Learning Influence Representations : Methods and Applciations

Georgios Panagopoulos

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAX014

Thèse de doctorat

ÉCOLE POLYTECHNIQUE

IP PARIS

Learning Influence Representations: Methods and Applications

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Ecole Polytechnique

École doctorale n°626 Ecole Doctorale IP Paris (ED IP Paris)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 2/2/2022, par

PANAGOPOULOS GEORGIOS

Composition du Jury :

Ioana Manolescu Senior Research Scientist, CEDAR project-team, Inria Saclay	Présidente
Manuel Gomez Rodriguez PhD Faculty (W2), Max Planck Institute for Software Systems	Rapporteur
Aditya Prakash Associate Professor, School of Computational Science and Engineering, Georgia Institute of Technology	Rapporteur
Francesco Bonchi Scientific Director, Institute for Scientific Interchange Foundation	Examinateur
Ryan Rossi Senior Research Scientist, Adobe Research	Examinateur
Georgios Stamou Associate Professor, Department of Electrical and Computer Engineering, National Technical University of Athens	Examinateur
Michalis Vazirgiannis Professor, Laboratoire d'Informatique de l'Ecole polytechnique	Directeur de thèse
Fragkiskos Malliaros Associate Professor, CentraleSupelec, Paris-Saclay University	Co-directeur de thèse

ABSTRACT

Online influence is the plinth of the social networks' effect in our lives and its impact has been steeply increasing. From viral marketing to political campaigns and from news adoption to disease transmission, the way we are influenced by others is more prevalent than ever. In this thesis, we address the problem of efficiently learning and analyzing influence representations for numerous graph mining problems that are apropos.

The first half of the thesis is devoted to the problem of influence maximization, an NP-hard combinatorial optimization problem. The aim is to find the nodes in a network that can maximize the spread of information, where the spread is typically defined by random influence probabilities and simple diffusion models. To address this, in the thesis' first part, we devise a node representation learning model based on diffusion cascades along with an adaptation of a traditional influence maximization algorithm that utilizes the output of the model. This framework surpasses competitive methods, evaluated in terms of computational time and the influence of the predicted seeds in cascades of the immediate future.

The second part is devoted to learning how to perform influence maximization. We develop a graph neural network that inherently parameterizes an upper bound of influence estimation, and train it on small simulated graphs. We experimentally show that it can provide accurate estimations faster than the alternatives for graphs 10 times larger than the train set. Furthermore, we use the models' predictions and representations to propose three new influence maximization methods. An adaptation of Cost Effective Lazy Forward that surpasses SOTA but with significant computational overhead, a Q-learning model that learns to retrieve seeds sequentially, and a submodular function that acts as proxy for the marginal gain and can be optimized adaptively and greedily with a theoretical guarantee. The latter strikes the best balance between efficiency and accuracy in our experiments.

In the second half of the thesis, we focused on specific applications of influence in real data. In the third part we approach epidemic forecasting using influence learning. We utilize the inherent message passing of Graph Neural Networks to learn node representations based on mobility networks of a country's regions and the history of the disease progression. These representations aim to capture how the epidemic diffuses through regions, and are used to predict the number of new COVID-19 cases with a forecasting window of up to 14 days. Furthermore, to capitalize on the lag of the COVID-19 spreading between countries, a meta-learning algorithm is proposed to transfer knowledge between models trained in some countries' whole epidemic circle, to a model predicting cases for another country at the start of the outbreak, where the available training data is limited. Our approach outperforms baseline, time-series, and other deep learning models.

In the final part, we analyze different versions of academic influence and device methods to quantify and predict it. Initially we utilize the Microsoft Academic Graph to build an author-citation network with billions of edges. We subsample it and perform directed-core decomposition to quantify it and visualize it through an interactive web-app. Subsequently we experiment with classifying the h-index of an author based on a GNN on her coauthorship graph and the text of her papers. We conclude the thesis with future directions regarding learning-based influence maximization with heterogeneous data and efficient neural network training through submodular active learning.

PUBLICATIONS

The following publications and submissions under review are included in parts or in an extended version in this thesis:

[1] Giannis Nikolentzos, George Panagopoulos, Iakovos Evdaimon, and Michalis Vazirgiannis. *Can Author Collaboration Reveal Impact? The Case of h-index*. 2021.

[2] George Panagopoulos, Fragkiskos D Malliaros, and Michalis Vazirgianis. "Influence maximization using influence and susceptibility embeddings." In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 14. 2020, pp. 511–521.

[3] George Panagopoulos, Fragkiskos Malliaros, and Michalis Vazirgiannis. "Multi-task learning for influence estimation and maximization." In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[4] George Panagopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. "Transfer Graph Neural Networks for Pandemic Forecasting." In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. 2021.

[5] George Panagopoulos, Nikolaos Tziortziotis, Fragkiskos D Malliaros, and Michalis Vazirgiannis. "Learning to Maximize Influence." In: *arXiv preprint arXiv:2108.04623* (2021).

[6] George Panagopoulos, Christos Xypolopoulos, Konstantinos Skianis, Christos Giatsidis, Jie Tang, and Michalis Vazirgiannis. "Scientometrics for success and influence in the microsoft academic graph." In: *International Conference on Complex Networks and Their Applications*. Springer. 2019, pp. 1007–1017.

