In other words, the current view of the friendship network does not explain all the diffusion episodes in the selected dataset; in fact, it can only explain 49% of them.

We can therefore control the feasibility of our result to match the feasibility of the dataset given the ground truth through the parameter β: for an inferred network to be feasible, we want the false positive utilization rate β, i.e. the average number of inferred edges that pass through an edge that does not exist in the inferred graph, to be as close to 0 as possible. If β is close to a non-zero value, it means that there is a $\beta > 0$ probability that influence has happened through a nonexistent edge in the inferred network and therefore some diffusion episodes may be left unexplained. Consequently, we can set β equal to a constant - instead of updating it through Eq. 4.15 or 4.24 - whose value depends on the feasibility that we wish the outcome to have. Hence, we will examine the relation of the inferred network to the ground truth given different constant values of β. In general, we expect

the inferred network to be more precise when the feasibility rate is close to this of the ground truth network (= 49%).

First of all, as we show in Table 4.12 when β increases feasibility decreases. For example, when $\beta = 0.7$ the feasibility of the dataset given the inferred network is 52.78% and 58% for CEM-ER and CEM-SBM respectively. We note that the value of β changes only the overall number of edges inferred, which indirectly affects the number of diffusion episodes that are explained in the dataset. Furthermore, as β increases, and hence feasibility decreases, we get closer to the actual 50% feasibility and Precision improves. We should underline that when the feasibility rate falls lower than 50% (for $\beta > 0.7$), Precision falls dramatically since the algorithm starts inferring edges randomly, without really respecting the constraints.

4.8.6 Evaluation with no ground truth

Overall, we observed that the metric of feasibility can be beneficial by controlling the quality of the inferred graph. For example, we saw that the methods with the lowest feasibility rates infer graphs with low predictive quality and present statistics that are far from those of the real-world graph. The benefit of CEM over other methods is especially apparent when we have collected sufficient data between edges, as was the case in the synthetic dataset case. Consequently, when the underlying friendship network is not available, which is often the case in network reconstruction problems, the feasibility rate of the inferred network could be an effective indicator of a method's performance. However, feasibility is not a sufficient condition for better prediction results. As we see in the case of Star, Chain, and CEM for $\lambda = 0$, a 100% feasibility rate cannot guarantee a precise result. Moreover, as we showed, we can use empirical values about how much feasibility to require in the inferred network-based, for example, on how old the dataset is, or how often users retweet outside of their connections, e.g., using recommendations. On top of feasibility, we could look into the inferred graph's statistics and evaluate to what extent they are similar to these of a general, real-world graph. Usual indicators of such real-world properties are the average degree, the diameter, the average shortest path, and the strongly connected components of the graph.

4.9 Conclusions

As we observed above, CEM successfully produces feasible graphs that are closer to reality when compared to heuristic and state-of-the-art methods. We validated the results both on synthetic and real-world datasets, using two different network priors, Erdős-Rényi (ER) and Stochastic Block Model (SBM), and noticed that CEM produces results that in most cases return the two most accurate values among all chosen compared metrics, and does so significantly faster than the state-of-the-art. Moreover, by selecting values between 0 and 1 for the hyperparameter λ, we can control the trade-off between the Precision and Recall of the result. The choice of SBM as a prior was motivated by the fact that it might better inform link inference given that it provides a model that is more realistic to the way that users online interact with each other and form communities. An important consequence of this choice in our case is that in each EM iteration the probability of a link existing between the users now depends on whether we have inferred that a user pair belongs to the same community or not. This adds an additional type of information during the EM updates and thus returns better inference and community detection results compared to the ER prior. We show that the contribution of SBM is more apparent when we have sufficient information on how nodes interact between different communities, like in the synthetic dataset case.

Furthermore, we observe that enforcing feasibility can guide the inference process towards more accurate, real-world graphs. For example, we show experimentally that when the dataset is 100% feasible given the ground truth, as is the case in the synthetic dataset, enforcing 100% feasibility produces graphs that are closer to reality. If we assume that the source of information in the dataset comes outside of the users' friendship connections, we can lower the percentage of feasibility that we expect from the inference. For example, in the case of the #Élysée2017fr dataset, where the dataset is only 50% feasible given the real graph, we observed that Precision improves as we force the feasibility of the inferred network to be lower, closer to the real percentage (through the parameter β). However, if we cannot be sure about the ground truth's real feasibility percentage, we still suggest working with $\beta = 0$, since it returns more realistic networks compared to other inference methods.

Keep in mind, that as we saw in the case of Star and Chain, feasibility is not a sufficient condition for the accuracy of the result: the graphs inferred by both

these methods are 100% feasible but present some extreme properties (e.g., large diameter, low maximum degree) that make the results less trustworthy. Therefore, given that the underlying data is usually not available in applications like ours, the evaluation of an inference technique can be quite challenging and may require looking into several metrics together with feasibility in order to decide the level of trustworthiness of the inference results. We could for example examine the inferred graph's statistics to determine whether or not they are comparable to those of a real-world graph, e.g., average degree, diameter, average shortest path, strongly connected components, which are typical real-world network indicators. We should note here that our method works with a specific dataset structure that is based on the data that most social media platforms currently offer. In principle, CEM has the potential to serve various network inference purposes, in other fields like epidemiology, biology, physics and more. However, leveraging it effectively in these cases would necessitate domain-specific expertise to tailor feasibility constraints accordingly.

* * *

The contributions of this Chapter have led to the following publications:

☐ Effrosyni Papanastasiou, and Anastasios Giovanidis. Constrained expectation-maximisation for inference of social graphs explaining online user–user interactions." *Journal of Social Network Analysis and Mining 13.1, Springer 2023: 41.* (Papanastasiou and Giovanidis, 2023)

☐ Effrosyni Papanastasiou, and Anastasios Giovanidis. Bayesian inference of a social graph with trace feasibility guarantees. *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), ACM, 2021.* (Papanastasiou and Giovanidis, 2021)

Additionally, the Twitter friendship network that we collected for the purposes of our study has also contributed to a different domain, beyond the scope of this thesis, specifically in the context of opinion dynamics:

☐ Antoine Vendeville, Anastasios Giovanidis, Effrosyni Papanastasiou, and Benjamin Guedj. Opening up echo chambers via optimal content recommendation. *International Conference on Complex Networks and Their Applications (pp. 74-85). Cham: Springer International Publishing.* (Vendeville et al., 2022a)

□ Antoine Vendeville, Anastasios Giovanidis, Effrosyni Papanastasiou, and Benjamin Guedj. Recommendation of content to mitigate the echo chamber effect. Extended abstract. *Conference on Complex Systems, 2022.* (Vendeville et al., 2022b)

* * *

A contrastive approach using Graph Neural Networks

5

Contents

As a final investigation for this thesis, this chapter introduces an alternative method for inferring feasible OSNs graphs that avoids the need for solving a strictly constrained optimization problem, as is the case of the previously proposed CEM approach. Although methods like these can deliver nearly optimal

solutions, they struggle to scale when given large datasets typical of OSNs; as a result, they depend heavily on industry-level solvers for optimization (like Gurobi). Considering the exponential growth in computational power of the 21st century, one could naturally start looking toward *Articifical Intelligence (AI)* methods. With its popular subfields like *Machine Learning (ML)* and *Deep Learning (DL)* that are constantly witnessing new innovations, we now have promising new alternatives that can provide more scalable and flexible algorithms for data-dependent applications that are too difficult to be solved via traditional methods (LeCun et al., 2015; Alpaydin, 2020). Network inference can be considered as such a problem, motivating us to raise the following question: *Can we leverage scalable learning techniques from the field of ML to provide equivalent solutions to the feasible network inference problem for OSNs at a lower memory and computational cost?* In this case, we find it preferable to avoid the strict combinatorial approach which demands constructing a full set of constraints as in Chapter 4, searching instead for an AI approach that can indirectly incorporate the feasibility into the learning process.

In this final chapter, we first provide an overview of ML and the different learning settings where it can be applied, including when targeting the graph domain of OSNs. In an effort to find alternatives to our CEM inference framework, we conduct an extensive literature review of various ML and DL categories and algorithms tailored to graphs, questioning whether they could potentially be applied to our problem. After addressing their respective limitations when it comes to feasible network inference in unreliable settings, we are motivated to propose a novel, simple solution as a flexible alternative. We present our method along with some preliminary experiments on the #Élysée2017fr dataset, which provide promising results, and conclude with a discussion on its limitations and potential future improvements.

5.1 Machine learning background

While traditional statistical methods focus on modeling distributions and diffusion dynamics, the field of ML is primarily concerned with developing algorithms that can make predictions based on data inputs (Goodfellow et al., 2016). ML algorithms are mostly task-driven, providing models that can learn from data to perform specific tasks. This approach is particularly useful for tasks that are too complex to solve for traditional programs, which lack the ability to learn (LeCun et al., 2015). Goodfellow et al. (2016) illustrate this difference with a

simple example: If our goal is to make a robot walk, the task is walking. We could either manually write a program with fixed rules for walking, or develop an ML algorithm that enables the robot to learn to walk on its own, without explicit instructions. Similarly, when we apply this line of thought to our problem of feasible network inference, we have two options: we can either manually define the set of feasibility constraints and solve them using statistical optimization methods, or we can select ML algorithms capable of incorporating feasibility autonomously. However, these algorithms must be adapted considerably to handle the intricacies of graph-structured data, leading to the specialized and growing field of *graph machine learning*(Hamilton, 2020). In this section, we provide a brief overview of the standard machine learning approaches and explain how they can be adapted to handle the specific characteristics of graph-shaped data, particularly in the context of OSNs.

5.1.1 Supervised vs unsupervised learning

The different ML approaches that exist can be initially divided into two global categories, according to the type of task that we are focusing on: It can either be a *supervised task*, where the general goal is to predict a target output given an input dataset, or an *unsupervised* task, where the goal is to detect patterns, such as clusters of data points in the data (Berry et al., 2019; Hamilton, 2020). Various kinds of tasks may fall under these two categories. The most common one is the task of *classification*, where a program is asked to specify in which out of k possible categories a data point belongs to (Kotsiantis et al., 2007). This is also viewed as a process of predicting *labels* for each data point[1]. In any kind of ML task, the dataset holds the most important role in the learning process.

The standard practice when the learning is supervised, is to divide the available data into three different subsets: (i) the *training set*, which consists of a certain portion of the data points with known labels from which the model will learn; (ii) the *validation set*, which consists of the same splitting technique but on a different subset of points, typically used to tune and monitor the model's performance during training; and (iii) the *test set*, which serves as an unseen set of data points for which we make predictions after training, providing an unbiased estimate of how well the model is expected to perform on new data points (Raschka, 2018). Partitioning the data into training-validation-test sets is crucial to ensure that

[1] For a more extensive list of the various ML tasks, we refer the reader to the book "Deep learning: foundations and concepts" by Bishop and Bishop (2024).

the model generalizes effectively to unseen data. Various splitting techniques are available, and the ideal split ratio usually varies depending on the application. Generally, the training set should be large enough to capture the inherent variability in the data, but not so large to result in *overfitting* (Ying, 2019). A common approach for example involves dividing the data into 60-80% for training, 10-20% for validation, and 10-20% for testing (Dobbin and Simon, 2011).

When the learning is unsupervised instead of supervised, it is considered more challenging since it assumes that there are no labeled data points to guide the learning process. In this setting, the focus is less on predicting accurate labels and more so on exploratory data analysis tasks such as clustering, anomaly detection, and visualization, among others (James et al., 2023). In our case, the problem of network inference in OSNs deviates from supervised learning, since, by definition, the dataset set at hand does not provide reliable or sufficient information. As a result, the data-splitting strategy is not as straightforward and traditional supervised learning approaches cannot be directly applied to the task. Instead, we may need to look further into the unsupervised learning techniques that do not rely on labels, while also addressing the additional challenge of handling graph-structured data. These techniques will be examined in the following sections.

5.1.2 Graph machine learning for Online Social Networks

When dealing with graphs composed of multiple components, it becomes important to categorize the machine learning task based on the specific parts of the graph we are examining, rather than simply distinguishing between supervised and unsupervised methods. We can either stay on a local level, extracting information about individual nodes or edges, or broaden our understanding to a global level, learning about clusters, communities, or the overall structure of the graph (Hamilton, 2020). Below we list some of the most well-known tasks for graphs and examples of their usage given datasets from OSNs:

– *Node classification* is the task of learning how to classify the nodes of a graph into different *classes* or *labels*. Depending on the application, a node can have one or more labels (*binary* versus *multi-label classification*) (Bhagat et al., 2011). In the context of OSNs for example, depending on how we represent a graph, we can use this task to classify whether a user node is a bot or a human, if a tweet includes false or true information, or

the kind of community that a user belongs to (*community detection*) (Xiao et al., 2022).

– *Link prediction* or *link classification* focuses on classifying the edges of a graph (Martínez et al., 2016). As in the case of node classification, the labels of the edges can either be binary, e.g., predicting if an edge exists or not, or belong to a wider set of categories. Social media platforms are benefiting greatly from link prediction algorithms, utilizing them as tools to predict the type of content a user is likely to engage with or the formation of connections between users, among various other applications (Hasan and Zaki, 2011; Daud et al., 2020).

– *Graph classification* aims to classify entire graphs based on their overall structure. In these problems, multiple graphs are available, and the goal is to learn how to classify each one of them into distinct classes (Cai et al., 2018). For example, in the context of OSNs, the structure of a tweet can be represented as an individual graph and the goal of graph classification will be to detect whether they contain false information (Shu and H. Liu, 2022).

Although the general learning process for these tasks remains similar, extending ML methods to graph-shaped data requires some additional attention. This is because, in standard machine learning models, the individual data points are assumed to be *independent and identically distributed (i.i.d.)* to allow for the model to generalize to new data points. However, this assumption is violated in graphs where the nodes are interconnected and dependent on each other (Hamilton, 2020). Thus, it is crucial to model the relationships between nodes rather than treating them as independent data points.

To address this, research has suggested leveraging a key property of real-world graph structures known as homophily — the tendency of nodes to connect to other nodes with similar characteristics (refer to Section 2.3.2 for more details). Motivated by the concept of homophily, we can develop learning models that assign similar labels to nodes within the same neighborhoods, capturing the inter-dependencies and contextual information that are unique to a graph's structure. This intuition has led to novel strategies, extending beyond conventional machine learning techniques, with many of them relying on deep neural networks, such as *Graph Neural Networks (GNNs)* (Scarselli et al., 2008).

In the present thesis: focusing on link prediction

Since the focus of network inference is on edges, rather than on node labels, the task that is more relevant to our context is link prediction. From a statistical standpoint, there is a valid question about whether the term "inference" can be used interchangeably with the term "prediction", as is often the case in machine learning. Central to this debate is the need for a distinction between the concept of inference as used in statistics, which typically involves inferring latent variables and addressing associated uncertainties, and prediction/inference as part of an ML algorithm, which focuses more on generating accurate predictions. For this chapter of the thesis, we adopt a broader interpretation of "inference", encompassing all approaches aimed at providing edges that have not yet been provided by the data.

5.2 Review of representation learning approaches

When looking for more scalable and flexible solutions to modeling feasible network inference for OSNs, the first step is to dive into the existing literature that may already provide some alternatives. Indeed, at the beginning of the 2010s, a growing number of works started looking into different models that may provide such flexibility. Falling under the general category of *graph representation learning*, these techniques may intersect between different fields, such as graph theory, statistics and machine learning, among others (F. Chen et al., 2020). The general principle is the assumption of a latent space, where nodes are embedded into their lower-dimensional vector representations that can be learned through different algorithms. We can detect different related approaches in the literature. The first, expands the information diffusion paradigm, which models node interactions using node embeddings instead of solely edge probabilities. A different line of work avoids using models of diffusion, and can be decomposed into various approaches, such as random walks, recurrent and graph neural networks, and other more targeted unsupervised learning strategies. This section aims to present these different categories, along with their limitations, exploring whether they could offer the network inference alternatives that we are looking for.

5.2.1 Modeling information diffusion

Some of the earliest graph representation methods expanded the existing diffusion modeling approaches (see Section 3.3.1), with the main goal of accurately predicting whether a node will be "infected" by a cascade or not. Kurashima et

al. (2014) were among the first to learn node embeddings using this approach. They assume that each node has latent coordinates in the visualization space and that diffusion of information is more likely between nodes that are placed closer to each other. To learn the coordinates that best explain the observed cascades, they devise a model based on maximum a posteriori estimation. In the same year, Bourigault et al. (2014) proposed a method that projects nodes in a continuous representation space in such a way that information diffusion can be modeled using a heat diffusion process. They employ a classical stochastic gradient descent method to optimize a diffusion kernel, ensuring that the proximity of nodes in the latent space corresponds to the proximity of their infection times in cascades. However, the primary focus is not on modeling the dynamics of diffusion but rather on accurately estimating which nodes will be infected given a set of initial seeders. In a later work, Bourigault et. al. (2016) proposed a diffusion model that learns probability distributions capable of capturing the hidden influence relationships between users. They assume an Independent Cascade model embeds users into distinct sender and receiver representations and learns them through an EM process. These approaches are pretty similar to the classical methods in the information diffusion literature, with the added flexibility of assuming a latent space for the users that can be learned according to our goals. In a different work, Zhang et al. (2018) observe a gap in previous works, noticing the absence of consideration for communities within domains like OSNs. They proposed an approach to learn embeddings that preserve these community structures from timestamped cascades. Their method assumes a Gaussian mixture model as prior on the users' embeddings, and learns its parameters with an EM algorithm. All in all, we quickly notice that the goal of this line of work is not so much determining directly the network underlying the user interactions, but more so making the correct predictions on the diffusion process itself.

5.2.2 Random walk approaches

While the methods discussed focus on modeling temporal processes using diffusion models, a new branch of techniques was being developed in parallel, using *random walk sampling*. The key idea is that the embeddings of two nodes should become more similar if they frequently appear together in short random walks across a graph (Xia et al., 2019). The pioneering method in this field is *DeepWalk*, drawing inspiration from the rapid advancements in language modeling of that time (Perozzi et al., 2014). As a result, Deepwalk treats random walks on a graph as the equivalent of sentences in a language: each node corresponds to a word,

and a random walk corresponds to a sentence. To learn the node representations, it uses techniques from natural language processing, such as the *SkipGram* model, which maximizes the co-occurrence probability among the words that appear within a specific window in a sentence (Mikolov et al., 2013). DeepWalk has been particularly successful in domains such as OSNs where capturing the local neighborhood structure of nodes is important.