During my Ph.D., I worked on other problems. The following publications were part of my research but are not covered in this thesis as the topics are outside the scope of the covered material:

[1] Paul Boniol, George Panagopoulos, Christos Xypolopoulos, Rajaa El Hamdani, David Restrepo Amariles, and Michalis Vazirgiannis. "Performance in the courtroom: Automated processing and visualization of appeal court decisions in france." In: *arXiv preprint arXiv:2006.06251* (2020).

[2] Giannis Nikolentzos, George Panagopoulos, and Michalis Vazirgiannis. "An Empirical Study of the Expressiveness of Graph Kernels and Graph Neural Networks." In: (2021).

[3] George Panagopoulos and Hamid Jalalzai. "Graph Neural Networks with Extreme Nodes Discrimination." In: *Deep Learning for Graphs Workshop at KDD 2020*. 2018.

[4] George Panagopoulos, Fragkiskos D Malliaros, and Michalis Vazirgiannis. “Diffugreedy: An influence maximization algorithm based on diffusion cascades.” In: *International Conference on Complex Networks and their Applications*. Springer. 2018, pp. 392–404.

[5] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Nicolas Collignon, and Rik Sarkar. “PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models.” In: *arXiv preprint arXiv:2104.07788* (2021).

ACKNOWLEDGMENTS

Much to the accordance of the current dissertation, the influence of others throughout our professional, social and personal lives is prevalent, and I am no exception. Throughout these three years of my PhD, and 5 years overall of graduate studies, I had the opportunity to collaborate and interact with some very interesting people, who I believe have contributed to this thesis in their own way.

First and foremost, I will have to thank my primary supervisor, Prof. Vazirgiannis, who believed in me and supported me every step of the way. He was patient enough to hear all my ideas, and experienced enough to provide excellent feedback everytime. Most importantly he never put excessive pressure on me, but rather inspired me to deepen my knowledge and solidify my understanding of concepts, such that my work is as complete as possible. Supervising during the pandemic was particularly hard due to the physical distance, but Prof. Vazirgiannis always found a way to clarify things for me and ease my anxiety.

In addition, I have to thank my co-supervisor, Prof. Fragkiskos Malliaros, without whom I would have never decided to come to Paris and spend some of the best years of my life. Fragkiskos' ingenuity in choosing problems/methods and endurance through overnights before deadlines were an integral part of my dissertation. Moreover, Fragkiskos convinced me about the virtues of teaching through his exquisite example.

I would also like to express my gratitude to the prominent researchers who comprise my Ph.D. thesis defense committee for their valuable comments during my defence. Specifically, the reviewers of the dissertation had a significant contribution in the improvement of the manuscript through their insightful comments.

During all these years I was fortunate enough to work with amazing people, whose list is too big, so I will restrict my acknowledgements to the most recent. I have to start with Dr. Giannis Nikolentzos, who inspired me to deepen my knowledge of machine learning and improve my scientific methods. Apart from Giannis, I had the pleasure to collaborate with Dr Hamid Jalalzai,Dr Konstantinos Skianis, Dr Nikolaos Tziortziotis, Dr. Johannes Lutzeyer and Christos Xypolopoulos, each of whom helped me to improve my way of thinking and expand my horizons. I also have to thank all my prior colleagues in University of Houston, the National Center of Scientific Research in Athens. In addition, my professors in Harokopio University of Athens and my diseased teacher and friend, Manolis, had a substantial impact in the research paths I chose to follow.

In the broader aspect of life, the people I met throughout these years outside of work helped me reshape my values and made me more compassionate, wise and toughtfull. I have to thank Maria first, for her love and support throughout this PhD. Thank you for being there and helping me with everything. I also have to thank my "parisian"

friends, who helped me retain my mental health after paper rejections: Zacharias, George, Dimitris, George, Kyriakos and many more that I can not mention due to limited space. Furthermore, I have to thank my Houston friends: Nikos, Eleni, Christos and Michalis with whom I spent my MS. Last but not least, my friends from Greece, who have been with me despite my absence, through video calls, short trips and summer vacations: Alexandros, Nikos, Dimitris and Spyros.

Finally, I want to thank wholeheartedly my familiy, Anastasia, Christos and Konstantina, as without their unconditional love and support throughout all these years, my PhD would be impossible, and I would be a different, sorrowful person. This dissertation is dedicated to all of them, as well as the rest of the family, my aunts, uncles and grandparents.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 MOTIVATION

Online social networks have come to play a substantial role in numerous economical, social and political events, exhibiting their effect on shaping public opinion [5, 10, 147]. The core component of their potential in micro level is how a user influences another user online, which is depicted in the users' connections and activity [193]. Formally, social influence is defined as a directed measure between two users and represents how possible is for the target user to adapt the behavior or copy the action of the source user. Influence is the commodity that governs information passing through the network [119], and can thus be considered the central tube from which the effect of online social networks passes on to the real world. In other words, without users influencing each other in multiple ways, social networks would be meaningless.