With these methods, we are now able to scale the representation algorithms to graphs of larger sizes, leveraging various random path sampling algorithms to capture both local and global structural information (Sajjad et al., 2019). For example, Tang et al. (2015), designed LINE, a model that can capture both first-order and second-order node proximity information in networks with millions of nodes. A later significant contribution by Grover and Leskovec (2016) introduced *Node2Vec* — while DeepWalk uniformly samples random walks, Node2Vec introduces hyperparameters that enable an interpolation between walks, prioritizing either breadth-first search or depth-first search across the graph. This allows for the exploration of both local neighborhoods and global network structures, providing more informative node embeddings. Concurrently with these works, GraphSAGE was introduced by Hamilton et al. (2017) as an alternative method to integrate both local and global graph structures. It achieves this by employing a sampling strategy that selects and aggregates features from a node's neighbors, both immediate and higher-order ones[2].

Still, these methods were not explicitly designed to target unreliable data settings. Subsequent works began applying random-walk approaches with assumptions of uncertain conditions. For example, Wang et al. (2019) employed a joint optimization problem that maximizes both the likelihood of the observed cascades and a random walk-based objective that regularizes learned representations. They showed that when the network is sparse or relatively inaccurate, the performance of DeepWalk and Node2Vec, drops down remarkably. However, they claim that this is not surprising since both methods assume a deterministic graph structure without explicitly accounting for unreliability in the data. Integrating the dimension of time into these approaches poses a challenge as well. Nguyen et al. (2018) addressed this by devising a framework enabling random walk methods to incorporate temporal dependencies into existing node embeddings. They argue

[2]For an extensive comparison between these works we recommend the paper by Khosla et al. 2019.

that by preserving temporal ordering during learning, inaccuracies or impossible event sequences can be avoided. However, their framework still primarily focuses on optimizing prediction tasks for future links. Shi et al. (2019) proposed a sampling procedure derived from simulating the diffusion structure underlying the network. They view this as a means to augment random walks' ability to capture proximity information. According to their findings, compared to random walk sampling, diffusion processes generate more informative traces and ensure that nodes co-occurring in the same cascades receive similar representations.

The limitations of random walk sampling were discussed in Lyu et al. (2017) highlighting that local node sequences mapped directly into the latent space fail to capture global information. Another drawback of the most popularized random walk approaches like Deepwalk and Node2Vec is their heavy reliance on an existing graph of known relations as input. This prerequisite is problematic in our setting since we assume that the input graph is not always available, or may not accurately represent the true diffusion pathways of the network. Additionally, other research has found that both of these methods, even if they operate on directed random walks, are still ignoring edge directionality in practice (Khosla et al., 2019). This is significant when considering the importance of the notion of direction during the diffusion of information.

5.2.3 Recurrent Neural Networks

The node embedding techniques we examined above employed a shallow embedding approach by optimizing individually the embedding vectors for each node. However, in a time-dependent setting such as information diffusion the history of a node is crucial in understanding a network's dynamics (Lamprier, 2018). As a result, we need an architecture that is capable of incorporating this aspect in its modeling, such as *Recurrent Neural Networks (RNNs)*. They are a class of artificial neural networks (see Section 1.1.5), designed to model long sequences of data, such as time series or natural language (Salehinejad et al., 2017). RNNs include feedback loops where the output from the previous time step is fed back into the network as input for the current time step. This way, RNNs can maintain a hidden state that captures information from the past. They are therefore particularly useful for tasks where the input data is sequential and the past history is important, as is the case with the interactions observed on OSNs.

Various works have attempted this approach for diffusion prediction tasks. For example, the paper by Cao et al. (2017), encoded the dynamics of cascades combining Hawkes processes and RNNs, taking advantage of both generative processes and DL techniques to predict retweet cascades. In our case, when modeling the information propagation process, it is important to consider the impact of the underlying network topology on propagation. In a paper by Lamprier (2018) the challenge of not having available the topology of diffusion during learning is emphasized. A work that tackled this is by Wang et al. (2017), where they assume that a cascade is not only a sequence of nodes ordered by their respective timestamps, but contains a richer structure that guides the diffusion process over a network. To model this, they incorporate possible diffusion topologies in the hidden infected node states of an RNN architecture. However, this work ignores the past trajectory of a cascade; in response, Lamprier (2018), proposes a variational model that incorporates it. It is assumed that the diffusion probabilities between nodes depend on a latent space encapsulating the nodes' past trajectory of the diffused content. This can be incorporated into an RNN mechanism that models the process of diffusion. Trajectory distributions can then be inferred from the observed infections, creating an iterative learning process that estimates node-node infection probabilities. One of the targets of this work is to assess the model's ability to pinpoint the true infectors in the observed diffusion episodes, which is very similar to what we are trying to achieve. However, on top of not taking explicit consideration for the hidden paths or the erroneous information between the nodes, the model is limited to an evaluation on a synthetic dataset where the infectors are known.

Attention mechanism. To enhance prediction results, various models integrate additional types of information when estimating edge probabilities, such as user-user similarities (Feng et al., 2018), and other forms of microscopic or macroscopic information (Cheng Yang et al., 2021). Instead of manually engineering these features, an *attention mechanism* can be embedded within the RNN architecture. This mechanism guides the model to focus on important structural or spatial properties of the data. For example, Wang et al. (2017) developed a model that uses an attention mechanism in RNNs to estimate the conditional probabilities of subsequent resharing actions. Additionally, Wang et al. (2018) proposed a structural attention mechanism that merges users' diffusion contexts, and, more recently, Zhihao et al. (2021) introduced an attention mechanism that captures the spatial relationships among social network users. However, as was the case

in the previous categories, the focus of these works, rather than pinpointing the structures underlying the diffusion, is generating representations that can predict the next user in an information spread as accurately as possible. In general, while RNNs can indeed capture long-range time dependencies, overcoming the primary weakness of Markov models (Lipton et al., 2015), an important limitation arises in the context of graph data: Two nodes that are nearby in terms of graph topology may be far apart in the sequential node ordering. This results in the so-called long-term bottleneck (Liao et al., 2019).

5.2.4 Graph Neural Networks

In response to the RNNs bottleneck, Graph Neural Networks (GNNs) present a compelling alternative for representing complex relational structures in OSNs, including the way that information is diffusing. Unlike RNNs, which operate sequentially, GNNs operate directly on the graph topology, allowing for simultaneous consideration of node attributes and graph structure (Scarselli et al., 2008). Graph convolutional networks (GCNs) are the most standard and widely used type of GNN models — they bridge the gap between spectral graph theory whose representation potentials are often limited, and the more recent advancements of graph theory (T. N. Kipf and Welling, 2016a) (see next section for more).

An example that combines GNNs and the notion of information diffusion is the model DeepInf by Qiu et al. (2018). They incorporate both network structure with a GNN model and user-specific features via an attention mechanism, to predict a user's future infection status. Other works, instead of focusing on individual user embeddings, focus on entire subgraphs of cascades (X. Chen et al., 2019; Q. Zhao et al., 2022). For example, the model by Chen et al. (2019) first samples representations of sub-cascade networks and then trains these using graph convolutions for a prediction task. However, an overemphasis on subgraphs may lead to poor model interpretability (B. Zhou et al., 2024).

Bayesian deep learning. When it comes to incorporating uncertainty in graph-like or other data, a notable recent line of work involves the integration of the established theory of Bayesian modeling with DL architectures. For example, Zhang et al. (2019) have proposed a Bayesian framework for GCNs, viewing observed graphs as samples from a random graph family. By inferring the joint posterior of graph parameters and node labels using assortative mixed-membership stochastic block models, their approach enhances performance, particularly with limited labeled data. In a different context, Ryu et al (2019) devised a Bayesian

GCN to deal with uncertain molecular data and found that Bayesian inference can be indeed reliable in making predictions. This verifies the results of Kendall and Gal (2017), who were among the first to claim that uncertainty in AI tasks such as computer vision can be addressed with novel Bayesian deep learning tools. Their Bayesian deep learning framework emphasizes the distinction between *aleatoric* uncertainties, which are data-driven and *epistemic* uncertainty, which captures our ignorance about the model that generated the available data. Other works have followed in the context of graphs, proposing different Bayesian graph neural network architectures (Hasanzadeh et al., 2020). Yet again, these methods prioritize the improvement of the prediction's accuracy over utilizing uncertainty estimates to address the unreliability of the data itself.

Graph autoencoders. A category of models that seems to be concurrent to the Bayesian approach, is that of graph autoencoders (VGAEs), a special class of unsupervised neural networks tailored to extract latent graph representations (T. N. Kipf and Welling, 2016b; T. Kipf et al., 2018). Unlike traditional autoencoders, VGAEs encode the topology and features of a graph over a latent probabilistic space and then decode it back to reconstruct the original graph. They have gained traction for their ability to capture complex graph structures for all kinds of ML tasks without the explicit need for labels. For example, in our context of information diffusion, the model by Sankar et al. (2020) has proposed a variational autoencoder coupled with GNNs to model social homophily and temporal influence for a diffusion prediction task. Most notably, Elinas et al. (2020) provided a more general VGAE framework for GCNs, solving a gap in the graph machine learning literature that considers networks as noiseless. As a solution, they designed a mechanism that explicitly deals with the absence of predefined input graphs or with noisy/adversarially perturbed structures. They proposed a joint probabilistic model that incorporates a prior distribution over graphs and a GCN-based likelihood, employing stochastic variational inference to estimate graph posteriors on top of the GCN parameters. Their approach seems to outperform existing Bayesian and non-Bayesian GCN algorithms, particularly in semi-supervised classification tasks. Unfortunately, although they do make direct posterior estimates, they do not take advantage of them to make inferences about the network, focusing instead on the accuracy of downstream prediction tasks.

Combinatorial optimization for GNNs. A very recent and promising line of research is aiming to solve graph combinatorial optimization problems with GNNs, instead of powerful solvers like Gurobi. According to Karalias and Loukas (2020), this is a task that is very challenging for GNNs, especially in the absence of labeled instances. As a solution, they proposed an unsupervised learning framework for solving the maximum clique problem, training a GNN to identify distributions of solutions that provably abide by the constraints of the combinatorial problem. They show that their approach is able to obtain valid solutions, remaining competitive with Gurobi in terms of accuracy, while also being faster. However, existing GNN architectures might not be able yet to detect important structural patterns in the data for these purposes, while more expressive approaches may lack scalability when handling large-scale inputs. A very recent article provides an extensive review of these key limitations, along with the recent advancements and promising avenues for this emerging field (Cappart et al., 2023).

5.2.5 Focusing on contrastive learning

Up to this point, the majority of the studies we have examined for inferring/predicting edges have operated under the assumption that either some labels are available or that the input structure is an accurate portrayal of its genuine dynamics, disregarding potential data unreliability. However, if we take into account the intrinsic uncertainty surrounding edge existence and equate NI to the problem of link prediction with limited or no labels, we should focus on the techniques that consider these settings as the default. As a result, we are now targeting fields like *unsupervised learning*, and *self-supervised learning (SSL)* techniques which operate independently of labels (Y. Liu et al., 2022; Ghahramani, 2003). Particularly in the last years, the field of SSL has been standing out as a novel field with promising potential (Jin et al., 2020). It is a framework that leverages the inherent patterns and relationships that are present within the input data itself, enabling the creation of meaningful node representations. Intuitively, these could potentially capture the existence of edges as well.

A special category of SSL techniques is *contrastive learning*, which leverages latent information already existent in the data without the need for any labels. The general concept is learning representations of unstructured information by contrasting similar pairs of data points with dissimilar ones (Hadsell et al., 2006; Van Den Oord et al., 2018; Bachman et al., 2019; T. Chen et al., 2020). The basic contrastive framework consists of selecting data points as samples

of three different types: the *anchor*, which is a random data point from the dataset; its *positive* samples, data points belonging to the same type or distribution as the anchor; and its *negative* samples, data points belonging to a different distribution. A model is then trained to maximize the similarity in the learned latent space between the representations of the anchor and the positive samples, while simultaneously maximizing the distance (or minimizing the similarity) between the anchor and the negative samples. Contrastive learning originally enabled the success of computer vision tasks, serving as a way to learn on a large scale how to differentiate between different classes of visual images without the need for labeling which might be expensive to get (T. Chen et al., 2020; Le-Khac et al., 2020).

Contrastive learning and graphs. However, in the domain of graphs, contrastive learning is a subject that remains relatively underexplored. Only recently, we have noticed a growing trend in extending contrastive learning methods to graph data using GNNs (Xie et al., 2023). Many such methods are focused on learning representations of entire graph instances and are based on the following principle: For each graph, we first generate different *views* or *transformations*. Two samples from the views generated from the same graph instance stand as a *positive* pair and two samples from the views generated from different instances stand as a *negative* pair. Their representations can then be learned with the help of a GNN model that is guided by a contrastive loss, and can be used later for downstream tasks, more commonly graph classification (Xie et al., 2023). Contrastive learning in a link prediction setting is less straightforward: the objective is to infer edges between nodes, without having access to any ground truth labels during training (Jaiswal et al., 2020). Additionally, generating views or transformations, as explained earlier, might be more complicated on an edge level, especially considering the uncertainty associated with the presence of edges.

An alternative approach could be to leverage the concept of positive/negative pairs with a different strategy, focusing on a node level instead. Intuitively, in the context of our feasible network inference problem, this may involve sampling nodes in a way that *indirectly* captures the presence of edges in the underlying graph, while also accounting for the temporal aspect of the observed interactions.

In the present thesis: accounting for feasibility

After reviewing the literature, we quickly realize that while many papers model the way information diffuses online, most of them emphasize increasing prediction accuracy for a specific task, rather than uncovering the hidden structures underlying the interactions. As a result, for these works, when prediction results are satisfactory, there is little interest in explicitly modeling the unreliability of the data. In this thesis, we aim to bridge this gap by focusing on what happens one step before the final task: What can the learned node representations of now standardized models like Node2Vec tell us about the underlying interactions in the context of OSNs? Can they guide us toward structures that are feasible, and if not, how can we devise a learning method that guides us toward representations that account for feasibility without relying on labels? The rest of this chapter is focused on exploring these questions.

5.3 Methods for network inference

Building on these questions, we aim to devise an alternative to our previously proposed CEM method, targeting a representation learning method that can infer the network underlying a set of node-to-node interactions while also incorporating the notion of feasibility. First, we need to introduce the key methods and functions that will be employed, requiring careful consideration of various aspects. Following the general graph representation learning framework proposed by Hamilton (2020), we will utilize an Encoder model which model assumes that nodes can be encoded into meaningful representations optimized through an iterative learning process. For the encoding part, many models can be used, but GNNs are the most commonly preferred. Consequently, in this section, we present the Encoder model that can be tailored to our task, along with an introduction to GNNs, their functions, and the critical factors to consider when selecting them as encoding models.

5.3.1 An Encoder model for link prediction

Why would an Encoder model be needed for inferring meaningful edges between nodes? If we consider the standard link prediction framework, one could simply leverage knowledge about the nodes and edges of a graph that is readily available. In the domain of OSNs, for example, we might have information on the characteristics of the nodes (e.g., political affiliation, gender, country of origin,

etc), also known as *node features*. Based on these, we could infer the presence or absence of an edge between nodes by quantifying the similarity between them. The question that arises is what kind of similarity measure is the most appropriate to achieve this. While we could manually test different measures (see Box 2.3.1), this approach quickly becomes inflexible and time-consuming. On top of that, the availability of reliable labels is not always guaranteed and collecting them is a very strenuous task to achieve (Northcutt et al., 2021). Instead, we can *indirectly* learn how similar they are via graph representation learning techniques. One general framework that is very commonly used in these cases is the *Encoder model*. Rather than modeling information diffusion through explicit probability distributions and diffusion models, this approach involves *embedding* the entities of a network into a *latent space*, capturing indirectly the underlying features and relationships that influence the formation of links (Kurashima et al., 2014). We should note here that in this section we give the general framework, without going into the details of how it can be adapted to our problem, feasible network inference.

The Encoder. Encoding is the process of transforming the raw input dataset (in this case, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and node features) into an embedding of a latent dimension d. Formally, the Encoder is defined as the mapping of nodes $i \in \mathcal{V}$ to their latent embeddings $\mathbf{z}_i \in \mathbb{R}^d$:

$$\text{ENC} : \mathcal{V} \rightarrow \mathbb{R}^d.$$

An Encoder can either be *shallow*, relying simply on each node's unique ID to map it to an embedding, or may include richer information, such as the node features. Overall, it is important that the Encoder preserves a semantic meaning, ensuring that similar inputs are mapped to nearby points in the embedding space according to a given definition of proximity.

Learning the representations. In most traditional applications, we operate within a supervised learning setting, meaning that we possess a dataset of *training data* $\mathcal{D}$ where the edges between nodes are known in advance. However, in the case of network inference, where we have no labels, the problem can be regarded as a self-supervised or contrastive learning one. In this case, the model can be trained on the whole dataset or informative parts of it with different sampling methods (Jaiswal et al., 2020). After sampling the data, the primary objective is to train the Encoder so that the pairwise node connections of the training set $\mathcal{D}$

are effectively reconstructed. To achieve this, the most common approach is to minimize a reconstruction loss $\mathcal{L}$ over $\mathcal{D}$:

$$\mathcal{L} = \sum_{(i,j)\in\mathcal{D}} \ell\left(\mathbf{z}_i, \mathbf{z}_j\right),$$

where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function that estimates the similarity between the learned representations of $\mathbf{z}_i, \mathbf{z}_j$ that we are trying to learn. This general learning architecture allows the model to automatically learn the respective parameters of the encoding mechanism in a way that resembles the existence of important connections.

5.3.2 Graph Neural Networks as better encoders

As we saw in Section 5.2, the simplest way to encode nodes into their latent representations is to treat them as shallow individual vectors that can be optimized via the above Encoder architecture (Hamilton, 2020). However, this way we may be missing valuable information that exists in the structure of the graph and that is not taken into account during the encoding. This can be alleviated instead with the use of GNNs, a general ML framework explicitly designed to capture the relational information encoded in graph-structured data, as we introduced it in Section 5.2.4 (Gori et al., 2005; Scarselli et al., 2008). Their key concept is that they can learn node representations by aggregating information from their neighboring nodes, thereby encoding the local graph structure on top of any node features that may be available (Hamilton, 2020).