The immense gravity of social media is depicted in the ever increasing number of users that has now reached almost 4 billion[189]. The effect of social influence has been studied for more than a decade, starting for commercial purposes such as for viral marketing campaigns, where the influence between users is utilized as a means to advertise a product in an indirect manner [11]. As the use of social media for news sharing became more prominent, viral sharing throughout the network has come to transfer news faster than traditional journalism [90]. Moreover, studies have revealed that influence has a significant effect on the adoption of social responsibility e.g. the effect of advertising for voting was more prevalent when the user is informed about friends who have voted [20]. Quantifying the online influence between ideologically dissimilar friends was also indicative of the users' openness to politically different opinions [15]. More recently, social networks helped sustain communication between isolated people during the COVID-19 pandemic, and social media accounted for a breadth of their entertainment. The data stemming from social network cell phone applications, such as mobility and population density indices, have improved our understanding and prediction of the epidemic spread [62].

On the other side of the spectrum, the negative impact of social networks has been more prevalent than ever. Along with the easier news transmission came a trend for sharing unverified fake news [210], which has became one of the most serious modern problems in social networks with immense implications, calling for a revolution in journalism [220]. Simultaneously, this fast-pace transmission of unverified information along with targeted advertisements allows political campaigns to shape public opinion in unprecedented volumes [172]. Another example of negative influence is cyber bullying, which goes hand in hand with the overall toxicity in online discussions and is known to impose a dramatic effect on the youth [89]. These problems call for better identification and

regulatory strategies, that can result from deeper understanding of the influence mechanisms. For example influence maximization algorithms have been utilized as counter strategies to misinformation spreading [30] and to balance the users' exposure to politically diverse news and advertisements [202].

These applications are indicative of the importance on analyzing and using influence in social networks, which leads unavoidably to the hinders of the task. Firstly, exact influence quantification is an elusive concept for two reasons. The first is that homophily can account for the effects of influence [184] i.e. two users might perform similar actions not solely due to coping, but also because their inherent similarity caused their connection in the first place. The second is exogenous stimulants that can not be measured and hence can be an overlooked confounder to the user's activity. The majority of computational methods, including ours, adopt these assumptions and work with a corresponding approximation of influence. Influence is inherently a temporal [14] and stochastic concept, which requires sequential probabilistic methods [179]. Moreover, social networks are a representative example of big data, demanding for scalable solutions. Furthermore, the online presence of a user in the network can be multifaceted e.g. profile information, text or image posts calling for multimodal approaches. Although machine learning has proven effective in such settings, it lacks theoretical guarantees and principled solutions, which have been prevailing in influence algorithms and influence analysis.

This dissertation aims to tackle some of these challenges. We address methods and applications that revolve around influence in various types of networks. Our work aspires to improve the current methods in terms of accuracy and scalability using state-of-the-art representation learning methods. In the subsequent sections, we give a brief overview of the central concepts and clarify the current problems. We than delineate the contributions of the thesis and give an outline.

1.2 INFLUENCE & EPIDEMICS

Influence can be direct e.g. sending a news post to a friend or indirect e.g. buying a product and collaborative filtering recommending the product to a friend. The social network users are represented as nodes in a graph and the edges depict a relationship, like following, friendship and coauthorship. Influence is used to simulate how information flows through the network using a stochastic diffusion model, such as the independent cascade [106]. It can also be used to rank users in order to find who spreads information more effectively in the network. Moreover, these simulations serve as means of evaluation on the potential of graph-theoretic metrics as faster substitutes for influence [139, 151]. This stems from the fact that evaluating the influence of every node in a large graph to rank them, is computationally demanding and in most cases infeasible.

In many cases, simple ranking is not enough to spread information effectively. If we want to find the optimal set of nodes, we need to

maximize their combined reach, the well-known influence maximization problem [106]. The aim is to find the optimal set of k users that would maximize the influence spread i.e. the total number of users influenced, if a diffusion simulation starts from that set. The complexity of the problem pertains to the influence overlap between seeds, which has to be minimized while searching for the strongest seeds. This problem lies in the heart of opinion shaping [20] , diminishing misinformation [30], mediating online polarization [202] and many pressing problems.

The analysis of influence relationships is not contained solely in the online world. The study of influence dynamics shares similarities with epidemiology, enough so that the initial diffusion models used for information diffusion where based on epidemic models. The analogy can be clarified if we think of a person having a certain probability of influencing another person as similar to the probability of infecting the other person with a disease if they come in close contact. Analogously, the probability of a user adopting a certain opinion or getting infected by an epidemic, is analogous to the amount of its connections that are already infected and the nature/intensity of their interactions. Methodologies pertaining to influence can thus be utilized for strategies of epidemic containment [222] or epidemic forecasting at different scales [103].

1.3 PROBLEMS

1.3.1 *Influence probabilities*

One of the main problems in the influence maximization literature pertains to the standard of utilizing random or uniform influence probabilities in diffusion models. Tha majority of the literature focuses on overcoming the computational burdain that we analyze further below, while overlooking the importance of accurate influence probabilities. Experiments have shown that such approaches produce a less realistic influence spread compared to simulation models with empirical influence probabilities [9]. These probabilities can be computed using real traces of information i.e. diffusion cascades such as retweets or reshares, which actually indicate which users copy each other. Several models have been developed to measure effectively this quantity for tasks like predicting the next node that will appear in a diffusion [124, 216], the next time that someone will copy the diffused content [52, 97] or the overall course of the spreading [27, 66, 79]. However, there is an open problem pertaining on how to utilize such neural-network-based models of learning influence in the context of influence maximization.