Message passing updates. The most fundamental mechanism of a GNN is that it uses a form of *message passing* between nodes that is updated iteratively via an architecture composed of several *artificial neural network layers* (Scarselli et al., 2008). This update is expressed by the following equation, which is called the *message-passing function*:

$$\begin{aligned}\mathbf{h}_i^{(l+1)} &= \mathrm{UPDATE}^{(l)}\left(\mathbf{h}_i^{(l)}, \mathrm{AGGREGATE}^{(l)}\left(\{\mathbf{h}_j^{(l)}, \forall j \in \mathcal{N}(i)\}\right)\right) \\ &= \mathrm{UPDATE}^{(l)}\left(\mathbf{h}_i^{(l)}, m_{\mathcal{N}(i)}^{(l)}\right),\end{aligned}$$

where:

- $\mathbf{h}_j^{(l)}$ is the representation, or hidden embedding, of node i at layer l.
- $\mathcal{N}(i)$ is the set of neighboring nodes of node i.

– $\text{AGGREGATE}^{(l)}$ is a permutation invariant function that combines the representations of neighboring nodes, and potentially the nodes themselves, if self-loops are included.

– $m_{\mathcal{N}(i)}$ is the message that is aggregated from i's neighborhood $\mathcal{N}(i)$.

– $\text{UPDATE}^{(l)}$ is an arbitrary function responsible for updating the representation of a node i at layer l based on its current representation $\mathbf{h}^{(l)}$ and the aggregated messages from its neighboring nodes.

Graph convolutional networks (GCNs) are one of the most popular baseline GNNs, introduced by Kipf and Welling (2016). They suggest a simple, symmetric-normalized aggregation function, that includes self-loops. Following the same format as presented above, the message-passing function is defined as:

$$\mathbf{h}_i^{(l)} = \sigma\left(\mathbf{W}^{(l)} \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{\mathbf{h}_j}{\sqrt{|\mathcal{N}(i)||\mathcal{N}(j)|}}\right),$$

where σ denotes an element-wise non-linear function, such as the cosine function (Luo et al., 2018) or ReLU (Eckle and Schmidt-Hieber, 2019), and $\mathbf{W}$ is a trainable parameter matrix.

An intuitive explanation. The core idea behind GNNs is the concept of message passing, where nodes exchange information with their neighbors to update their own representations. By iteratively passing and aggregating messages through the nodes, GNNs can capture the relational dependencies and structural information encoded in the graph, enabling them to learn rich and informative representations of nodes. GNNs leverage both local and global information present in the graph. At each layer, nodes aggregate information from their immediate neighbors, allowing them to capture local structural patterns. For example, according to the message-passing function, after the first layer $(l = 1)$, every node embedding incorporates information from its 1-hop neighborhood (the neighbors which can be reached by a path of length 1 in the graph); after the second iteration $(l = 2)$ every node embedding incorporates information from its 2-hop neighborhood, and so on. As messages propagate through the graph, nodes accumulate information from distant parts, enabling them to capture more global graph properties.

Parameters of a GNN model. The strength of a GNN model is its flexibility to adapt to diverse datasets and tasks by changing its different parameters and

functions. They define the GNN's overall architecture and behavior and can be tuned during training to optimize predicting performance. The flexibility of the GNNs lies in its design (You et al., 2020), which gives the ability to the user to define and tune an important set of parameters such as: (i) the number of layers, which allows the model to capture more complex structures while controlling overfitting (Ying, 2019); (ii) the dimensionality of hidden representations, affecting the model's expressive power and computational complexity; (iii) the activation function (such as ReLU and cosine) which introduces non-linearity to capture complex patterns (Sharma et al., 2017; Kulathunga et al., 2020); and (v) the learning rate, which impacts the convergence speed of the training process.

5.4 Proposing a simple contrastive model

After establishing this general perspective on representation learning, we now formalize the setting in which we aim to offer a more adaptable alternative to our network inference method, CEM. As in the previous chapter, we stay in the context of OSNs and assume that we have a dataset of interactions among a set $\mathcal{U}$ of N user nodes expressed as a set $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_s, \dots\}$ of diffusion episodes (e.g., tweets), where a node i may have interacted with any $\mathcal{D}_s \in \mathcal{D}$ episode at a timestamps $t^s(i)$ by posting or reposting it. Our goal is to learn the latent representations $\mathbf{z} \in \mathbb{R}^d$ of users, according to a contrastive learning technique that takes into account the temporal order of the observed interactions. Despite the apparent simplicity of this requirement, it is still far from common practice to explicitly take temporal dynamics into account in a contrastive learning framework.

5.4.1 Model architecture

An important aspect of our approach is to assume that users can take the roles of senders or receivers of information, which, in a graph, is usually reflected in the direction of edges. However, the latent space of node embeddings lacks an inherent notion of direction, making it non-trivial to capture during the learning process. Our intuition is that a user's latent representation can differ significantly based on whether they send or receive information. The same intuition was given by an early work in representation learning by Bourigault et al. (2014), providing two distinct user representations for their sending and receiving behavior. In the domain of contrastive learning, this idea has been implemented in a broader context, rather than being specifically tailored to the temporal direction of information flow. For example, a recent model provides two different embeddings

to distinguish whether the node pairs are connected from a feature or structure perspective (J. Zhao et al., 2023). More formally, in our case, we introduce two distinct types of embeddings $\mathbf{z}_i^s, \mathbf{z}_i^r \in \mathbb{R}^d$, which can capture a user's sending and receiving behavior respectively in a decoupled fashion. Our intuition is that this will allow for a more explicit representation of the two roles that a user may take in a diffusion process and can potentially improve the quality of the learned embeddings.

The encoder. A neural encoder will be trained to extract the user-level representations $\mathbf{z}^s, \mathbf{z}^r \in \mathbb{R}^d$ for their sender and receiver behavior respectively. Each row in these two matrices corresponds to a different user $i \in \mathcal{U}$. As we explained in Section 5.3, this encoder is quite flexible and can be any kind of GNN such as GCN (T. N. Kipf and Welling, 2016a), GAT (Veličković et al., 2017), GraphSAGE (2017), and many others (J. Zhou et al., 2020).

5.4.2 Model training

In order to learn the latent user representations in a contrastive setting, we have to define our own contrastive learning sampling process that does not rely on explicit labels. Towards that goal, an idea is to devise a temporally feasible set of sample pairs capable of capturing the potential interactions within the dataset. In this Chapter the term "feasible" takes on a more loose interpretation, indicating that the sampled pairs should be created according to the temporal ordering of the observed interactions instead of a strict set of constraints as in the case of CEM. Our strategy unfolds as follows: First, we select an *anchor* user (a) that acts as the reference point for constructing the pair. Next, we randomly choose a diffusion episode $\mathcal{D}_s$ from the dataset $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_s, \ldots\}$ with which the anchor user has interacted by posting or reposting it. Then, we sample *positive* and *negative* user samples according to the following:

- *Positive samples* represent the potential recipients of a diffusion episode $\mathcal{D}_s \in \mathcal{D}$ (e.g., tweet) directly from the anchor a. To capture the feasible ordering of time, we select as positive sample a user j that has interacted with $\mathcal{D}_s$ at a *later* timestamp than the anchor user, $t^s(j) > t^s(a)$.

- *Negative samples* correspond to users that are unlikely to receive a diffusion episode from the anchor user a. This can occur either because they have not interacted with $\mathcal{D}_s$ at all, $t^s(j) = \infty$, or because their interaction took place at a non-feasible time, i.e. $t^s(j) < t^s(a)$. Recognizing this distinction

allows us to develop a more sophisticated selection strategy for negative samples, compared to selecting them in a random, time-agnostic fashion.

This is a similar sampling approach to the SimGRL model by Huang et al (2023), with the difference that in their case they depend on pre-known positive/negative labels whereas we establish our own nuanced definition of these labels based on the concept of temporal feasibility. In Algorithm 2, we describe the details of the sampling process for each anchor user. The final contrastive pairs of users that are sampled are structured as (anchor, positive, negative). In practice, we only choose a predefined number of positive and negative samples for each anchor, that can be tuned by the user. Since the underlying graph is known to be sparse (meaning few positive edges), we limit the positive user samples of each user to only one, and we experiment with different sizes of negative samples. If we had both positive and negative samples of size one, we would be dealing with *triplets*; however, this is mostly preferred in the domain of computer vision tasks, and in our case it would be less informative (Weinberger and Saul, 2009; Yingying Zhang et al., 2019). By employing this simple sampling strategy, our aim is to equip the model with feasible sample pairs that take into account the temporal ordering of observed interactions. We believe that this approach can allow the model to capture meaningful patterns in the data via a training process that can be more easily interpreted.

5.4.3 Contrastive loss

Let $\mathbf{z}_i$ represent the embedding of a user i that is unknown and has to be learned in a contrastive fashion. A *triplet loss* could then be computed for each pair of anchor (a), positive (p), and negative (n) samples. The general principle is to enforce the similarities between the embeddings of the anchor and its positive samples to be higher than the similarities between the anchor and its negative samples, by at least a margin m. The function used to measure this similarity is denoted as $sim(\cdot, \cdot)$ and the contrastive objective function, originally proposed in the context of face recognition by Schroff et al. (2015), is defined as:

$$\mathcal{L} = \sum_{(a,p,n)\in\mathcal{T}} \max(0, sim(\mathbf{z}_a, \mathbf{z}_p) - sim(\mathbf{z}_a, \mathbf{z}_n) + m),$$

where $\mathcal{T}$ is the set of all sampled pairs of users. In the context of our problem, where we use different embeddings for the sending and receiving behavior of users it would become:

$$\mathcal{L} = \sum_{(a,p,n)\in\mathcal{T}} \max(0, sim(\mathbf{z}_a^s, \mathbf{z}_p^r) - sim(\mathbf{z}_a^s, \mathbf{z}_n^r) + m),$$

With this simple change in formula, we are using instead the similarity between each user's *sending* behavior vector $\mathbf{z}_i^s$ and every other user's *receiving* behavior vector $\mathbf{z}_j^r$. This step is crucial, since it incorporates a notion of "direction" in the way that information diffuses from user to user, from the "sender" of an item to its potential timely feasible "receiver". However, since we do not only want to focus on triplets, but on multiple samples of neighbors at a time, we can use the following formula, similar to the InfoNCE loss (Van Den Oord et al., 2018):

$$\mathcal{L}_{FeasCL} = -\log \frac{e^{sim(\mathbf{z}_a^s, \mathbf{z}_p^r)}}{e^{sim(\mathbf{z}_a^s, \mathbf{z}_p^r)} + \sum_n e^{sim(\mathbf{z}_a^s, \mathbf{z}_n^r)}}. \tag{5.1}$$

We call this the *Feasible Contrastive Loss (FeasCL)*. In some cases, the exponents can be divided by a temperature τ to control the impact of the similarity function (X. Liu et al., 2021; Jovanović et al., 2021):

$$\mathcal{L}_{FeasCL} = -\log \frac{e^{sim(\mathbf{z}_a^s, \mathbf{z}_p^r)/\tau}}{e^{sim(\mathbf{z}_a^s, \mathbf{z}_p^r)/\tau} + \sum_n e^{sim(\mathbf{z}_a^s, \mathbf{z}_n^r)/\tau}}, \tag{5.2}$$

where we choose the cosine as a measure of similarity:

$$sim(\mathbf{z}_a^s, \mathbf{z}_p^r) = \frac{\mathbf{z}_a^s \cdot \mathbf{z}_p^r}{\|\mathbf{z}_a^s\| \|\mathbf{z}_p^r\|}. \tag{5.3}$$

The main idea is that the node embeddings of users who could be feasibly connected in the underlying network should also be moved closer together in the latent space. By using this *contrastive loss* we can guide the similarities between the *sender* embeddings of the anchor and the *receiver* embeddings of its timely feasible (positive) samples, to be higher than the similarities with the receiver embeddings of its timely non-feasible (negative) samples in the latent space. Notice again how in the loss we only choose the sender embeddings for the anchor samples, while keeping only the receiver embedding for the positive and negative samples. The intuition behind this choice is to give a nuance of direction in the latent space during the process of learning.

Mini-batching. The optimization of DL architectures is typically based on *stochastic gradient* methods (Robbins and Monro, 1951; Bottou et al., 2018). The main principle is the use of stochastic gradient descent update rules that provide an approximation of the expected value of the gradient of the loss function over the training data (Masters and Luschi, 2018). This stochastic approximation is based on using small subsets of training examples, known as *mini-batches* (Masters and Luschi, 2018). While mini-batching is more commonly associated with tasks on large-scale datasets such as in image classification or natural language processing, it can also be beneficial for training on graphs (Berg et al., 2017). In our case therefore we can train the model to learn the encodings $\{\mathbf{z}_i^s, \mathbf{z}_i^r\}_{i=1}^N$ by using mini-batches of nodes, reducing the memory footprint required to store the contributions to the loss function, which can be particularly important for graphs of larger sizes (Berg et al., 2017).

5.5 Experimental evaluation

5.5.1 Environment

The dataset

In the preceding chapter, we discussed how since its 2017 creation, the Twitter dataset #Élysée2017fr has most certainly undergone important changes, such as profile deletions and users unfollowing each other (see Section 4.8.5). As we explained, it is also common for users to retweet content beyond their immediate followees, utilizing features like the search function, the platform's recommendation algorithm, or trending topics. Consequently, the #Élysée2017fr reposting dataset itself may not be entirely explained by the ground truth friendship graph that we had collected (as we showed, the latter can only explain 49% of the diffusion episodes in their entirety). In order to get a clearer view of the performance of our method, as a pre-processing step, we opt to keep only the subset of users and episodes that are adequately explained by the #Élysée2017fr friendship graph. The way we achieve this is as follows: we look into each diffusion episode $\mathcal{D}_s \in \mathcal{D}$, and if there is no feasible subgraph (as defined in Section 4.2) that connects all the users in $\mathcal{D}_s$, we remove $\mathcal{D}_s$ from the episodes set. In the end, we only keep the subset of users that are participating in the remaining episodes.

As a result, instead of having 293,405 episodes as before, we now have 144,264. For completeness, Tables 5.1 and 5.2 show in detail the differences between the original (non-feasible) dataset and the final feasible one for both the friendship

and the retweet network. By comparing against a less noisy ground truth that adequately explains the provided dataset, it will be easier to evaluate whether the model can indeed infer edges that exist in the ground truth friendship graph. The most substantial difference is in the number of edges connecting the users, which directly affects the graph's density. Interestingly, the decrease in the size of the most strongly connected component (max(scc)) of the friendship network is relatively smaller. This suggests that, despite a reduction in overall connectivity, there remains a core group of users who are strongly connected to each other.

Learning process

For the encoding part of the training, we choose a GNN architecture and a graph structure $\mathcal{G}$ to provide as input. Since we assume that the underlying friendship graph is unknown during training, we construct a baseline graph structure $\mathcal{G}$ from the available reposting interactions, by drawing an edge (i, j) from i to j if $t^s(i) < t^s(j)$ for at least one diffusion episode $\mathcal{D}_s \in \mathcal{D}$. This represents the fact that it is possible for the information to have passed through the edge. Although unrealistic, this graph allows for the nodes to incorporate information from a wider set of possible neighbors. We opt for the GraphSAGE model as the core GNN architecture (Hamilton et al., 2017) due to its stochastic sampling of node neighborhoods (see Section 5.2.4). Since we do not assume to have any node features available, we give randomly initialized vectors as features, which surprisingly, has not been found to not affect the performance of GNNs (Abboud et al., 2020). We train the FeasCL loss on one NVIDIA GPU, with a batch size of 640 and a temperature $\tau = 10$, on well-known *Adam optimizer* (Kingma and Ba, 2014) with a learning rate of 0.01. It is widely used in DL as an efficient way to optimize the parameters of neural networks, leading to faster convergence and improved performance in a more flexible fashion compared to traditional optimization algorithms. As is usually the case in a DL context, our model involves many parameters that can be tweaked to optimize its performance (You et al., 2020). We list all of them in more detail in Appendix B.

Hard negative sampling

A key factor we identified for optimizing FeasCL is an additional *hard sampling* that we should perform from the set of negative samples after the first iteration. Specifically, instead of sampling negative users as usual, most of whom might already be dissimilar to the anchor and thus contribute little to the triplet loss, we further refine the sampling process. This is achieved by looking into the learned

Tab. 5.1. Original (non-feasible) #Élysée2017fr dataset.

Source	Users	Edges	Density	Average degree	#scc	max(scc)	intra	inter
Friendship network	11,404	1,555,718	0.0120	136.42	641	10,747	0.84	0.16
Retweet network	11,404	6,922,990	0.0523	601.42	527	10,984	0.77	0.23

Tab. 5.2. Feasible #Élysée2017fr dataset.

Source	Users	Edges	Density	Average degree	#scc	max(scc)	intra	inter
Friendship network	8,437	260,247	0.0004	30.85	883	7,479	0.95	0.05
Retweet network	8,437	470,539	0.0066	55.77	1,712	6,712	0.92	0.07

representations at the end of each epoch and selecting only those negative samples (a predefined number of them) that are still the most similar to the anchor. These hard negatives stand as sample points that are the most difficult to distinguish from an anchor point. It is a strategy that has been recently proven beneficial for contrastive learning tasks (Robinson et al., 2020). Moving these negative samples farther away will have a stronger impact on the loss, helping to correct the semantic "mistakes" more quickly (Schroff et al., 2015; Oh Song et al., 2016). In our case, a "mistake" refers to a negative user who could not have possibly received a piece of information from the anchor (meaning that no path should exist between them) due to their time ordering of interactions not justifying it in the reposting dataset provided as input. In practice, this either means that the anchor has reposted something *after* the negative sample, and therefore it is not possible for them to have passed on the information to the negative user, or that they have not appeared in the same diffusion episode at all.

Evaluation metric

At the end of the training, our model provides a set of latent user representations $\mathbf{z}^s, \mathbf{z}^r$ that we need to verify whether they correspond to any real underlying friendship connections. Given that we do not explicitly receive graph edges as output, we can indirectly examine the k most similar neighbors j for each user i in terms of the chosen similarity metric $sim(\mathbf{z}_i^s, \mathbf{z}_j^r)$. In accordance with the optimization process, we keep the use of the cosine similarity, and examine the mean Precision results, compared to the ground truth friendship network, and across all users, for different values of k.