1.3.2 *Diffusion models*

The diffusion models suffer from fundamentally oversimplifying assumptions that ignore several characteristics of real cascades [70, 169]. Specifically, the interactions are assumed to take place in discrete synchronous steps. In reality nodes act at different times and copying times/burstiness effects play an important role in quantifying influence

spreading [104]. Moreover, the diffusion models overlook higher order influence effects because of their Markovian mechanism of influence propagation. Meaning, the probability of influencing a node depends only on the direct neighbors, which is not fully accurate, since the history of the cascade has proven beneficial in predicting the participation of specific nodes [41, 97, 124, 213, 230]. Furthermore, the sensitivity of such models to these parameters has been found significant [53], which explains empirical results on the disagreement between simulated influence spread and actual diffusion cascades [177]. Overall we can hypothesize that in the presence of diffusion cascades and an underlying network, model-independent approaches to influence maximization could potentially exhibit more realistic solutions.

1.3.3 *Efficiency*

The problem with efficiency pertains to two aspects. Influence maximization itself is proved to be NP-hard, from a reduction to the set-cover [106], for the most prominent diffusion models. This is the first source of inefficiency, that is addressed using greedy solutions. Additionally, the influence estimation problem that is embedded in influence maximization, i.e. estimating the number of nodes influenced by a given seed set is #P-hard. Specifically, it is analogous to counting simple paths and would require $2^{|E|}$ possible combinations to compute exactly, where $|E|$ is the number of network edges [37]. Typically, influence estimation is approximated using repetitive Monte-Carlo simulations of the independent cascade diffusion model. This is problematic, as this process has to rerun for every candidate seed in every step of building the seed set, rendering the basic model [106] unable to scale in graphs with more than thousands of edges. Several scalable algorithms [22, 196] and heuristics [38, 101] have been proposed capitalizing on sketches and the structure of the graph to produce more efficient solutions. A central hypothesis of the thesis is whether we can utilize machine learning to accelerate influence maximization.

1.4 APPLICATIONS

1.4.1 *Epidemic forecasting*

During a pandemic, accurately predicting the spread of the infection is of paramount importance to governments and policymakers in order to impose measures to combat the spread of the virus or decide on the allocation of healthcare resources. The majority of the approaches rely either on epidemic models [243] or on time series methods and the history of the outbreak. Given the severity of the problem and the need for accurate forecasting of the disease spread, machine learning, and artificial intelligence approaches have emerged as a promising methodology to combat epidemics such as COVID-19. However, the problem is rather challenging because of the incosistencies and burts that are present in the time series of positive cases [234]. Since the

spreading of a pandemic is a spatiotemporal effect, we would like to couple the history of the disease with complementary data that indicate the spatial spreading, which would allow to capture latent cases that emerge as bursts in the time series.

1.4.2 *Academic Influence*

Influence relationships in the academic world impacts research fundings, industrial lab hiring and overall the path of scientific research. Generally, the effect of preferential attachement is well known in the academic world, signifying how prominent researchers tend to have more citations and collaborations [2], hence shaping the scientific landscape with their ideas. Past attempts to identify influencers in the academic world relied on influencer ranking techniques on the coauthorship network [126]. This suffers from two problems, the first being overlooking the field of a scientist. The second is that coauthorships are not a straightforward indicator of the researcher's impact, meaning that a researcher might have a large number of collaborators due to her academic environment or their field, not because of her popularity. In contrast we argue that neural networks can be used to fuse the structure and the fields using the articles' text representations. Moreover, we claim that citation networks are a more valid indicator as they are also used to measure scientific success in metrics like h-index [24].

1.5 THESIS GOAL AND CONTRIBUTIONS

This dissertation's main goal is to develop influence representations that can be used for large scale graph mining tasks. Inspired by the challenges mentioned above, we break the thesis in two parts the first part addresses the aforementioned problems of influence maximization, while the second targets apropos applications of influence in real world graphs. Our contributions can be summarized as follows:

1.5.1 *Contributions on Influence Maximization*

DIFFUSION REPRESENTATIONS FOR INFLUENCE MAXIMIZATION. *How to incorporate information from diffusion cascades for model-independent influence maximization?* To address problems (1.3.1) and (1.3.2), we develop a method that substitutes traditional influence estimation on random probabilities with learnt representations from diffusion cascades. Initially, we juxtapose the parallel literature of influence learning and influence maximization to highlight the aparrent gap of their combination. Subsequently, we propose a multi-task learning neural network that learns from a history of cascades the diffusion probability between nodes, irrespectively of their distance on the network, and the aptitude of a node to create large cascades. Using the diffusion probabilities and the influence representations, we define a submodular and monotone function that computes the influence spread of a user without simulations.We optimize this function using a variation of the

CELF [121] algorithm, to form a model-independent influence maximization method, called IMINFECTOR (Influence Maximization with INFluencer VECTORs). We evaluate the method using ground truth cascades instead of simulations on large scale datasets and compare with other similar methods.