Comparison

We compare our results with two methods: our previous approach, CEM, and Node2Vec (Grover and Leskovec, 2016), which is very often employed in link prediction tasks (H. Wu et al., 2022). Node2Vec generates node embeddings through unsupervised learning using random walk sampling (refer to Section 5.2.2 for details). It is important to note that the aim of this comparison is not to provide an exhaustive evaluation against the various other existing graph representation learning methods that could have been applied. Rather, our primary goal is to compare first the results to CEM, exploring whether a simple and intuitive contrastive approach can indeed provide node representations that incorporate feasibility for the problem of network inference. This can potentially serve as a promising and more scalable baseline for future exploration.

5.5.2 Results

In this section, we train FeasCL for a different number of epochs (500/1,000/5,000) according to the methods described above and report the results in Table 5.3. As explained before, for each sending user in the user set $\mathcal{U}$, we examine the mean Precision in selecting their k most similar receiving neighbors in terms of their embeddings. We notice that our initial method, CEM, still performs the best in this feasible dataset, achieving a Precision of almost 77% on average for the $k = 1$ closest neighbor of users. However, FeasCL provides a relatively close Precision of 71.25%, which is competitive. However, we notice that as the number of k increases, the performance gap between CEM and FeasCL narrows. For instance, at the average node degree in the friendship graph, which is 30 (see Table 5.1), the mean Precision of CEM is 29.87%, while FeasCL achieves 27.52%. Additionally,

Tab. 5.3. k-nearest neighbors comparison (in %)*.

Mean Prec@k	@1	@2	@3	@5	@10	@15	@20	@30	@40	@50	@100
CEM-ER	**76.32**	67.94	62.20	54.78	44.48	38.63	34.56	29.13	25.53	22.89	15.62
CEM-SBM	**76.91**	68.95	63.58	56.11	45.61	39.59	35.41	29.87	26.16	23.46	15.93
Node2Vec	0	13.21	14.83	14.11	11.46	9.64	8.43	6.9	5.97	5.36	3.95
FeasCL-5K	71.25	63.76	58.61	51.89	42.01	36.31	32.50	27.52	24.32	22.03	15.70
FeasCL-1K	68.14	58.89	53.54	46.88	36.94	31.87	28.65	24.49	21.93	20.07	14.55
FeasCL-500	66.06	57.07	51.95	44.82	35.43	30.64	27.62	23.86	21.32	19.57	14.24
FeasCL-5K-40	61.15	52.10	48.24	42.17	33.55	29.16	26.24	22.71	20.38	18.68	13.68
FeasCL-5K-60	62.14	53.31	49.07	42.79	34.15	29.52	26.64	23.14	20.74	18.98	13.86
FeasCL-5K-125	69.35	60.15	54.86	48.13	38.62	33.54	30.17	25.84	23.03	20.99	15.28

* *-5K, -1K, -500 is the number of epochs, -40, -60, -125 is the number of dimensions for the output*

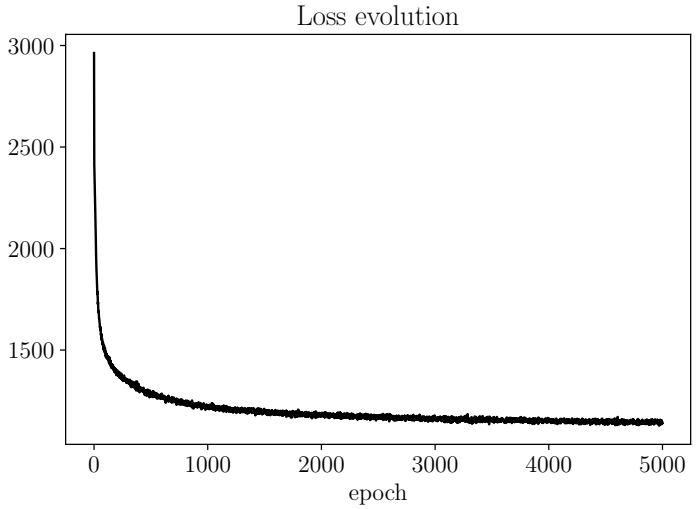


Fig. 5.1. – *Loss evolution of FeasCL-5K per epoch.*

we notice that FeasCL is outperforming significantly the baseline Node2Vec results, which exhibit very low performance. The most important takeaway is that a simple random walk strategy on a noisy graph does not adequately capture its underlying interactions. This outcome is less surprising when we consider that Node2Vec has not been explicitly designed to handle such unreliable cases. This result underscores the importance of making a clear distinction between the data and the graph that underlies it. It also shows the ability of a simple contrastive strategy to make this distinction possible, without the need for any labels or node features, relying only on the temporal information provided by the data.

Regarding the impact of hidden dimensions, increasing their size allows the model to capture more complex patterns and nuanced relationships within the data. Specifically, as shown in Table 5.3, Precision improves from 61.15% to 62.14% and then to 69.35% as we increase the output node representations' dimension from 40 to 60 and then to 125. Higher-dimensional representations provide the model with a greater capacity to move the user representations around closer or farther away in the latent space, which is beneficial when subtle differences in latent node relationships are crucial.

We should note that the results we report for FeasCL represent the best outcomes from various parameter combinations we tried. A critical point that we identified experimentally is the improvement in Precision levels as we increase both the number of epochs and the size of the hidden dimensions. As shown in Table

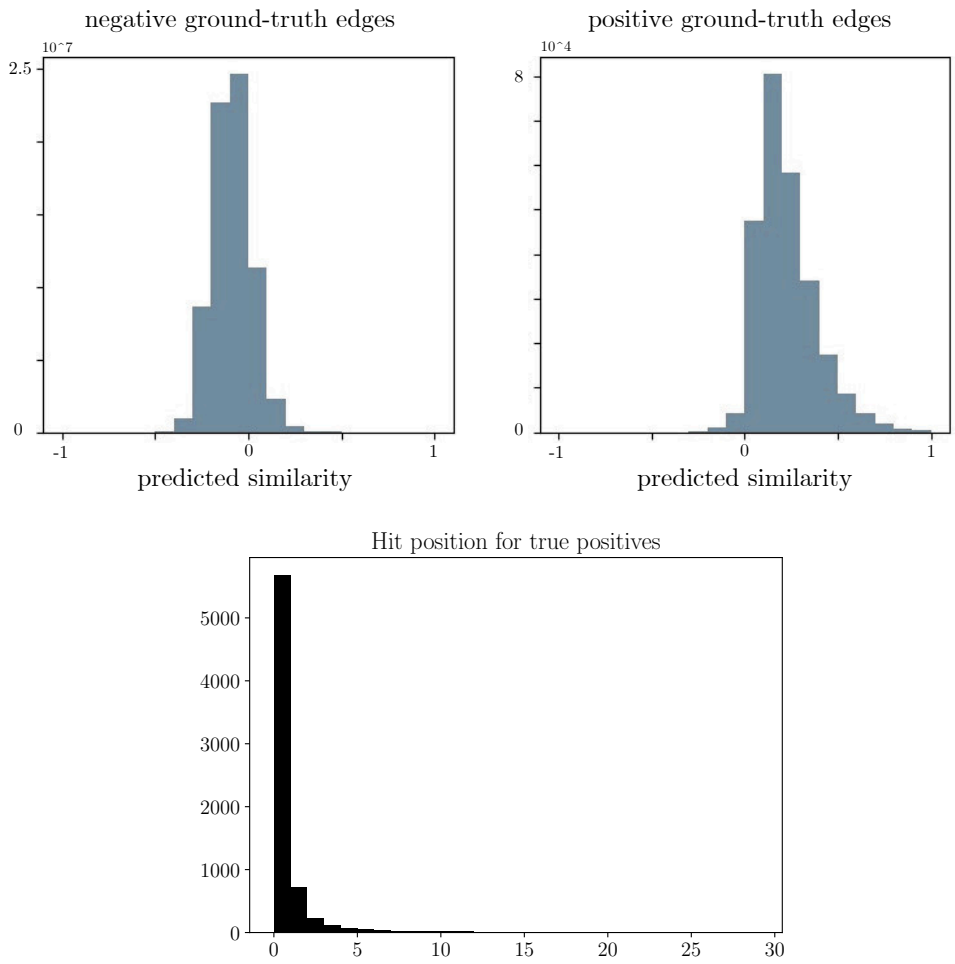


Fig. 5.2. – *Results compared to the ground truth friendship graph (FeasCL-5K).*

5.3, mean Precision increases from 66.06% to 68.14% and then to 71.25% as the number of epochs rises from 500, to 1,000 and then to 5,000. Figure 5.1, which shows the triplet loss of the FeasCL-5K model, may explain this trend: the loss of the model rapidly decreases in the first hundreds of epochs, but, after 1,000 epochs, it reaches a point where little to no changes occur. This plateau may indicate that the model has sufficiently learned the relationships within the data and further updates provide diminishing returns. In our case, this may reflect the difficulty in pushing farther away from the anchor the appropriate negative samples as training continues — initially, the model can easily find dissimilar negative samples, but as the learning progresses, most negative samples become less informative for improving the triplet loss, leading to slower improvements.

Tab. 5.4. Comparison when choosing a threshold*.

Method (%)	AUROC	AVGPR	PR	ACC	REC	TN	FP	FN	TP
CEM-ER	84.36	44.58	85.85	99.74	34.04	70,908,123	14,599	171,651	88,596
CEM-SBM	85.28	44.13	86.98	99.73	30.31	70,910,911	11,811	181,375	78,872
Node2Vec(>0)	87.44	1.52	0.37	0.67	100	216,434	70,706,288	8	260,239
Node2Vec(>0.5)			1.55	77.23	97.72	54,722,508	16,200,214	5,926	254,321
FeasCL-5K(>0)	96.91	24.47	1.77	80.20	97.75	56,830,745	14,091,977	5,869	254,378
FeasCL-5K(>0.5)			57.50	99.64	6.33	70,910,544	12,178	243,769	164,788

*an edge from i to j is drawn if $sim(\mathbf{z}_i^s, \mathbf{z}_j^r) > threshold$

To examine the overall pairwise similarities that FeasCL has learned, we can look into Figure 5.2. It shows the learned similarities for all possible user pairs ($N(N-1)$ in total) against the information of whether the edge exists in the ground truth friendship graph or not. For non-existent edges in the ground truth (top left figure), the learned similarities correctly weigh around 0 and toward the negative x-axis. In contrast, for existent ground truth edges (top right figure), the similarities weigh higher than 0. Still, the majority of edges are predicted between 0 and 0.5, which may be due to the loss struggling to push away the negative samples from the anchor as we explained before. It leads to a challenge in distinguishing between existing and non-existing edges in the final friendship graph via a strict threshold, as done in the case of CEM (refer to Section 4.6.3). In the bottom figure, showing the positions where we have first found a true positive neighbor for each user, we see the following: for the vast majority of users, the neighbors that are ranked the closest to them (in the first 1-5 positions) are the ones that are also existing in the ground truth graph (true positives). This means that if in absolute values the similarities are not high, looking into the neighbor rankings for each user individually may give us some useful insights regarding the existing underlying connections.

Consequently, the above further reinforces our choice of the k-closest neighbors as a more suitable metric for comparison. To explore this more in practice, we can look into Table 5.4. It displays the results for different performance metrics when inferring the edges in the final graph using a threshold strategy. Specifically, it considers the scenario of inferring a directed edge (i, j) in the final friendship graph whenever the user-user similarity $sim(\mathbf{z}_i^s, \mathbf{z}_j^r)$ learned by FeasCL is greater than a threshold. Given that the cosine similarity gives values between -1 and 1 we try two thresholds for Node2Vec and FeasCL-5K (0 and 0.5). For CEM, we show only the results for a threshold of 0.5, since the derived edge probabilities

are between 0 and 1 (as in Section 4.6.3). As we see, in the case of the threshold 0, the results of FeasCL are less informative, notably because of the inference of too many positive edges, resulting in many false positives (FP). This may result in a high Recall (=97.75), compared to CEM (=34.04), but with a heavily penalized metric of Precision (PR) (=1.77). However, the metrics of the Area under the ROC Curve (AUROC) and the Average Precision (AVGPR) perform relatively better (96.91 and 24.47), showing the impact that different thresholds can have on the performance (they are reported only once in the Table for each method, because they try all possible thresholds to give the results, instead of just one).

Overall, FeasCL still performs significantly better than Node2Vec when following the same threshold strategy, especially compared to the metrics of Accuracy (ACC) and Precision. As we see, Node2Vec draws many positive edges, which is not surprising since it has not been tailored to the task of inferring underlying graphs of connections, but rather to traditional ML classification tasks. When we draw a stricter threshold of 0.5 for both methods, we see that Precision for FeasCL-5k increases significantly (=57.50), at the expense of Recall (=6.33). This means that we draw fewer edges, but more precise ones. This is not the case for Node2Vec however, which still draws many false positives. Here we notice a trade-off between FeasCL-5k and CEM, with the latter drawing fewer true positive edges (TP) than FeasCL-5k, but showing also less false negatives (FP). This raises the question of what kind of prediction class we are prioritizing (see Section 2.5). If our goal is to draw as much as positive edges as possible, with a relatively good Precision, one could choose FeasCL. If, however, Precision is a priority, CEM would be the preferable choice.

For the same reasons, it is more challenging to compute the feasibility of the inferred graph, since we do not have available a reliable posterior structure as in the case of CEM. However, if we keep the strict threshold strategy presented above, we find that it gives 100% feasibility — in this case, however, this is less informative since the number of positive edges inferred is very high, and feasibility is trivial. If we increase the threshold to 0.5, the feasibility becomes significantly lower, equal to 26%.

As we show in the Appendix (B), the optimization process of FeasCL, like most DL models, may include many parameters that need tuning but offers significant advantages in memory and running costs compared to a strict optimization approach. For example, one epoch in FeasCL might take less than 15 seconds on

average (including the relatively expensive function of hard sampling) whereas an iteration in CEM can take several minutes each, on top of the large memory cost of storing constraints and its heavy reliance on professional tools like Gurobi for faster convergence.

5.6 Discussion and conclusion

With the FeasCL approach, we attempted to show that a simple and scalable contrastive strategy can give promising results when trying to infer the graph underlying a set of user-user interactions in a timely feasible manner. Most importantly, this can be achieved without the need for any labels or node features, relying solely on the temporal information provided by the data itself. The goal is to provide an alternative to the method CEM that we proposed in Chapter 4, which relies on constrained combinatorial optimization and can be both memory-demanding (due to the explicit set of feasibility constraints) and time-consuming to solve. FeasCL proposes instead a more relaxed interpretation of feasibility, that does not require explicit constraints, learning instead from the data in a contrastive, time-aware fashion.

However, it is crucial to highlight that achieving these promising results involved extensive experimentation with various parameters and settings. This stands as one of the main weaknesses of this method compared to traditional optimization approaches. Additionally, for more competitive results, we had to substantially increase the number of epochs for the loss function, consequently prolonging the computation time. Nevertheless, the method requires less memory since it no longer relies on an explicit set of feasibility constraints for inference that need to be stored at each update step. It also allows for applicability in more domains where the exact constraints for feasibility might not be as straightforward to construct as in the case of OSNs. However, as we demonstrated, this lack of explicit guidance may affect its performance penalizing the precision performance in the inferred connections. Yet, it may be more suitable in the common real-life scenario where the ground truth is not 100% feasible itself, relaxing the requirement for feasibility in the inference.

Moreover, unlike traditional statistical methods such as CEM that provide principled ways to derive posterior distributions for edge probabilities using well-established ML and EM methods, FeasCL lacks direct access to them. Instead, it estimates the existence of edges indirectly via similarity functions like cosine

similarity. As a solution, in this contrastive setting, we showed that it would be preferable to focus on the user neighborhoods, as they might provide more informative insights into the true underlying connections.

Of course, all these provide opportunities for further refinement. For instance, incorporating a time-decaying effect between sampled users or exploring alternative, more refined sampling strategies, could enhance the model's performance. Furthermore, investigating more expressive GNN architectures, or incorporating explicitly some notion of Bayesian modeling into the contrastive setting could also lead to improvements. On top of that, one should certainly compare with more relevant architectures aside from Node2Vec, that are more expressive and can deal with similar contrastive settings. This will provide further evaluation insights regarding the potential of the existing representation learning algorithms to provide intuitive underlying structures in unreliable and time-dependent settings.

All in all, we believe that FeasCL stands as a promising foundation, from which one can continue building representation learning techniques in unreliable settings, in a fashion that is intuitive and incorporates the concept of time feasibility. It also emphasizes the importance of evaluating the graph structure that can be inferred from the learned representations, which happens to be completely overlooked in standard practices, where the focus instead is on optimizing downstream tasks.

* * *

6 Conclusion

In this thesis, we tackled the problem of network inference in the context of Online Social Networks, where the available data is often unreliable, containing noise and missing information. Our work aligns with recent research emphasizing the necessity to acknowledge the unreliability inherent in all forms of data, including those derived from networks. Although a simple idea, it is an aspect that is frequently overlooked in practice, potentially compromising the quality of findings and conclusions. This hints at the existence of an underlying network shaping the dataset, which is often very challenging or impossible to access.

Further investigating this problem, we reviewed various inference techniques that exist and could infer the network underlying a set of temporal interactions. These methods span from statistical methods, using concepts like information diffusion, maximum likelihood, and expectation maximization, to more recent deep learning and graph representation learning techniques. However, we identified only a minority of works explicitly tackling unreliable settings, and potential shortcomings in the methods employed to evaluate their performance. More specifically, the absence of public evaluation datasets containing both the data and the real underlying network, often led to an over-reliance on artificially created datasets, or heuristics which might be insufficient in providing a full assessment. Most notably, many works completely ignored to evaluate the network inferred, optimizing instead on indirect goals and downstream tasks.

Bridging this gap, this thesis introduces the novel evaluation metric of feasibility, evaluating to what extent the final network inferred can explain the dataset that we gave as input. This metric is specifically tailored to assess inference methods, since it does not depend on a ground truth, but rather on an intuitive explanation of the inferred results. It takes into account not only the existence of hidden paths between nodes, but also the dimension of time, ensuring that the inferred paths respect the temporal ordering of the node-to-node interactions observed in the dataset. The intuition behind this is that, if a dataset is feasible given an inferred network, then the result is also close to the true network that we are trying to infer. Perhaps not surprisingly, we found that the existing inference methods do

not guarantee feasibility, leading to results that cannot explain the dataset and are thus less intuitive in their assessment.

We provided a novel network inference method, called Constrained Expectation Maximization (CEM), to ensure the inference of a 100% feasible network when given datasets consisting of repost events from Online Social Networks. The assumption behind this is that users repost users they follow, and therefore the underlying network we are targeting is their friendship network. In practice, we incorporated into an Expectation Maximization-based optimization process, a series of additional linear optimization updates to account for the notion of hidden paths and feasibility constraints. This eventually guides the inference process towards feasibility. We presented two variations of CEM, each assuming either an Erdős–Rényi (ER) or an Stochastic Block Model (SBM) prior for the underlying graph's distribution.

Extensive experimentation on both synthetic and real-world Twitter datasets showed that regardless of the prior chosen, CEM can generate a posterior distribution of graphs that is 100% feasible, while closely approximating the ground truth. Notably, when employing the SBM prior, CEM simultaneously infers clusters of users during optimization, offering an additional benefit. We also proposed a heuristic to adapt the inference process to lower feasibility requirements, exploring its impact on precision. On top of feasibility, we evaluated the inferred networks in terms of a well-studied set of graph measures, evaluating to what extent they demonstrate real-world properties. All in all, in terms of feasibility, precision and real-world properties of the result, we found that CEM outperforms baselines as well as more novel algorithms.

These findings suggest that feasible graphs may indeed better approximate the underlying graph compared to non-feasible ones. Certainly, feasibility alone cannot guarantee accuracy, as evidenced by the unrealistic properties of some heuristically inferred networks despite them being 100% feasible. As a result, we recommend that in a network inference setting, where an underlying graph is not available for evaluation, the metric of feasibility should be explored on top of the different graph theory metrics to verify their real-world properties. We should also note that our method has only been validated given a very specific dataset structure that is based on the data that most social media platforms currently offer. To apply this method to other datasets, domain expertise might be needed to adapt the concept of feasibility accordingly.

As a final investigation, we explored recent advancements in graph representation learning, identifying machine and deep learning methods as potential flexible and scalable alternatives. The main difference to the CEM framework, is that the goal is no longer to derive feasible posterior distributions of the entire graph structure but to learn meaningful node embeddings. We provided an extensive literature review of such methods, relevant either directly to the inference of information diffusion or more general machine learning approaches that do not require supervision on known labels. We identified certain limitations in these methods regarding their applicability to our problem. The majority did not consider network unreliability and ignored to evaluate the underlying structure that was being inferred, optimizing instead on other machine learning tasks such as classifying the nodes in the correct classes.

The above prompted our proposal of a novel feasible network inference alternative, called Feasible Contrastive Learning (FeasCL): Initially, we suggested framing the general network inference problem within a self-supervised learning framework. This approach utilized an encoder model commonly deployed for link prediction tasks, leveraging Graph Neural Networks (GNNs). To integrate the concept of feasibility, we introduced a straightforward contrastive loss mechanism capable of learning node embeddings while considering time through a basic sampling technique. The main idea is that the node embeddings of users who could be feasibly connected in the underlying graph should also be encoded closer together in the latent space. We conducted preliminary experiments, comparing our approach against CEM and a deep learning baseline Node2Vec. The results were promising; they seemed to be competitive in terms of precision against CEM and significantly better than Node2Vec. As future improvements, we could explore the utilization of a more expressive GNN architecture, a refined sampling strategy, or integrate Bayesian modeling into the contrastive framework to explicitly handle uncertainty. Moreover, for a more thorough validation, conducting additional experiments across a broader spectrum of datasets and comparing them with state-of-the-art methods is an essential step. Nevertheless, the results hint at an exciting new direction that could enhance the interpretability of the learning process and the general scalability of the network inference problem.

In future endeavors, one could extend our method and constraints to diverse data types and graph inference scenarios, adapting them to a broader spectrum of networks used in epidemics, biology, physics, and other domains. In any

case, we believe that the concept of feasibility proposed in this thesis could be of greater interest to the scientific community, especially in today's era of "black box" Artificial Intelligence, promoting the development of algorithms that could guarantee more intuitive and interpretable results.

* * *

Bibliography

Abbe, Emmanuel, Afonso S Bandeira, and Georgina Hall (2015). "Exact recovery in the stochastic block model". In: *IEEE Transactions on information theory* 62.1, pp. 471–487 (cit. on p. 45).

Abboud, Ralph, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz (2020). "The surprising power of graph neural networks with random node initialization". In: *arXiv preprint arXiv:2010.01179* (cit. on p. 122).

Adamic, Lada A and Eytan Adar (2003). "Friends and neighbors on the web". In: *Social networks* 25.3, pp. 211–230 (cit. on p. 47).

Aicher, Christopher, Abigail Z Jacobs, and Aaron Clauset (2015). "Learning latent block structure in weighted networks". In: *Journal of Complex Networks* 3.2, pp. 221–248 (cit. on p. 47).

Airoldi, Edo M, David Blei, Stephen Fienberg, and Eric Xing (2008). "Mixed membership stochastic blockmodels". In: *Advances in neural information processing systems* 21 (cit. on p. 48).

Albert, Réka (2007). "Network inference, analysis, and modeling in systems biology". In: *The Plant Cell* 19.11, pp. 3327–3338 (cit. on p. 7).

Aldrich, John (1997). "RA Fisher and the making of maximum likelihood 1912-1922". In: *Statistical science* 12.3, pp. 162–176 (cit. on p. 43).

Allahverdyan, Armen E, Greg Ver Steeg, and Aram Galstyan (2010). "Community detection with and without prior information". In: *Europhysics Letters* 90.1, p. 18002 (cit. on p. 35).

Alpaydin, Ethem (2020). *Introduction to machine learning*. MIT press (cit. on p. 100).

Amaral, Luis A Nunes (2008). "A truer measure of our ignorance". In: *Proceedings of the National Academy of Sciences* 105.19, pp. 6795–6796 (cit. on p. 23).

Anderson, Roy M and Robert M May (1991). *Infectious diseases of humans: dynamics and control*. Oxford university press (cit. on p. 33).

Appel, Kenneth, Wolfgang Haken, and John Koch (1977). "Every planar map is four colorable. Part II: Reducibility". In: *Illinois Journal of Mathematics* 21.3, pp. 491–567 (cit. on p. 5).

Bachman, Philip, R Devon Hjelm, and William Buchwalter (2019). "Learning representations by maximizing mutual information across views". In: *Advances in neural information processing systems* 32 (cit. on p. 111).

Backstrom, Lars, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna (2012). "Four degrees of separation". In: *Proceedings of the 4th Annual ACM Web Science Conference*, pp. 33–42 (cit. on p. 32).

Balach, Prakash, Eric D Kolaczyk, Weston D Viles, et al. (2017). "On the propagation of low-rate measurement error to subgraph counts in large networks". In: *Journal of Machine Learning Research* 18.61, pp. 1–33 (cit. on p. 47).

Ball, Walter William Rouse (1893). "Mathematical Recreations and Essays". In: *Bulletin des sciences mathématiques* 17, pp. 105–107 (cit. on p. 3).

Barabási, Albert-László (2013). "Network science". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1987, p. 20120375 (cit. on pp. 4, 5, 35).

Barabási, Albert-László and Réka Albert (1999). "Emergence of scaling in random networks". In: *Science* 286.5439, pp. 509–512 (cit. on pp. 5, 31).

Barrat, Alain and Martin Weigt (2000). "On the properties of small-world network models". In: *The European Physical Journal B-Condensed Matter and Complex Systems* 13, pp. 547–560 (cit. on p. 31).

Baumgartner, Jason, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn (2020). "The pushshift reddit dataset". In: *Proceedings of the international AAAI conference on web and social media*. Vol. 14, pp. 830–839 (cit. on p. 49).

Bavelas, Alex (1950). "Communication patterns in task-oriented groups". In: *The journal of the acoustical society of America* 22.6, pp. 725–730 (cit. on p. 29).

Beal, Matthew J and Zoubin Ghahramani (2006). "Variational Bayesian learning of directed graphical models with hidden variables". In: (cit. on pp. 44, 47).

Bellman, Richard (1958). "On a routing problem". In: *Quarterly of applied mathematics* 16.1, pp. 87–90 (cit. on pp. 4, 28).

Berg, Rianne van den, Thomas N Kipf, and Max Welling (2017). "Graph convolutional matrix completion". In: *arXiv preprint arXiv:1706.02263* (cit. on p. 121).

Bernard, H Russell, Peter Killworth, David Kronenfeld, and Lee Sailer (1984). "The problem of informant accuracy: The validity of retrospective data". In: *Annual review of anthropology* 13.1, pp. 495–517 (cit. on p. 22).

Berry, Michael W, Azlinah Mohamed, and Bee Wah Yap (2019). *Supervised and unsupervised learning for data science*. Springer (cit. on p. 101).

Bhagat, Smriti, Graham Cormode, and S Muthukrishnan (2011). "Node classification in social networks". In: *Social network data analytics*, pp. 115–148 (cit. on p. 102).

Bishop, Christopher M and Hugh Bishop (2024). "Deep learning: foundations and concepts". In: *(No Title)* (cit. on p. 101).

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe (2017). "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518, pp. 859–877 (cit. on p. 48).

Blondel, Vincent, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre (2008). "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* (cit. on pp. 31, 71, 79).

Bollobás, Béla (1998). *Random graphs*. Springer (cit. on p. 36).

Bollobás, Béla, Christian Borgs, Jennifer T Chayes, and Oliver Riordan (2003). "Directed scale-free graphs." In: *SODA*. Vol. 3. Baltimore, MD, United States, pp. 132–139 (cit. on p. 31).

Bottou, Léon, Frank E Curtis, and Jorge Nocedal (2018). "Optimization methods for large-scale machine learning". In: *SIAM review* 60.2, pp. 223–311 (cit. on p. 121).

Bourigault, Simon, Cedric Lagnier, Sylvain Lamprier, Ludovic Denoyer, and Patrick Gallinari (2014). "Learning social network embeddings for predicting information diffusion". In: *Proceedings of the 7th ACM international conference on Web search and data mining*, pp. 393–402 (cit. on pp. 33, 105, 117).

Bourigault, Simon, Sylvain Lamprier, and Patrick Gallinari (2016). "Representation learning for information diffusion through social networks: an embedded cascade model". In: *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, pp. 573–582 (cit. on p. 105).

Brugere, Ivan, Brian Gallagher, and Tanya Y Berger-Wolf (2018). "Network structure inference, a survey: Motivations, methods, and applications". In: *ACM Computing Surveys (CSUR)* 51.2, pp. 1–39 (cit. on pp. 6, 8, 13, 14, 23, 45, 46).

Bürkner, Paul-Christian, Maximilian Scholz, and Stefan T Radev (2023). "Some models are useful, but how do we know which ones? Towards a unified Bayesian model taxonomy". In: *Statistic Surveys* 17, pp. 216–310 (cit. on p. 48).

Butts, Carter T (2003). "Network inference, error, and informant (in) accuracy: a Bayesian approach". In: *social networks* 25.2, pp. 103–140 (cit. on pp. 6–10, 22, 41, 42).

– (2009). "Revisiting the foundations of network analysis". In: *science* 325.5939, pp. 414–416 (cit. on p. 9).

Cai, Hongyun, Vincent W Zheng, and Kevin Chen-Chuan Chang (2018). "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE transactions on knowledge and data engineering* 30.9, pp. 1616–1637 (cit. on p. 103).

Cao, Qi, Huawei Shen, Keting Cen, Wentao Ouyang, and Xueqi Cheng (2017). "Deephawkes: Bridging the gap between prediction and understanding of information cascades". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1149–1158 (cit. on p. 108).

Cappart, Quentin, Didier Chételat, Elias B Khalil, et al. (2023). "Combinatorial optimization and reasoning with graph neural networks". In: *Journal of Machine Learning Research* 24.130, pp. 1–61 (cit. on p. 111).

Cayley, Arthur (1890). "On the theory of the analytical forms called trees". In: *Mathematical papers* 3, pp. 242–246 (cit. on p. 3).

Chen, Fenxiao, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo (2020). "Graph representation learning: a survey". In: *APSIPA Transactions on Signal and Information Processing* 9, e15 (cit. on p. 104).

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton (2020). "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR, pp. 1597–1607 (cit. on pp. 111, 112).

Chen, Xueqin, Fan Zhou, Kunpeng Zhang, et al. (2019). "Information diffusion prediction via recurrent cascades convolution". In: *2019 IEEE 35th international conference on data engineering (ICDE)*. IEEE, pp. 770–781 (cit. on p. 109).

Chen, Zhihao, Jingjing Wei, Shaobin Liang, Tiecheng Cai, and Xiangwen Liao (2021). "Information cascades prediction with graph attention". In: *Frontiers in Physics* 9, p. 739202 (cit. on p. 108).

Clauset, Aaron, Cristopher Moore, and Mark EJ Newman (2008). "Hierarchical structure and the prediction of missing links in networks". In: *Nature* 453.7191, pp. 98–101 (cit. on pp. 8, 22, 23, 45).

Cointet, Jean-Philippe, Dominique Cardon, Andrei Mogoutov, et al. (2021). "Uncovering the structure of the French media ecosystem". In: *arXiv preprint arXiv:2107.12073* (cit. on p. 11).

Daley, Daryl J and Joseph Gani (1999). *Epidemic modelling: an introduction*. Cambridge University Press (cit. on p. 60).

DataReportal (2024). *Digital 2024: Global Overview Report*. Available online: https://datareportal.com/reports/digital-2024-global-overview-report (cit. on p. 12).

Daud, Nur Nasuha, Siti Hafizah Ab Hamid, Muntadher Saadoon, Firdaus Sahran, and Nor Badrul Anuar (2020). "Applications of link prediction in social networks: A review". In: *Journal of Network and Computer Applications* 166, p. 102716 (cit. on p. 103).

Daudin, J-J, Franck Picard, and Stéphane Robin (2008). "A mixture model for random graphs". In: *Statistics and computing* 18.2, pp. 173–183 (cit. on pp. 35, 47).

Dawid, Alexander Philip and Allan M Skene (1979). "Maximum likelihood estimation of observer error-rates using the EM algorithm". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28.1, pp. 20–28 (cit. on p. 44).

De Smet, Riet and Kathleen Marchal (2010). "Advantages and limitations of current network inference methods". In: *Nature Reviews Microbiology* 8.10, pp. 717–729 (cit. on p. 10).

Deane, Charlotte M, Łukasz Salwinski, Ioannis Xenarios, and David Eisenberg (2002). "Protein interactions: two methods for assessment of the reliability of high throughput observations". In: *Molecular & Cellular Proteomics* 1.5, pp. 349–356 (cit. on pp. 8, 23).

Dempster, Arthur P, Nan M Laird, and Donald B Rubin (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society* (cit. on p. 44).

Dijkstra, EW (1959). "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1.1, pp. 269–271 (cit. on pp. 4, 28).

Dobbin, Kevin K and Richard M Simon (2011). "Optimally splitting cases for training and testing high dimensional classifiers". In: *BMC medical genomics* 4, pp. 1–8 (cit. on p. 102).

Domingos, Pedro and Matt Richardson (2001). "Mining the network value of customers". In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 57–66 (cit. on p. 50).

Dooms, Simon, Toon De Pessemier, and Luc Martens (2013). "Movietweetings: a movie rating dataset collected from twitter". In: *Workshop on Crowdsourcing and human computation for recommender systems, CrowdRec at RecSys*. Vol. 2013, p. 43 (cit. on p. 49).

Eckle, Konstantin and Johannes Schmidt-Hieber (2019). "A comparison of deep networks with ReLU activation function and linear spline-type methods". In: *Neural Networks* 110, pp. 232–242 (cit. on p. 116).

Elinas, Pantelis, Edwin V Bonilla, and Louis Tiao (2020). "Variational inference for graph convolutional networks in the absence of graph data and adversarial settings". In: *Advances in neural information processing systems* 33, pp. 18648–18660 (cit. on p. 110).

Erdős, Paul and Alfréd Rényi (1959). "On random graphs I". In: *Publicationes Mathematicae Debrecen* 6.290-297, p. 18 (cit. on pp. 4, 35).

Euler, Leonhard (1741). "Solutio problematis ad geometriam situs pertinentis". In: *Commentarii academiae scientiarum Petropolitanae*, pp. 128–140 (cit. on p. 3).

Failla, Andrea and Giulio Rossetti (2024). ""I'm in the Bluesky Tonight": Insights from a Year Worth of Social Data". In: *arXiv preprint arXiv:2404.18984* (cit. on pp. 13, 25, 49).

Feng, Shanshan, Gao Cong, Arijit Khan, et al. (2018). "Inf2vec: Latent representation model for social influence embedding". In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, pp. 941–952 (cit. on p. 108).

Firestone, Simon M, Yoko Hayama, Max SY Lau, et al. (2020). "Transmission network reconstruction for foot-and-mouth disease outbreaks incorporating farm-level covariates". In: *PLoS One* (cit. on pp. 8, 51).

Ford, Lester Randolph (1956). "Network flow theory". In: (cit. on p. 4).

Fraisier, O, O Cabanac, Y Pitarch, R Besançon, and M Boughanem (2018). "#Elysee2017fr: The 2017 French Presidential Campaign on Twitter". In: *Proceedings of the 12th International AAAI Conference on Web and Social Media* (cit. on p. 75).

Freeman, Linton C (1977). "A set of measures of centrality based on betweenness". In: *Sociometry*, pp. 35–41 (cit. on p. 29).

Freeman, Linton C, A Kimball Romney, and Sue C Freeman (1987). "Cognitive structure and informant accuracy". In: *American anthropologist* 89.2, pp. 310–325 (cit. on p. 22).

Friedman, N, M Linial, I Nachman, and D Pe'er (2000). "Using Bayesian networks to analyze expression data". In: *Journal of Computational Biology* 7, pp. 601–620 (cit. on p. 7).

Ganley, Dale and Cliff Lampe (2009). "The ties that bind: Social network principles in online communities". In: *Decision support systems* 47.3, pp. 266–274 (cit. on p. 31).

Gao, Chao and Jiming Liu (2016). "Network-based modeling for characterizing human collective behaviors during extreme events". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.1, pp. 171–183 (cit. on p. 10).

Gavilanes, Ruth Garcia, Neil O'Hare, Luca Maria Aiello, and Alejandro Jaimes (2013). "Follow my friends this Friday! An analysis of human-generated friendship recommendations". In: *Social Informatics: 5th International Conference, SocInfo 2013, Kyoto, Japan, November 25-27, 2013, Proceedings 5*. Springer, pp. 46–59 (cit. on p. 33).