LEARNING TO ESTIMATE INFLUENCE EFFICIENTLY. *Can we use machine learning to improve the time effiicency of influence maximization?* To address problem 1.3.3, we develop a GLIE, a graph neural network that learns to predict the influence estimations of a seed set on a graph weighted based on the weighted cascade [106]. The architecture is inspired by a theoretical upper bound of influence estimation and can substitute the costly independent cascade simulations. It can be used as a standalone influence predictor with competitive results for graphs up to 10 times larger than the train set and is considerably faster compared to a dynamic message passing model. The computational difference accumulates in the context of influence maximization where the number of influence estimations on each round is analogous to the size of the network. We thus leverage GLIE as a substitute for influence estimation in CELF. The proposed method can run in graphs with millions of edges in seconds, and exhibits comparative accuracy with a state-of-the-art algorithm. Despite its significant acceleration, the method has a significant computational overhead over the SOTA algorithm, we thus develop two new methods to address this. The first is a Q-network architecture, that utilizes GLIE's hidden representations and predictions to obtain seeds sequentially, while minimizing the number of influence estimations per round. The second is a submodular ranking function that adaptively diminishes a node's neighborhood as the seed set increases, based on the influence representations of GLIE. The results indicate that the latter method is faster than the state-of-the-art while providing accurate seed set quality.

1.5.2 *Contributions on Influence Applications*

PREDICTING THE EPIDEMIC SPREADING ON SMALL REGIONAL LEVEL. *Can we predict the number of COVID-19 cases in coarse geographical regions on the first wave of the pandemic?* We first create graphs using the Facebook data indicating daily mobility between geographical regions from applications on phones. We gathered and map open data regarding the progress of the disease in these areas using governmental and open data. We propose a model for learning the spreading of COVID-19 in a country's graph of such regions. The model relies on the representational power of GNNs and their capability to encode the underpinnings of the epidemic. Our main hypothesis is that if we capture how people move between regions along with the disease history, the prediction will improve. We apply a method based on Model Agnostic Meta Learning to transfer a disease spreading model from countries where the outbreak has been stabilized, to another country where the disease is at its early stages. We evaluate the proposed

approach on data obtained from regions of 4 different countries, namely France, Italy, Spain, and England. We observe that it can indeed surpass the benchmarks and produce useful predictions.

VISUALIZING ACADEMIC INFLUENCE AT SCALE. *Can we provide accurate depictions of a scientist's influence on the academic world?* We utilize the Microsoft Academic Graph, the largest open-source bibliographical information to build a coauthorship graph. We use the graph along with learnt representations from the paper abstracts to perform h-index prediction for each author using a variety of graph neural networks. Moreover, we create a citation graph between authors, which includes billions of edges calling for big data management to clean it and diminish it. We develop a parallel implementation of D-core decomposition [73] to compute and visualize the influence score of the authors in an interactive web applications. Furthermore, we provide distribution plots indicating the quantile that a scientist's h-index lies on, based on the distribution of h-indices on her most relevant fields.

1.6 OUTLINE

The rest of the dissertation is organized as follows. In Chapter 2, we briefly describe the required background on graph theory, submodular optimization and deep learning required to buttress the rest of the thesis. We also summarize different approaches to influencer identification such as graph degeneracy measures which will act as baselines for the proposed methods. Chapter 3 describes IMINFECTOR, a two-step method that learns influence representations from diffusion cascades and uses them to perform influence maximization on the social network. Chapter 4 presents GLIE, a neural network for fast influence estimation based on the independent cascade, alongside three methods, a traditional algorithm, a reinforcement learning model and a submodular ranking function, to perform influence maximization based on GLIE's estimations and representations. Chapter 5 includes a neural model that performs COVID-19 predictions on geographic NUTS3 regional level using the history of the epidemic in the country and mass mobility indices as logged from Facebook mobile applications. Chapter 6 presents the pipeline we utilized to perform influencer ranking on the Microsoft Academic Graph and a system we developed to visualize the results interactively in the web. Finally, we conclude the dissertation in 7 with an overview of the new questions derived from the thesis and future research.

2

BACKGROUND

In this chapter, we describe the preliminary concepts that our research is based on. Initially, we introduce multiple concepts from graph theory and underline their relevance with influence analysis. Furthermore, we provide an overview of influence maximization and submodular combinatorial optimization. Subsequently we introduce the basics of neural networks and deep learning, and describe their use in node and graph representation learning. Finally, we introduce transfer and reinforcement learning, which are utilizedi n certain parts ofthe thesis. The reader may refer to these references for a general introduction to graphs [33, 57, 156], to submodular optimization [109, 154], to neural networks [78, 87], to graph [86], transfer [162] and reinforcement [192] learning.

2.1 GRAPH BASICS

A graph is the abstract mathematical term used to explain a set of objects that are connected with each other in certain ways. Throughout the thesis, we use the terms network and graph interchangeably.

Definition 2.1.1 (Graph)**.** A graph $G(V, E)$ consists of a non-empty set of nodes V and a set of edges $E \subset V \times V$.

Alternatively, some parts of the literature refer to the nodes as vertices and the edges as links, but we will use the former definitions. The number of nodes in the graph is $n = |V|$ and the number of edges $m = |E|$. There are multiple types of graphs, but the ones of interest in this dissertation are *directed*,*weighted* or *bipartite*. The edges of the graph might indicate a directed relationship e.g. a social network user followes another user or a paper cites another paper or an undirected e.g. two authors wrote a paper together.

Definition 2.1.2 (Directed Graph)**.** A graph is undirected iff for every edge $(i, j) \in E$ that links the ordered pair of node i and node j, there is a reciprocal edge $(j, i) \in E$. A graph without this property is directed.

Moreover, the edges might be accompanied by a weight property e.g. how many papers have the aforementioned authors wrote together.

Definition 2.1.3 (Weighted Graph)**.** A graph is weighted if there is a function $W : E \to \mathbb{R}$ that assigns a real number to each edge.

The final type of graph that we use is the bipartite, where the nodes are separated into two groups and the edges connect only nodes from different groups e.g. the users in an e-shop and the products they purchage.

Definition 2.1.4 (Bipartite Graph)**.** A graph is bipartite if there is a property $L : V \to [0, 1]$ that partitions the node set into two disjoint sets $V_a, V_b, V = V_a \cup V_b$, such that every edge $e = (i, j) \in E \to (i \in V_a \wedge j \in V_b) \vee (i \in V_b \wedge j \in V_a)$.

Every graph can be represented by its *adjacency matrix* $\mathbf{A}$, which has size $n \times n$, and each row and column represent the nodes of the graph and each element has one if an edge exist or 0 if it does not. Expanding it to weighted graphs where the connections are represented by different volumes $w_{i,j}$, we have:

Definition 2.1.5 (Weighted Adjacency Matrix)**.** The weighted adjacency matrix $\mathbf{A}$ is a $n\dot{n}$ matrix such that:

$$\mathbf{A}_{i,j} = \begin{cases} w_{i,j}, \text{iff}(i,i) \in E, \forall i, j \in 1, ..., |V| \\ 0 \end{cases} . \tag{2.1}$$

The *degree* is the most prominent characteristic of a node inside a network. It corresponds to the number of edges a node has in an undirected graph. If the graph is directed, it breaks to indegree and outdegree i.e. the number of incoming edges and the number of outgoing edges. The *degree matrix* $\mathbf{D}$ is the diagonal $n \times n$ matrix where each element represents the degree of the corresponding node.

The final concept we will need is based on graph degeneracy [140] i.e. the hierarchical decomposition of a graph into its nested subgraphs until it can not be decomposed further. The graph is recursively peeled off based on a certain criteria, in the case of k-core decomposition, based on every node's degree surpassing a certain number k.

Definition 2.1.6 (k-core Decomposition)**.** A subgraph $H \subseteq G$ is a k-core of G iff $d_v \geq k, \forall v \in H$.

The main notion that separates k-core decomposition with a degree threshold filter is its recursiveness, meaning that the latter would remove all nodes with degree less then k and keep the resulting graph. In contrast, core decomposition continues removing new nodes that emerge with degree less than k, due to their neighbors being removed in the previous step. This process is repeated until no other node can be removed. An example can be seen in Figure (2.1). The k-core decomposition is rather important in the context of this thesis, as it is the primary metric to perform influencer identification. Influencer identification is a ranking procedure, where the nodes are sorted in decreasing function based on their potential on influencing the rest of the network. The k-core decomposition, due to extracting nodes that are part of dense cliques and lie in the center of the network, provides a solid substitute for influence based on the diffusion models used for influence estimation, which we analyze further below.

2.2 INFLUENCE MAXIMIZATION

Formally, let a graph G with influence probabilities p on the edges. A real world example of an influence graph is a follow network such

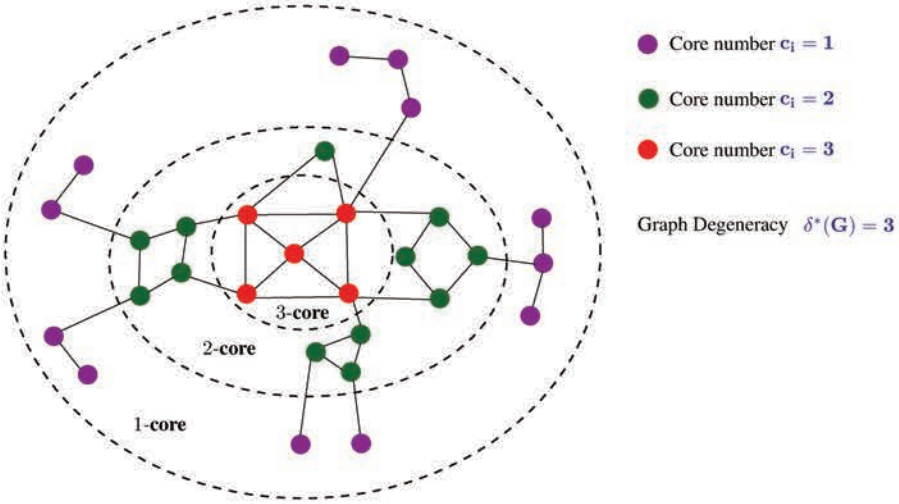


Figure 2.1: Example of k-core decomposition [136].

as Twitter [13] reversed, such that the user that follows is the user influenced. In case of undirected graphs such as Facebook, we typically split the edge in two reciprocal edges. Influence maximization is the problem of finding a set of nodes in the network such that if a diffusion starts from them, it would cover the maximum possible part of the network. Note that it in contrast to the the aforementioned influencer identification problem, in this case you aim to maximize the influence spread of a whole set, which is not necessarily the set of the individually most influential nodes, as their influence spreads may overlap. In essence it is a discrete optimization problem that has proven to be NP-hard [106] and it is solved through greedy submodular maximization. Embedded to the problem of influence maximization is influence estimation, which is a a #P-hard counting problem [212] and it is calculated based on an influence model. We analyze both problems separately.