Gell-Mann, Murray (1994). "Complex adaptive systems". In: *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*. Vol. 19. ADDISON-WESLEY PUBLISHING CO, pp. 17–17 (cit. on p. 5).

Gelman, Andrew (2004). "Parameterization and Bayesian modeling". In: *Journal of the American Statistical Association* 99.466, pp. 537–545 (cit. on p. 42).

Ghahramani, Zoubin (2003). "Unsupervised learning". In: *Summer school on machine learning*. Springer, pp. 72–112 (cit. on p. 111).

Ghasemian, Amir, Homa Hosseinmardi, Aram Galstyan, Edoardo M Airoldi, and Aaron Clauset (2020). "Stacking models for nearly optimal link prediction in complex networks". In: *Proceedings of the National Academy of Sciences* 117.38, pp. 23393–23400 (cit. on p. 10).

Giesecke, Kay, Gustavo Schwenkler, and Justin A Sirignano (2020). "Inference for large financial systems". In: *Mathematical Finance*, pp. 3–46 (cit. on p. 8).

Gilbert, Edgar Nelson (1956). "Enumeration of labelled graphs". In: *Canadian Journal of Mathematics* 8, pp. 405–411 (cit. on p. 3).

– (1959). "Random graphs". In: *The Annals of Mathematical Statistics* 30.4, pp. 1141–1144 (cit. on pp. 4, 35).

Giovanidis, Anastasios, Bruno Baynat, Clémence Magnien, and Antoine Vendeville (2021). "Ranking online social users by their influence". In: *IEEE/ACM Transactions on Networking* 29.5, pp. 2198–2214 (cit. on p. 73).

Girvan, Michelle and Mark EJ Newman (2002). "Community structure in social and biological networks". In: *Proceedings of the national academy of sciences* 99.12, pp. 7821–7826 (cit. on p. 45).

Goldberg, Debra S and Frederick P Roth (2003). "Assessing experimentally derived interactions in a small world". In: *Proceedings of the National Academy of Sciences* 100.8, pp. 4372–4376 (cit. on pp. 7, 8, 23, 41, 42).

Gomez-Rodriguez, Manuel, Jure Leskovec, and Andreas Krause (2012). "Inferring networks of diffusion and influence". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* (cit. on pp. 11, 13, 77, 82–86, 89–91, 93).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press (cit. on p. 100).

Gori, Marco, Gabriele Monfardini, and Franco Scarselli (2005). "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. IEEE, pp. 729–734 (cit. on pp. 6, 115).

Goyal, Amit, Francesco Bonchi, and Laks VS Lakshmanan (2010). "Learning influence probabilities in social networks". In: *Proceedings of the third ACM international conference on Web search and data mining*, pp. 241–250 (cit. on p. 50).

Grandjean, Martin (2016). "A social network analysis of Twitter: Mapping the digital humanities community". In: *Cogent arts & humanities* 3.1, p. 1171458 (cit. on p. 31).

Granovetter, Mark S (1973). "The strength of weak ties". In: *American journal of sociology* 78.6, pp. 1360–1380 (cit. on pp. 4, 33).

Griffiths, Thomas L and Mark Steyvers (2004). "Finding scientific topics". In: *Proceedings of the National academy of Sciences* 101.suppl_1, pp. 5228–5235 (cit. on p. 48).

Grover, Aditya and Jure Leskovec (2016). "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864 (cit. on pp. 106, 124).

Gruhl, Daniel, Ramanathan Guha, David Liben-Nowell, and Andrew Tomkins (2004). "Information diffusion through blogspace". In: *Proceedings of the 13th international conference on World Wide Web*, pp. 491–501 (cit. on p. 33).

Guimerà, Roger and Marta Sales-Pardo (2009). "Missing and spurious interactions and the reconstruction of complex networks". In: *Proceedings of the National Academy of Sciences* 106.52, pp. 22073–22078 (cit. on pp. 8, 22, 23, 48, 51).

Hadsell, Raia, Sumit Chopra, and Yann LeCun (2006). "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE, pp. 1735–1742 (cit. on p. 111).

Hagmann, Patric, Leila Cammoun, Xavier Gigandet, et al. (2008). "Mapping the structural core of human cerebral cortex". In: *PLoS biology* 6.7, e159 (cit. on p. 10).

Hamilton, William L (2020). *Graph representation learning*. Morgan & Claypool Publishers (cit. on pp. 47, 101–103, 113, 115).

Hamilton, William L, Zhitao Ying, and Jure Leskovec (2017). "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (cit. on pp. 106, 118, 122, 162).

Handcock, Mark S, Adrian E Raftery, and Jeremy M Tantrum (2007). "Model-based clustering for social networks". In: *Journal of the Royal Statistical Society Series A: Statistics in Society* 170.2, pp. 301–354 (cit. on p. 48).

Harary, Edgar N (1969). *Graph Theory*. Adisson-Wesley (cit. on p. 4).

Hasan, Mohammad Al and Mohammed J Zaki (2011). "A survey of link prediction in social networks". In: *Social network data analytics*, pp. 243–275 (cit. on p. 103).

Hasanzadeh, Arman, Ehsan Hajiramezanali, Shahin Boluki, et al. (2020). "Bayesian graph neural networks with adaptive connection sampling". In: *International conference on machine learning*. PMLR, pp. 4094–4104 (cit. on p. 110).

Hastings, W Keith (1970). "Monte Carlo sampling methods using Markov chains and their applications". In: (cit. on p. 48).

Hayashi, Kohei, Takuya Konishi, and Tatsuro Kawamoto (2016). "A tractable fully bayesian method for the stochastic block model". In: *arXiv preprint arXiv:1602.02256* (cit. on p. 45).

Hethcote, Herbert W (2000). "The mathematics of infectious diseases". In: *SIAM review* 42.4, pp. 599–653 (cit. on p. 34).

Hoff, Peter D, Adrian E Raftery, and Mark S Handcock (2002). "Latent space approaches to social network analysis". In: *Journal of the american Statistical association* 97.460, pp. 1090–1098 (cit. on p. 48).

Holland, Paul W, Kathryn Blackmond Laskey, and Samuel Leinhardt (1983). "Stochastic blockmodels: First steps". In: *Social networks*, pp. 109–137 (cit. on pp. 37, 69).

Holme, Petter (2005). "Network reachability of real-world contact sequences". In: *Physical Review E* 71.4, p. 046119 (cit. on p. 26).

Holme, Petter and Jari Saramäki (2012). "Temporal networks". In: *Physics reports* 519.3, pp. 97–125 (cit. on p. 33).

Hopkins, Brian and Robin J Wilson (2004). "The truth about Königsberg". In: *The College Mathematics Journal* 35.3, pp. 198–207 (cit. on p. 2).

Hu, Weihua, Matthias Fey, Marinka Zitnik, et al. (2020). "Open graph benchmark: Datasets for machine learning on graphs". In: *Advances in neural information processing systems* 33, pp. 22118–22133 (cit. on p. 6).

Huang, Da, Fangyuan Lei, and Xi Zeng (2023). "SimGRL: a simple self-supervised graph representation learning framework via triplets". In: *Complex & Intelligent Systems* 9.5, pp. 5049–5062 (cit. on p. 119).

Ito, Takashi, Tomoko Chiba, Ritsuko Ozawa, et al. (2001). "A comprehensive two-hybrid analysis to explore the yeast protein interactome". In: *Proceedings of the National Academy of Sciences* 98.8, pp. 4569–4574 (cit. on p. 23).

Jaiswal, Ashish, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon (2020). "A survey on contrastive self-supervised learning". In: *Technologies* 9.1, p. 2 (cit. on pp. 112, 114).

James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor (2023). "Unsupervised learning". In: *An Introduction to Statistical Learning: with Applications in Python*. Springer, pp. 503–556 (cit. on p. 102).

Jin, Wei, Tyler Derr, Haochen Liu, et al. (2020). "Self-supervised learning on graphs: Deep insights and new direction". In: *arXiv preprint arXiv:2006.10141* (cit. on p. 111).

Jordan, Michael I, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul (1999). "An introduction to variational methods for graphical models". In: *Machine learning* 37, pp. 183–233 (cit. on p. 47).

Jovanović, Nikola, Zhao Meng, Lukas Faber, and Roger Wattenhofer (2021). "Towards robust graph contrastive learning". In: *arXiv preprint arXiv:2102.13085* (cit. on p. 120).

Karalias, Nikolaos and Andreas Loukas (2020). "Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs". In: *Advances in Neural Information Processing Systems* 33, pp. 6659–6672 (cit. on p. 111).

Karrer, Brian and Mark EJ Newman (2011). "Stochastic blockmodels and community structure in networks". In: *Physical review E* 83.1, p. 016107 (cit. on p. 38).

Kemp, Charles, Thomas L Griffiths, and Joshua B Tenenbaum (2004). "Discovering latent classes in relational data". In: (cit. on p. 48).

Kempe, David, Jon Kleinberg, and Amit Kumar (2000). "Connectivity and inference problems for temporal networks". In: *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pp. 504–513 (cit. on p. 49).

Kempe, David, Jon Kleinberg, and Éva Tardos (2003). "Maximizing the spread of influence through a social network". In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146 (cit. on p. 50).

Kendall, Alex and Yarin Gal (2017). "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in neural information processing systems* 30 (cit. on p. 110).

Kermack, William Ogilvy and Anderson G McKendrick (1927). "A contribution to the mathematical theory of epidemics". In: *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115.772, pp. 700–721 (cit. on p. 34).

Le-Khac, Phuc H, Graham Healy, and Alan F Smeaton (2020). "Contrastive representation learning: A framework and review". In: *Ieee Access* 8, pp. 193907–193934 (cit. on p. 112).

Khosla, Megha, Vinay Setty, and Avishek Anand (2019). "A comparative study for unsupervised network representation learning". In: *IEEE Transactions on Knowledge and Data Engineering* 33.5, pp. 1807–1818 (cit. on pp. 106, 107).

Killworth, Peter and H Bernard (1976). "Informant accuracy in social network data". In: *Human organization* 35.3, pp. 269–286 (cit. on p. 22).

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (cit. on p. 122).

Kipf, Thomas, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel (2018). "Neural relational inference for interacting systems". In: *International conference on machine learning*. PMLR, pp. 2688–2697 (cit. on p. 110).

Kipf, Thomas N and Max Welling (2016a). "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907. arXiv: 1609.02907 (cit. on pp. 6, 109, 116, 118).

– (2016b). "Variational graph auto-encoders". In: *arXiv preprint arXiv:1611.07308* (cit. on p. 110).

Kitchin, Rob (2014). *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage (cit. on p. 6).

König, Dénes (1936). "Theorie der endlichen und unendlichen Graphen: Kombinatorische Topologie der Streckenkomplexe". In: *(No Title)* (cit. on p. 3).

Kossinets, Gueorgi (2006). "Effects of missing data in social networks". In: *Social networks* 28.3, pp. 247–268 (cit. on pp. 9, 23, 47).

Kossinets, Gueorgi, Jon Kleinberg, and Duncan Watts (2008). "The structure of information pathways in a social communication network". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 435–443 (cit. on pp. 26, 49).

Kostakos, Vassilis (2009). "Temporal graphs". In: *Physica A: Statistical Mechanics and its Applications* 388.6, pp. 1007–1023 (cit. on p. 33).

Kotsiantis, Sotiris B, Ioannis Zaharakis, P Pintelas, et al. (2007). "Supervised machine learning: A review of classification techniques". In: *Emerging artificial intelligence applications in computer engineering* 160.1, pp. 3–24 (cit. on p. 101).

Kulathunga, Nalinda, Nishath Rajiv Ranasinghe, Daniel Vrinceanu, et al. (2020). "Effects of the nonlinearity in activation functions on the performance of deep learning models". In: *arXiv preprint arXiv:2010.07359* (cit. on p. 117).

Kurashima, Takeshi, Tomoharu Iwata, Noriko Takaya, and Hiroshi Sawada (2014). "Probabilistic latent network visualization: Inferring and embedding diffusion networks". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1236–1245 (cit. on pp. 105, 114).

Lamprier, Sylvain (2018). "A variational topological neural model for cascade-based diffusion in networks". In: *arXiv preprint arXiv:1812.10962* (cit. on pp. 107, 108).

Latouche, Pierre, Etienne Birmele, and Christophe Ambroise (2012). "Variational Bayesian inference and complexity control for stochastic block models". In: *Statistical Modelling* 12.1, pp. 93–115 (cit. on p. 47).

Le, Can M., Keith Levin, and Elizaveta Levina (2018). "Estimating a network from multiple noisy realizations". In: *Electronic Journal of Statistics*, pp. 4697–4740 (cit. on p. 46).

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *nature* 521.7553, pp. 436–444 (cit. on pp. 5, 100).

Leskovec, Jure, Lada A Adamic, and Bernardo A Huberman (2007). "The dynamics of viral marketing". In: *ACM Transactions on the Web (TWEB)* 1.1, 5–es (cit. on p. 49).

Leskovec, Jure and Eric Horvitz (2007). *Worldwide buzz: Planetary-scale views on an instant-messaging network*. Tech. rep. Technical report, Microsoft Research (cit. on p. 49).

Leskovec, Jure, Mary McGlohon, Christos Faloutsos, Natalie Glance, and Matthew Hurst (2007). "Patterns of cascading behavior in large blog graphs". In: *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, pp. 551–556 (cit. on p. 49).

Liao, Renjie, Yujia Li, Yang Song, et al. (2019). "Efficient graph generation with graph recurrent attention networks". In: *Advances in neural information processing systems* 32 (cit. on p. 109).

Liben-Nowell, David and Jon Kleinberg (2003). "The link prediction problem for social networks". In: *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559 (cit. on p. 47).

Lin, Xialing, Kenneth A Lachlan, and Patric R Spence (2016). "Exploring extreme events on social media: A comparison of user reposting/retweeting behaviors on Twitter and Weibo". In: *Computers in human behavior* 65, pp. 576–581 (cit. on p. 24).

Lipton, Zachary C, John Berkowitz, and Charles Elkan (2015). "A critical review of recurrent neural networks for sequence learning". In: *arXiv preprint arXiv:1506.00019* (cit. on p. 109).

Liu, Xiao, Fanjin Zhang, Zhenyu Hou, et al. (2021). "Self-supervised learning: Generative or contrastive". In: *IEEE transactions on knowledge and data engineering* 35.1, pp. 857–876 (cit. on p. 120).

Liu, Yixin, Ming Jin, Shirui Pan, et al. (2022). "Graph self-supervised learning: A survey". In: *IEEE transactions on knowledge and data engineering* 35.6, pp. 5879–5900 (cit. on p. 111).

Lokhov, Andrey (2016). "Reconstructing parameters of spreading models from partial observations". In: *Advances in Neural Information Processing Systems*, pp. 3467–3475 (cit. on p. 51).

Loossens, Tim, Francis Tuerlinckx, and Stijn Verdonck (2021). "A comparison of continuous and discrete time modeling of affective processes in terms of predictive accuracy". In: *Scientific reports* 11.1, p. 6218 (cit. on p. 49).

Lü, Linyuan and Tao Zhou (2011). "Link prediction in complex networks: A survey". In: *Physica A: statistical mechanics and its applications* 390.6, pp. 1150–1170 (cit. on pp. 46, 51).

Luo, Chunjie, Jianfeng Zhan, Xiaohe Xue, et al. (2018). "Cosine normalization: Using cosine similarity instead of dot product in neural networks". In: *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*. Springer, pp. 382–391 (cit. on p. 116).

Lyu, Tianshu, Yuan Zhang, and Yan Zhang (2017). "Enhancing the network embedding quality with structural similarity". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 147–156 (cit. on p. 107).

Malhotra, Arvind, Claudia Kubowicz Malhotra, and Alan See (2012). "How to get your messages retweeted". In: *MIT Sloan Management Review* 53.2, pp. 61–66 (cit. on p. 24).

Marin, Alexandra and Barry Wellman (2011). "Social network analysis: An introduction". In: *The SAGE handbook of social network analysis*, pp. 11–25 (cit. on p. 31).

Marsden, Peter V (1990). "Network data and measurement". In: *Annual review of sociology* 16.1, pp. 435–463 (cit. on pp. 9, 22).

Martínez, Víctor, Fernando Berzal, and Juan-Carlos Cubero (2016). "A survey of link prediction in complex networks". In: *ACM computing surveys (CSUR)* 49.4, pp. 1–33 (cit. on p. 103).

Masters, Dominic and Carlo Luschi (2018). "Revisiting small batch training for deep neural networks". In: *arXiv preprint arXiv:1804.07612* (cit. on p. 121).

McPherson, Miller, Lynn Smith-Lovin, and James M Cook (2001). "Birds of a feather: Homophily in social networks". In: *Annual review of sociology* 27.1, pp. 415–444 (cit. on p. 31).

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (cit. on p. 106).

Milgram, Stanley (1967). "The small world problem". In: *Psychology today* 2.1, pp. 60–67 (cit. on p. 32).

Mislove, Alan, Massimiliano Marcon, Krishna P Gummadi, Peter Druschel, and Bobby Bhattacharjee (2007). "Measurement and analysis of online social networks". In: *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 29–42 (cit. on p. 30).

Mukherjee, Sach and Terence P Speed (2008). "Network inference using informative priors". In: *Proceedings of the National Academy of Sciences* 105.38, pp. 14313–14318 (cit. on pp. 7, 10, 41, 42, 45).

Musiał, Katarzyna and Przemysław Kazienko (2013). "Social networks on the Internet". In: *World Wide Web* 16, pp. 31–72 (cit. on p. 31).

Myers, Seth and Jure Leskovec (2010). "On the convexity of latent social network inference". In: *Advances in neural information processing systems* 23 (cit. on p. 24).

Myers, Seth A, Aneesh Sharma, Pankaj Gupta, and Jimmy Lin (2014). "Information network or social network? The structure of the Twitter follow graph". In: *Proceedings of the 23rd International Conference on World Wide Web*, pp. 493–498 (cit. on pp. 31, 32).

Naseem, Usman, Imran Razzak, Matloob Khushi, Peter W Eklund, and Jinman Kim (2021). "COVIDSenti: A large-scale benchmark Twitter data set for COVID-19 sentiment analysis". In: *IEEE transactions on computational social systems* 8.4, pp. 1003–1015 (cit. on p. 49).