2.2.1 *Influence Estimation*

Influence or diffusion models are stochastic computations utilized to estimate influence over a network. The Independent Cascade model, is the most well known influence model over a graph in computer science. The model runs in discrete time steps. The main assumption is that each node can infect each of its neighbors only at the next step after its own infection. This means that a node can be infected in numerous time steps by its neighbors, but once infected it stays this way, and it can only infect its neighbors once. A toy example can be seen in Figure 2.2 We can formulate it as a type of graph traversal algorithm as shown in Algorithm 1 :

Since the model is stochastic, one has to take into account the different worlds created by each probability included in the system to come up with the actual *influence spread*. Given that each edge is accompanied by an influence probability, then that means 2^e different combinations need to be computed. The reason we have to check all possible edges that are not directly associated with S, is that the existence of an edge

Algorithm 1 INDEPENDANT CASCADE

Input: Weighted Adjacency Matrix $\mathbf{A}$, Seed Set S
Output: Influence Spread σ

```
procedure IC(A, S)
    σ ← 0
    while |S| > 0 do
        σ ← σ + 1
        v ← S[0]
        for u = 0; u < |A|; u++ do
            p ~ U(0, 1)
            if A_{v,u} > p then:
                S.add(u)
        S ← S[1 :]
    return σ
```

Figure 2.2: Example of one step of the Independent Cascade simulation (red is infected).

may form paths of varying length, to nodes of varying degree. From a theoretical perspective, the problem is was proven to be #P-hard under the Independent Cascade as a reduction from counting s-t connectedness in a directed graph, which can not be performed in polynomial time [212].

In practice, a good approximation can be achieved by running 10,000 simulations of the Independent Cascade model starting from the seed set. The influence spread denoted as $\sigma(S)$ corresponds to the average number of nodes reached by the set. An analogous formulation relies on the live-edge model, a Monte Carlo sampling of edges that creates subgraphs based on the edge probabilities and calculates the final influence spread as the expected number of reachable nodes from the seed set over these graphs:

$$\sigma(S) = \sum_{g \subseteq G} P(g)\sigma_g(S) \tag{2.2}$$

$$P(g) = \prod_{e \in E(g)} p_e \prod_{e' \in E(G)/E(g)} (1 - p_{e'}) \tag{2.3}$$

The influence spread $\sigma_g(S)$ is the nodes that are connected with S by a path in g. An example can be seen in Figure 2.3.This does not alleviate the the required 10,000 samples to form an accurate estimate.

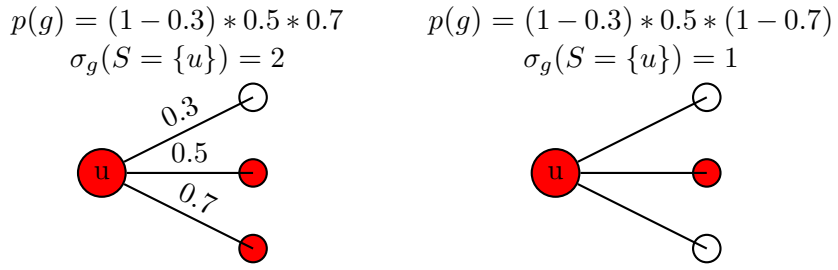


Figure 2.3: Influence estimation based on the live edge model under the Independent Cascade simulation (red is infected).

2.2.2 *Submodular Maximization*

As mentioned above, the aim of influence maximization is to find the seeds that exhibit the maximum combined influence spread, which is achieved by finding the most influential nodes while ensuring their influence overlap is minimized. We thus seek for a balance between how influential is a potential seed on its own and how much it overlaps with the rest of the seed set. As mentioned in Section 1.3, it is an NP-hard problem, because we need to take all possible combinations of nodes as seed sets and evaluate their influence spread in order to find the maximum. It is however possible to reach a guaranteed near-optimum solution using a greedy approach, based on the submodularity of the influence spread.

Let us first introduce the concept of submodularity. A function σ is submodular iff:

$$\sigma(S \cup u) - \sigma(S) \geq \sigma(S' \cup u) - \sigma(S'), \forall S \subseteq S' \tag{2.4}$$

We can can understand submodularity as a diminishing returns property. In our case, we see that as S increases, the marginal gain of a new node $u \notin S$ can only decrease or stay the same. Intuitively, as more nodes are added to S, the influence spread of u can either be left intact (if it does not overlap with S's) or can diminish if nodes in its influence spread are already influenced by the new additions on S. The influence estimation function based on the Independent Cascade is submodular as proven by a reduction from the set cover problem [106].

Algorithm 2 GREEDY

Input: Graph G, seed set size ℓ
Output: Seed set S

procedure GREEDY(G, ℓ)
  $S \leftarrow \emptyset$
  while $|S| \leq \ell$ **do**
    $u = \arg\max_{w \in V \setminus S} \sigma(S \cup \{w\}) - \sigma(S)$
    $S \leftarrow S \cup \{\}$
  return S

Monotonicity implies that adding new nodes to the seed set can only increase the spread, which is easy to deduce based on the Independent

Cascade which assumes influenced nodes stay influenced throughout the whole diffusion spread. Formally , $\forall u \in G$, we have:

$$\sigma(S' \cup u) \geq \sigma(S \cup u) \tag{2.5}$$

Submodularity and monotonicty allows us to use a greedy algorithm, which sequentially choose the node that provides the best marginal gain i.e. the maximum increase of the set's influence spread, such as the one presented in Algorithm (2). The algorithm is guaranteed a solution at $(1-1/e)OPT$ where OPT is the optimum spread [154].