Nelwamondo, Fulufhelo V, Shakir Mohamed, and Tshilidzi Marwala (2007). "Missing data: A comparison of neural network and expectation maximization techniques". In: *Current Science*, pp. 1514–1521 (cit. on p. 44).

Newman, Mark EJ (2002). "Assortative mixing in networks". In: *Physical review letters* 89.20, p. 208701 (cit. on p. 38).

– (2001). "Clustering and preferential attachment in growing networks". In: *Physical review E* 64.2, p. 025102 (cit. on p. 47).

– (2018a). "Estimating network structure from unreliable measurements". In: *Physical Review E* (cit. on pp. 25, 53).

– (2000). "Models of the small world". In: *Journal of Statistical Physics* 101, pp. 819–841 (cit. on p. 32).

– (2018b). "Network structure from rich but noisy data". In: *Nature Physics* 14, pp. 67–75 (cit. on pp. 7, 9, 13, 22, 46, 48, 52, 66, 67, 77, 82–86, 89–93).

– (2018c). *Networks*. Oxford university press (cit. on pp. 5, 21, 26).

Newman, Mark EJ and Michelle Girvan (Feb. 2004). "Finding and evaluating community structure in networks". In: *Phys. Rev. E* 69 (2), p. 026113 (cit. on p. 31).

Newman, Mark EJ, Duncan J Watts, and Steven H Strogatz (2002). "Random graph models of social networks". In: *Proceedings of the national academy of sciences* 99.suppl_1, pp. 2566–2572 (cit. on p. 6).

Nguyen, Giang Hoang, John Boaz Lee, Ryan A Rossi, et al. (2018). "Continuous-time dynamic network embeddings". In: *Companion proceedings of the the web conference 2018*, pp. 969–976 (cit. on p. 106).

Northcutt, Curtis G, Anish Athalye, and Jonas Mueller (2021). "Pervasive label errors in test sets destabilize machine learning benchmarks". In: *arXiv preprint arXiv:2103.14749* (cit. on p. 114).

Oh Song, Hyun, Yu Xiang, Stefanie Jegelka, and Silvio Savarese (2016). "Deep metric learning via lifted structured feature embedding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4004–4012 (cit. on p. 123).

Onnela, J-P, Jari Saramäki, Jorkki Hyvönen, et al. (2007). "Structure and tie strengths in mobile communication networks". In: *Proceedings of the national academy of sciences* 104.18, pp. 7332–7336 (cit. on pp. 26, 49).

Otte, Evelien and Ronald Rousseau (2002). "Social network analysis: a powerful strategy, also for the information sciences". In: *Journal of information Science* 28.6, pp. 441–453 (cit. on p. 26).

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd (1999). *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford infolab (cit. on p. 29).

Palla, Gergely, Albert-László Barabási, and Tamás Vicsek (2007). "Quantifying social group evolution". In: *Nature* 446.7136, pp. 664–667 (cit. on p. 33).

Papanastasiou, Effrosyni and Anastasios Giovanidis (2021). "Bayesian inference of a social graph with trace feasibility guarantees". In: *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 317–324 (cit. on p. 97).

– (2023). "Constrained expectation-maximisation for inference of social graphs explaining online user–user interactions". In: *Social Network Analysis and Mining* 13.1, p. 41 (cit. on pp. 19, 97).

Peel, Leto, Tiago Peixoto, and Manlio De Domenico (2022). "Statistical inference links data and theory in network science". In: *Nature Communications*, pp. 1–15 (cit. on pp. 7, 22, 23, 52).

Peixoto, Tiago (2019). "Network reconstruction and community detection from dynamics". In: *Physical Review Letters* (cit. on pp. 13, 78, 82–86, 89–91, 93).

– (2018). "Reconstructing networks with unknown and heterogeneous errors". In: *Physical Review X* 8.4, p. 041011 (cit. on pp. 7, 9, 22, 48, 51, 52).

Peixoto, Tiago P and Martin Rosvall (2017). "Modelling sequences and temporal networks with dynamic community structures". In: *Nature communications* 8.1, p. 582 (cit. on p. 33).

Peng, Huan-Kai, Jiang Zhu, Dongzhen Piao, Rong Yan, and Ying Zhang (2011). "Retweet modeling using conditional random fields". In: *2011 IEEE 11th international conference on data mining workshops*. IEEE, pp. 336–343 (cit. on p. 24).

Pereira-Kohatsu, Juan Carlos, Lara Quijano-Sánchez, Federico Liberatore, and Miguel Camacho-Collados (2019). "Detecting and monitoring hate speech in Twitter". In: *Sensors* 19.21, p. 4654 (cit. on pp. 9, 11).

Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena (2014). "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710 (cit. on p. 105).

Pitsilis, Georgios K, Heri Ramampiaro, and Helge Langseth (2018). "Effective hate-speech detection in Twitter data using recurrent neural networks". In: *Applied Intelligence* 48, pp. 4730–4742 (cit. on p. 11).

Poblete, Barbara, Ruth Garcia, Marcelo Mendoza, and Alejandro Jaimes (2011). "Do all birds tweet the same? Characterizing Twitter around the world". In: *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 1025–1030 (cit. on p. 31).

Pólya, George (1937). "Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen". In: (cit. on p. 3).

Priebe, Carey E, Daniel L Sussman, Minh Tang, and Joshua T Vogelstein (2015). "Statistical inference on errorfully observed graphs". In: *Journal of Computational and Graphical Statistics* 24.4, pp. 930–953 (cit. on p. 47).

Qiu, Jiezhong, Jian Tang, Hao Ma, et al. (2018). "Deepinf: Social influence prediction with deep learning". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2110–2119 (cit. on p. 109).

Raschka, Sebastian (2018). "Model evaluation, model selection, and algorithm selection in machine learning". In: *arXiv preprint arXiv:1811.12808* (cit. on p. 101).

Redner, Sid (2008). "Teasing out the missing links". In: *Nature* 453.7191, pp. 47–48 (cit. on p. 45).

Richardson, Matthew and Pedro Domingos (2002). "Mining knowledge-sharing sites for viral marketing". In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 61–70 (cit. on p. 50).

Robbins, Herbert and Sutton Monro (1951). "A stochastic approximation method". In: *The annals of mathematical statistics*, pp. 400–407 (cit. on p. 121).

Robinson, Joshua, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka (2020). "Contrastive learning with hard negative samples". In: *arXiv preprint arXiv:2010.04592* (cit. on p. 123).

Rodriguez, Manuel Gomez, David Balduzzi, and Bernhard Schölkopf (2011). "Uncovering the temporal dynamics of diffusion networks". In: *arXiv preprint arXiv:1105.0697* (cit. on p. 50).

Rosenblatt, Frank (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory (cit. on p. 5).

Ryu, Seongok, Yongchan Kwon, and Woo Youn Kim (2019). "A Bayesian graph convolutional network for reliable prediction of molecular properties with uncertainty quantification". In: *Chemical science* 10.36, pp. 8438–8446 (cit. on p. 109).

Sadri, Arif Mohaimin, Samiul Hasan, Satish V Ukkusuri, and Juan Esteban Suarez Lopez (2017). "Analyzing social interaction networks from twitter for planned special events". In: *arXiv preprint arXiv:1704.02489* (cit. on p. 31).

Saito, Kazumi, Ryohei Nakano, and Masahiro Kimura (2008). "Prediction of information diffusion probabilities for independent cascade model". In: *International conference on knowledge-based and intelligent information and engineering systems*. Springer, pp. 67–75 (cit. on pp. 8, 13, 34, 50, 77, 82–84, 86, 89–93).

Sajjad, Hooman Peiro, Andrew Docherty, and Yuriy Tyshetskiy (2019). "Efficient representation learning using random walks for dynamic graphs". In: *arXiv preprint arXiv:1901.01346* (cit. on p. 106).

Salehinejad, Hojjat, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee (2017). "Recent advances in recurrent neural networks". In: *arXiv preprint arXiv:1801.01078* (cit. on p. 107).

Sales-Pardo, Marta, Roger Guimera, André A Moreira, and Luís A Nunes Amaral (2007). "Extracting the hierarchical organization of complex systems". In: *Proceedings of the National Academy of Sciences* 104.39, pp. 15224–15229 (cit. on p. 45).

Sankar, Aravind, Xinyang Zhang, Adit Krishnan, and Jiawei Han (2020). "Inf-VAE: A variational autoencoder framework to integrate homophily and influence in diffusion prediction". In: *Proceedings of the 13th international conference on web search and data mining*, pp. 510–518 (cit. on p. 110).

Scarselli, Franco, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini (2008). "The graph neural network model". In: *IEEE transactions on neural networks* 20.1, pp. 61–80 (cit. on pp. 6, 103, 109, 115).

Schafer, Joseph L and John W Graham (2002). "Missing data: our view of the state of the art." In: *Psychological methods* 7.2, p. 147 (cit. on p. 23).

Schroff, Florian, Dmitry Kalenichenko, and James Philbin (2015). "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823 (cit. on pp. 119, 123).

Shannon, Claude Elwood (1948). "A mathematical theory of communication". In: *The Bell system technical journal* 27.3, pp. 379–423 (cit. on p. 33).

Sharma, Sagar, Simone Sharma, and Anidhya Athaiya (2017). "Activation functions in neural networks". In: *Towards Data Sci* 6.12, pp. 310–316 (cit. on p. 117).

Shi, Yong, Minglong Lei, Hong Yang, and Lingfeng Niu (2019). "Diffusion network embedding". In: *Pattern Recognition* 88, pp. 518–531 (cit. on p. 107).

Shu, Kai, H Russell Bernard, and Huan Liu (2019). "Studying fake news via network analysis: detection and mitigation". In: *Emerging research challenges and opportunities in computational social network analysis and mining*, pp. 43–65 (cit. on p. 11).

Shu, Kai and Huan Liu (2022). *Detecting fake news on social media*. Springer Nature (cit. on p. 103).

Sprinzak, Einat, Shmuel Sattath, and Hanah Margalit (2003). "How reliable are experimental protein–protein interaction data?" In: *Journal of molecular biology* 327.5, pp. 919–923 (cit. on pp. 8, 23, 41).

Stopczynski, Arkadiusz, Vedran Sekara, Piotr Sapiezynski, et al. (2014). "Measuring large-scale social networks with high resolution". In: *PloS one* 9.4, e95978 (cit. on p. 9).

Stumpf, Michael PH, Thomas Thorne, Eric De Silva, et al. (2008). "Estimating the size of the human interactome". In: *Proceedings of the National Academy of Sciences* 105.19, pp. 6959–6964 (cit. on p. 23).

Tabouy, Timothée, Pierre Barbillon, and Julien Chiquet (2019). "Variational inference for stochastic block models from sampled data". In: *Journal of the American Statistical Association* (cit. on p. 47).

Tang, Jian, Meng Qu, Mingzhe Wang, et al. (2015). "Line: Large-scale information network embedding". In: *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077 (cit. on p. 106).

Tang, John, Mirco Musolesi, Cecilia Mascolo, and Vito Latora (2009). "Temporal distance metrics for social network analysis". In: *Proceedings of the 2nd ACM workshop on Online social networks*, pp. 31–36 (cit. on p. 49).

Trezza, Domenico (2023). "To scrape or not to scrape, this is dilemma. The post-API scenario and implications on digital research". In: *Frontiers in Sociology* 8, p. 1145038 (cit. on pp. 13, 25).

Tymoczko, Thomas (1979). "The four-color problem and its philosophical significance". In: *The journal of philosophy* 76.2, pp. 57–83 (cit. on p. 5).

Van Den Oord, Aaron, Yazhe Li, and Oriol Vinyals (2018). "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (cit. on pp. 111, 120).

Van Der Hofstad, Remco (2013). "Random graphs and complex networks". In: *Lecture notes* (cit. on p. 35).

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, et al. (2017). "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (cit. on p. 118).

Vendeville, Antoine, Anastasios Giovanidis, Effrosyni Papanastasiou, and Benjamin Guedj (2022a). "Opening up echo chambers via optimal content recommendation". In: *International Conference on Complex Networks and Their Applications*. Springer, pp. 74–85 (cit. on pp. 19, 97).

– (2022b). "Recommendation of content to mitigate the echo chamber effect". In: *Conference on Complex Systems* (cit. on pp. 19, 98).

Von Mering, Christian, Roland Krause, Berend Snel, et al. (2002). "Comparative assessment of large-scale data sets of protein–protein interactions". In: *Nature* 417.6887, pp. 399–403 (cit. on pp. 8, 22).

Waldrop, Mitchell M (1993). *Complexity: The emerging science at the edge of order and chaos*. Simon and Schuster (cit. on p. 5).

Wang, Jia, Vincent W Zheng, Zemin Liu, and Kevin Chen-Chuan Chang (2017). "Topological recurrent neural network for diffusion prediction". In: *2017 IEEE international conference on data mining (ICDM)*. IEEE, pp. 475–484 (cit. on p. 108).

Wang, Yongqing, Huawei Shen, Shenghua Liu, Jinhua Gao, and Xueqi Cheng (2017). "Cascade Dynamics Modeling with Attention-based Recurrent Neural Network." In: *IJCAI*. Vol. 17, pp. 2985–2991 (cit. on p. 108).

Wang, Zhitao, Chengyao Chen, and Wenjie Li (2018). "A sequential neural information diffusion model with structure attention". In: *Proceedings of the 27th ACM international conference on information and knowledge management*, pp. 1795–1798 (cit. on p. 108).

– (2019). "Information diffusion prediction with network regularized role-based user representation learning". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)*, pp. 1–23 (cit. on p. 106).

Watts, Duncan J and Steven H Strogatz (1998). "Collective dynamics of 'small-world'networks". In: *nature* 393.6684, pp. 440–442 (cit. on pp. 27, 29, 32, 35).

Weinberger, Kilian Q and Lawrence K Saul (2009). "Distance metric learning for large margin nearest neighbor classification." In: *Journal of machine learning research* 10.2 (cit. on p. 119).

Whitney, Hassler (1932). "Congruent Graphs and the Connectivity of Graphs". In: *American Journal of Mathematics* 54.1, p. 150 (cit. on p. 4).

Wu, Haixia, Chunyao Song, Yao Ge, and Tingjian Ge (2022). "Link prediction on complex networks: An experimental survey". In: *Data Science and Engineering* 7.3, pp. 253–278 (cit. on p. 124).

Wu, Jia, Jiahao Xia, and Fangfang Gou (2022). "Information transmission mode and IoT community reconstruction based on user influence in opportunistic social networks". In: *Peer-to-Peer Networking and Applications*, pp. 1398–1416 (cit. on pp. 8, 51).

Wu, Xiaojian, Akshat Kumar, Daniel Sheldon, and Shlomo Zilberstein (2013). "Parameter learning for latent network diffusion". In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pp. 2923–2930 (cit. on p. 51).

Xia, Feng, Jiaying Liu, Hansong Nie, et al. (2019). "Random walks: A review of algorithms and applications". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 4.2, pp. 95–107 (cit. on p. 105).

Xiao, Shunxin, Shiping Wang, Yuanfei Dai, and Wenzhong Guo (2022). "Graph neural networks in node classification: survey and evaluation". In: *Machine Vision and Applications* 33.1, p. 4 (cit. on p. 103).

Xie, Yaochen, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji (2023). "Self-Supervised Learning of Graph Neural Networks: A Unified Review". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.2, pp. 2412–2429 (cit. on p. 112).

Yang, Chao, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu (2012). "Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter". In: *Proceedings of the 21st international conference on World Wide Web*, pp. 71–80 (cit. on p. 31).

Yang, Cheng, Hao Wang, Jian Tang, et al. (2021). "Full-scale information diffusion prediction with reinforced recurrent networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 34.5, pp. 2271–2283 (cit. on p. 108).

Yang, Jaewon and Jure Leskovec (2010). "Modeling information diffusion in implicit networks". In: *2010 IEEE international conference on data mining*. IEEE, pp. 599–608 (cit. on p. 50).

Ying, Xue (2019). "An overview of overfitting and its solutions". In: *Journal of physics: Conference series*. Vol. 1168. IOP Publishing, p. 022022 (cit. on pp. 102, 117).

You, Jiaxuan, Zhitao Ying, and Jure Leskovec (2020). "Design space for graph neural networks". In: *Advances in Neural Information Processing Systems* 33, pp. 17009–17021 (cit. on pp. 117, 122).

Young, Jean-Gabriel, Giovanni Petri, and Tiago Peixoto (2021). "Hypergraph reconstruction from network data". In: *Communications Physics* 4.1, p. 135 (cit. on p. 51).

Yu, Haiyuan, Pascal Braun, Muhammed A Yıldırım, et al. (2008). "High-quality binary protein interaction map of the yeast interactome network". In: *Science* 322.5898, pp. 104–110 (cit. on p. 23).

Zhang, Xiaoqi, Zi-Ke Zhang, Wenbo Wang, et al. (2021). "Multiplex network reconstruction for the coupled spatial diffusion of infodemic and pandemic of COVID-19". In: *International Journal of Digital Earth* 4, pp. 401–423 (cit. on p. 8).

Zhang, Yingxue, Soumyasundar Pal, Mark Coates, and Deniz Ustebay (2019). "Bayesian graph convolutional neural networks for semi-supervised classification". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01, pp. 5829–5836 (cit. on p. 109).

Zhang, Yingying, Qiaoyong Zhong, Liang Ma, Di Xie, and Shiliang Pu (2019). "Learning incremental triplet margin for person re-identification". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01, pp. 9243–9250 (cit. on p. 119).

Zhang, Yuan, Tianshu Lyu, and Yan Zhang (2018). "Cosine: Community-preserving social network embedding from information diffusion cascades". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32 (cit. on p. 105).

Zhao, Jianan, Qianlong Wen, Mingxuan Ju, Chuxu Zhang, and Yanfang Ye (2023). "Self-supervised graph structure refinement for graph neural networks". In: *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 159–167 (cit. on p. 118).

Zhao, Qihang, Yuzhe Zhang, and Xiaodong Feng (2022). "Predicting information diffusion via deep temporal convolutional networks". In: *Information Systems* 108, p. 102045 (cit. on p. 109).

Zhou, Bangzhu, Xiaodong Feng, and Hemin Feng (2024). "Structural-topic aware deep neural networks for information cascade prediction". In: *PeerJ Computer Science* 10, e1870 (cit. on p. 109).

Zhou, Jie, Ganqu Cui, Shengding Hu, et al. (2020). "Graph neural networks: A review of methods and applications". In: *AI open* 1, pp. 57–81 (cit. on p. 118).

Zhou, Tao, Linyuan Lü, and Yi-Cheng Zhang (2009). "Predicting missing links via local information". In: *The European Physical Journal B* 71, pp. 623–630 (cit. on p. 47).

Appendix for Chapter 4

A

A.1 Detailed derivations for the equations of CEM-ER

For the E-step, we modify the Newman algorithm by taking the expectation over the set of random variables Y_{ij} at both sides of (4.10):

$$\begin{aligned}\mathbb{E}[\log P(\theta \mid \mathcal{T})] &\geq \mathbb{E}[\sum_{\mathbf{A}} q(\mathbf{A}) \log \frac{P(\mathbf{A}, \theta \mid \mathcal{T})}{q(\mathbf{A}}] \\ &= \sum_{\mathbf{A}} q(\mathbf{A}) \Big(\mathbb{E}[\log P(\mathbf{A}, \theta \mid \mathcal{T})] - \log q(\mathbf{A}) \Big). \end{aligned} \tag{A.1}$$

To find $\mathbb{E}[\log P(\mathbf{A}, \theta \mid \mathcal{T})]$, we replace (4.9) into (4.8). Setting $\Gamma = P(\theta)/P(\mathcal{T})$, the expectation of the log of (4.8) becomes:

$$\begin{aligned}\mathbb{E}[\log P(\mathbf{A}, \theta \mid \mathcal{T})] = log\Gamma + \sum_{i \neq j} \Big[A_{ij} \Big(\log \rho + \mathbb{E}[Y_{ij}] \log \alpha + \\ + (M_{ij} - \mathbb{E}[Y_{ij}]) \log (1 - \alpha) \Big) + (1 - A_{ij} \Big(\log(1 - \rho) + \\ + \mathbb{E}[Y_{ij}] \log \beta + (M_{ij} - \mathbb{E}[Y_{ij}]) \log (1 - \beta) \Big) \Big]. \end{aligned} \tag{A.2}$$

Then, by replacing (4.7) into (A.2), and then (A.2) into (A.1), we get:

$$\mathbb{E}[\log P(\theta \mid \mathcal{T})] \geq \sum_{\mathbf{A}} q(\mathbf{A}) \log \frac{D_{ij}}{q(\mathbf{A})} \tag{A.3}$$

$$\begin{aligned}\text{where, } D_{ij} = \Gamma \prod_{i \neq j} \Big[\rho \alpha^{M_{ij}\sigma_{ij}} (1 - \alpha)^{M_{ij}(1-\sigma_{ij})} \Big]^{A_{ij}} \\ \times \Big[(1 - \rho) \beta^{M_{ij}\sigma_{ij}} (1 - \beta)^{M_{ij}(1-\sigma_{ij})} \Big]^{1 - A_{ij}}. \end{aligned} \tag{A.4}$$

For the M-step of the EM algorithm, the function that we want to maximize is $\mathbb{E}[\log P(\theta \mid \mathcal{T})]$. To do so, we need to find the unknown values, $q(\mathbf{A})$ and $\theta = \{\alpha, \beta, \rho, \boldsymbol{\sigma}\}$, that maximize the expectation on the left-hand side of (4.11), under the feasibility constraints on the parameters set θ. From these, only the σ_{ij} have an important constraint set, specified in (4.5) and (4.6).

Solution with respect to $q(\mathbf{A})$. We notice that the choice of $q(\mathbf{A})$ that achieves equality (i.e. maximizes the right-hand side) in (4.11) is:

$$q(\mathbf{A}) = \frac{D_{ij}}{\sum_{\mathbf{A}} D_{ij}}, \tag{A.5}$$

which leads us to Eq. (4.13).

Solution with respect to α, β, ρ. To maximize the right-hand side of (4.11) in terms of parameter α we differentiate it with respect to α and then setting it equal to zero (while holding σ_{ij}, q constant):

$$\sum_{i \neq j} Q_{ij} M_{ij} \left(\frac{\sigma_{ij}}{\alpha} - \frac{1 - \sigma_{ij}}{1 - \alpha} \right) = 0. \tag{A.6}$$

After rearranging, we get the updates shown in Eq. (4.15), and we repeat likewise for β and ρ.

Solution with respect to σ_{ij}. If we take into account that $Q_{ij} = \sum_{\mathbf{A}} q(\mathbf{A}) A_{ij}$ and also that $\sum_{\mathbf{A}} q(\mathbf{A}) = 1$, by rearranging the right-hand side of (4.11), the problem becomes equivalent to maximizing:

$$\begin{aligned} &\sum_{\mathbf{A}} q(\mathbf{A}) \sum_{i \neq j} \sigma_{ij} M_{ij} \left(A_{ij} \log \frac{\alpha}{1 - \alpha} + (1 - A_{ij}) \log \frac{\beta}{1 - \beta} \right) \\ &\quad = \sum_{i \neq j} \sigma_{ij} M_{ij} \left(Q_{ij} \log \frac{\alpha}{1 - \alpha} + (1 - Q_{ij}) \log \frac{\beta}{1 - \beta} \right). \end{aligned} \tag{A.7}$$

This leads us to the constrained optimization problem of Eq. (4.17).

A.2 Detailed derivations for the equations of CEM-SBM

For the E-step of the EM algorithm, we modify the Newman algorithm by taking the expectation over the set of random variables Y_{ij} at both sides of (4.20):

$$\begin{aligned} &\mathbb{E}[\log P(\theta \mid \mathcal{T})] \geq \mathbb{E}[\sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \log \frac{P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})}{q(\mathbf{A}, \mathbf{g})}] \\ &= \sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \Big(\mathbb{E}[\log P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})] - \log q(\mathbf{A}, \mathbf{g}) \Big). \end{aligned} \tag{A.8}$$

To find $\mathbb{E}[\log P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})]$, we replace (4.19) into (4.18). Setting $\Gamma = P(\theta)/P(\mathcal{T})$, the expectation of the log of (4.18) becomes:

$$\mathbb{E}[\log P(\mathbf{A}, \mathbf{g}, \theta \mid \mathcal{T})] = log\Gamma + \sum_{\substack{i \neq j \\ g_i = gj}} \Big[A_{ij} \Big(\log p + \mathbb{E}[Y_{ij}] \log \alpha +$$

$$+ (M_{ij} - \mathbb{E}[Y_{ij}]) \log (1 - \alpha) \Big) + (1 - A_{ij}) \Big(\log(1 - p) +$$

$$+ \mathbb{E}[Y_{ij}] \log \beta + (M_{ij} - \mathbb{E}[Y_{ij}]) \log (1 - \beta) \Big) \Big] + \sum_{\substack{i \neq j \\ g_i \neq gj}} \Big[A_{ij} \Big(\log q +$$

$$+ \mathbb{E}[Y_{ij}] \log \alpha + (M_{ij} - \mathbb{E}[Y_{ij}]) \log (1 - \alpha) \Big) + (1 - A_{ij}) \Big(\log(1 - q) +$$

$$+ \mathbb{E}[Y_{ij}] \log \beta + (M_{ij} - \mathbb{E}[Y_{ij}]) \log (1 - \beta) \Big) \Big]. \tag{A.9}$$

By replacing (4.7) into (A.9), and then (A.9) into (A.8), we get:

$$\mathbb{E}[\log P(\theta \mid \mathcal{T})] \geq \sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \log \frac{D(\mathbf{A}, \mathbf{g})}{q(\mathbf{A}, \mathbf{g})}, \tag{A.10}$$

where,

$$D(\mathbf{A}, \mathbf{g}) = \Gamma \prod_{\substack{i \neq j \\ g_i = gj}} \Big[p \alpha^{M_{ij} \sigma_{ij}} (1 - \alpha)^{M_{ij}(1 - \sigma_{ij})} \Big]^{A_{ij}}$$

$$\Big[(1 - p) \beta^{M_{ij} \sigma_{ij}} (1 - \beta)^{M_{ij}(1 - \sigma_{ij})} \Big]^{1 - A_{ij}} \prod_{\substack{i \neq j \\ g_i \neq gj}} \Big[q \alpha^{M_{ij} \sigma_{ij}} (1 - \alpha)^{M_{ij}(1 - \sigma_{ij})} \Big]^{A_{ij}}$$

$$\Big[(1 - q) \beta^{M_{ij} \sigma_{ij}} (1 - \beta)^{M_{ij}(1 - \sigma_{ij})} \Big]^{1 - A_{ij}}. \tag{A.11}$$

For the M-step of EM, we maximize the expectation $\mathbb{E}[\log P(\theta \mid \mathcal{T})]$ as we did in the CEM-er prior.

Solution with respect to $q(\mathbf{A}, \mathbf{g})$. We notice that the choice of $q(\mathbf{A}, \mathbf{g})$ that achieves equality (i.e. maximizes the right-hand side) in (A.10) is:

$$q(\mathbf{A}, \mathbf{g}) = \frac{D(\mathbf{A}, \mathbf{g})}{\sum_{\mathbf{A}} D(\mathbf{A}, \mathbf{g})}. \tag{A.12}$$

From (A.12), in a similar fashion to Newman's method [Eq. (13), 20], and because Γ cancels out, we get:

$$q(\mathbf{A}, \mathbf{g}) = \prod_{i \neq j, (g_i = g_j)} Q_{ij}(g_i, g_j)^{A_{ij}} (1 - Q_{ij}(g_i, g_j))^{1 - A_{ij}} \prod_{i \neq j, (g_i \neq g_j)} Q_{ij}(g_i, g_j)^{A_{ij}} (1 - Q_{ij}(g_i, g_j))^{1 - A_{ij}}. \tag{A.13}$$

Hence, given Eq. (A.11), the values of Q_{ij} are found to be the ones in Eq. (4.21) and (4.22). Our goal is to find the unknown parameters $\theta = \{\alpha, \beta, p, q, \boldsymbol{\sigma}\}$ that maximize the right-hand size of (A.10), given the maximising distribution for $q(\mathbf{A}, \mathbf{g})$ in (A.12), hence given the values of $Q_{ij}(g_i, g_j)$ in (A.13).

Solution with respect to $\boldsymbol{\alpha, \beta, p, q}$. To maximize the right-hand side of (A.10) in terms of parameter α, we differentiate the equation with respect to α and we set it equal to zero (while holding the rest of the parameters θ constant):

$$\sum_{i \neq j} Q_{ij}(g_i, g_j) M_{ij} \left(\frac{\sigma_{ij}}{\alpha} - \frac{1 - \sigma_{ij}}{1 - \alpha} \right) = 0. \tag{A.14}$$

After rearranging, we get the value in Eq. (4.23). By repeating the same procedure for β, we get Eq. (4.24). Likewise, differentiating the r.h.s. of (A.10) with respect to p and then setting it equal to zero we get:

$$\sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \sum_{\substack{i \neq j \\ g_i = gj}} \left(\frac{A_{ij}}{p} - \frac{1 - A_{ij}}{1 - p} \right) = 0. \tag{A.15}$$

This is how we get the updates for p in Eq. (4.25), and, likewise, for q in Eq. (4.26).

Solution with respect to $\boldsymbol{\sigma}_{ij}$. If we take into account that $Q_{ij}(g_i, g_j) = \sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) A_{ij}$ and also that $\sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) = 1$, by rearranging the right-hand side of (A.10), the problem becomes equivalent to maximizing:

$$\sum_{\mathbf{A}} q(\mathbf{A}, \mathbf{g}) \sum_{i \neq j} \sigma_{ij} M_{ij} \left(A_{ij} \log \frac{\alpha}{1 - \alpha} + (1 - A_{ij}) \log \frac{\beta}{1 - \beta} \right) = \sum_{i \neq j} \sigma_{ij} M_{ij} \left(Q_{ij}(g_i, g_j) \log \frac{\alpha}{1 - \alpha} + (1 - Q_{ij}(g_i, g_j)) \log \frac{\beta}{1 - \beta} \right). \tag{A.16}$$

This leads us to the optimization problem of Eq. (4.27) through which we can find the σ_{ij} values.

* * *

Appendix for Chapter 5

B

In this section of the Appendix, we provide more detailed information on how we configured the algorithms to obtain the reported results. For Node2Vec, we use the Python implementation already provided by https://github.com/eliorc/node2vec. We use dimensions=8 (we found that increasing them penalized Precision), walk_length=3, num_walks=30 and for the model fit we use window=4, min_count=1, and batch_words=1000. The code for our model FeasCL was written using Python and PyTorch, a machine learning library that is very commonly used in ML applications. To generate the results for FeasCL reported in Table 5.3, we experimented with various parameters affecting the training process. The description of these parameters is given in Table B.1.

First, since we assumed no features were available for the user nodes, we had to randomly initialize their sender and receiver embeddings. We observed that increasing the initial dimension INPUT_DIM had a positive effect on the results. This might suggest that a higher dimension allows the algorithm more freedom to learn within the latent space. Consequently, the most optimal results (FeasCL-5k) were achieved with a very large dimension of 1500. We also experimented with a smaller dimension of 500 (as indicated in the last three lines of Table B.1). As shown, in this case, the Precision slightly decreases. Additionally, we

Tab. B.1. Configuration Parameters for the FeasCL model

Parameter	Type	Description
–INPUT_DIM	int	Size of initial dimensions
–OUTPUT_DIM	int	Size of output dimensions
–TEMP	bool	Use of temperature
–TPOS	float	Temperature for positive samples
–TNEG	float	Temperature for negative samples
–NUM_POS_SAMPLES	int	Number of positives samples for each node
–NUM_NEG_SAMPLES	int	Number of negative samples for each node
–NUM_HARD_NEG_SAMPLES	int	Number of hard negative samples
–BATCH_SIZE	int	Number of batches
–LR	float	Learning rate
–EPOCHS	float	Number of epochs

set the dimensionality of the sender and receiver embeddings for the output (OUTPUT_DIM) to 375 each.

Next, we needed to decide whether to incorporate a temperature parameter for the contrastive loss, choosing between Eq. 5.1 and Eq. 5.2. We discovered that setting a global temperature for both types of samples to $\tau = 0.10$ significantly improved the results. For the loss function, we selected a learning rate (LR) of 0.01, with 5000 EPOCHS, and a BATCH_SIZE of 640.

In the sampling process for positive and negative samples, we first selected a diffusion episode $\mathcal{D}_S$ where the anchor appears. Then, we chose NUM_POS_SAMPLES=1 positive sample and NUM_NEG_SAMPLES=2000 negative samples for each anchor. We found that the best strategy for sampling these negatives was to include users who have interacted in the same diffusion episode $\mathcal{D}_s$ as the anchor, but their connection is not feasibly possible, as well as the users who have not interacted at all with it (Algorithm 2). We selected these numbers to reflect the higher chance for a user to be a negative than a positive neighbor (given the sparsity of OSNs). We also found that maintaining a hard negative sampling strategy (see Section 5.5.1) consistently had a positive effect on the results. Consequently, for each anchor, out of the 2000 negatives, we hard sample the NUM_HARD_NEG_SAMPLES=50 users that are the most similar to the anchor in terms of the learned embeddings.

For the ENCODER model, we used 3 GRAPHSAGE layers (using the SAGECONV PyTorch operator), with RELU as activation function and a dropout strategy (see Algorithm 4). The input graph structure $\mathcal{G}_D$ is built from $\mathcal{D}$ by drawing an edge (i, j) from i to j if $t^s(i) < t^s(j)$ for at least one diffusion episode $\mathcal{D}_s$. This might be considered a weakness of the model, since we had to provide a deterministic graph structure as input, which is not realistic enough. This is why we selected GRAPHSAGE which has the ability to *sample* the node neighbors, instead of using full neighborhoods (Hamilton et al., 2017).

* * *

Algorithm 2 Function SAMPLE_NEIGHBORS for sampling contrastive user pairs

Require: $\mathcal{U}$: Set of users
Require: $\mathcal{D}$: Set of diffusion episodes
for ANCHOR in $\mathcal{U}$ **do**
  $\mathcal{D}_s \leftarrow \{\mathcal{D}_s \in \mathcal{D} : \text{ANCHOR} \in \mathcal{D}_s\}$
  POSITIVES $\leftarrow \{j \in \mathcal{D}_s : t^s(j) > t^s(\text{ANCHOR})\}$
  NEGATIVES $\leftarrow \{j \in \mathcal{U} : j \notin \text{POSITIVES}\}$
  return (ANCHOR, POSITIVES , NEGATIVES)
end for

Algorithm 3 Training epoch for FeasCL

Require: $\mathbf{x}^s$, $\mathbf{x}^r$: initialized sender, receiver embeddings
for ANCHORS, POSITIVES, NEGATIVES in BATCH(SAMPLE_NEIGHBORS) **do**
  $\mathbf{z}^s$ = ENCODER($\mathbf{x}^s$)
  $\mathbf{z}^r$ = ENCODER($\mathbf{x}^r$)
  $\mathbf{z}_{\text{ANCHORS}} = \mathbf{z}^s[\text{ANCHORS}]$
  $\mathbf{z}_{\text{POSITIVES}} = \mathbf{z}^r[\text{POSITIVES}]$
  $\mathbf{z}_{\text{NEGATIVES}} = \mathbf{z}^r[\text{NEGATIVES}]$
  LOSS = FEASCL_LOSS($\mathbf{z}_{\text{ANCHORS}}, \mathbf{z}_{\text{POSITIVES}}, \mathbf{z}_{\text{NEGATIVES}}$) according to Eq. 5.2
  LOSS.BACKWARD()
  OPTIMIZER.STEP()
end for

Algorithm 4 ENCODER function

Require: $\mathbf{x}$: node embeddings
Require: $\mathcal{G}_D$: input graph
$\mathbf{x}$ = GRAPHSAGE($\mathbf{x}$, $\mathcal{G}$)
$\mathbf{x}$ = RELU($\mathbf{x}$)
$\mathbf{x}$ = DROPOUT($\mathbf{x}$)
$\mathbf{x}$ = GRAPHSAGE($\mathbf{x}$, $\mathcal{G}$)
$\mathbf{x}$ = RELU($\mathbf{x}$)
$\mathbf{x}$ = DROPOUT($\mathbf{x}$)
$\mathbf{x}$ = GRAPHSAGE($\mathbf{x}$, $\mathcal{G}$)
return $\mathbf{x}$