Throughout the thesis, we propose methods that are based on a faster version of the GREEDY, called Cost Effective Lazy Forward (CELF) [121]. CELF is an acceleration to the original greedy that is based on the constraint that a seed's spread will never get bigger in subsequent steps. The algorithm can be seen in Algorithm (3). The influence spread is computed for every node in the first iteration and kept in a sorted list, q, as shown in Figure 2.4. We sort this list in descending fashion and choose the first (u). In the next iteration, we start computing the marginal gain from top to bottom. Let the marginal gain of node v for iteration $t+1$ be $\sigma_{t+1}(v)$. Then, we check if $\sigma_{t+1}(v) \geq \sigma_t(z)$, and if that is the case, since we have the definition of the influence spread $\sigma_t(z) \geq \sigma_{t+1}(z)$, we know that $\sigma_{t+1}(v) \geq \sigma_{t+1}(z)$. This property stems from submodularity, i.e. the marginal gain of any node can never increase with the size of the seed set. In other words, if the influence spread of a node examined at the current step is better than the next best node that was evaluated in any previous step , it is chosen because it is necessarily bigger than the rest of the sorted list. This process continues by calculating the marginal gain of the top node and resorting the list to compare it with the rest, until the top node remains at the top after computing its new marginal gain. The worst-case complexity is similar to greedy but in practice it can be hundreds of times faster, while retaining greedy's original guarantee of $(1-1/e)$.

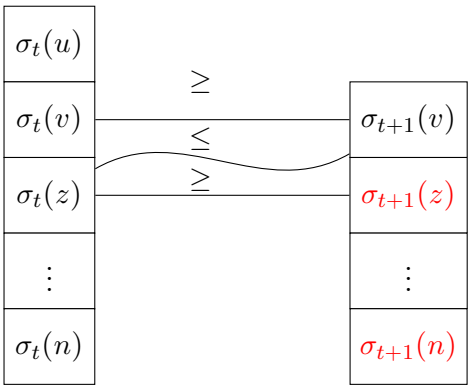


Figure 2.4: Visualization of the influence spread inequalities that allows CELF to reduce the number of influence estimations per step.

2.2.3 *Influence Sketches and Heuristics*

Several methods have been developed to accelerate the computations of influence maximization. The overwhelming majority focus on alleviating

Algorithm 3 CELF

Input: Graph G, seed set size ℓ
Output: Seed set S

```
procedure CELF(G, ℓ)
    set q ← [ ], S ← ∅, σ(S) ← 0
    for s ∈ V do
        R ← σ(s)
        q.append([s, R, 0])
    q ← sort(q, 1)
    while |S| < ℓ do
        u ← q[0]
        if u[2] == |S| then
            S.add(u[0])
            σ(S) ← σ(S) + u[1]
            q.delete(u)
        else
            R ← σ(S ∪ {u[0]}) − σ(S)
            q[0] ← [u[0], R, |S|]
            q ← sort(q, 1)
    return S
```

the computational demand of repeating simulations for influence estimation. We will briefly mention them as they form the set of baselines we utilize to quantify our proposed methods.

The most notable acceleration is achieved using reverse reachable sets as a means of sketching to estimate influence without simulations in every step. The main idea is to approximate the influence of a seed based on the sets of nodes that reach randomly sampled nodes [22]. Intuitively, if a node appears in many such sets, it can reach many random nodes. it was proven that under a high probability that depends on the number of reverse reachable sets sampled, the amount of sets that a node is found in is analogous to its actual influence [22]. This results in remarkable acceleration as not only the number of simulations required to form the initial reverse reachable sets is practically smaller than computing the influence spread with 10,000 Monte-Carlo Independent Cascade simulations for every node, but also the same reverse reachable sets can be used for influence estimation throughout all steps [197]. Several improved version have been developed, relying on faster reachability estimations [45], building reverse reachable sets that depend on each other and hence diminishing their required number [196], and testing the quality of the solution to stop the creation of reverse reachable sets early [157].

Moreover, numerous heuristics have been proposed that, though lacking guarantees, exhibit remarkable success in practice. These methods rely mostly on empirical and intuitive shortcuts. For the Independent Cascade model, the degree discount [38]. PMIA relies on the influence paths with the highest probability to compute the influence of a node [37] and IRIE relies on PMIA and computes the influence of one node in constant time to accelerate the procedure [101]. On the other hand,

for the linear threshold model, SimPath is estimating influence up to a certain number of hops [81] and Ldag relies on local directed acyclic decompositions of the graph to approximate the influence estimation [39].

As the problem of influence maximization became more popular, several methods were developed to construct more realistic versions by adding temporal [129], topic [127] and location constraints [237], or addressing the adversarial settings [98]. From the perspective of network immunization, the required seed set can be retrieved by finding the must pivotal nodes using optimal percolation [170], which is proven to be equivalent to IM under the linear threshold model [151]. Such methods are developed for epidemic containment and have not proven effective in typical influence maximization evaluation yet, hence are omitted in this paper. In order to better dissect the vast literature on influence maximization, we created a github repository[1] devoted to every relevant paper we could find, classified in different categories.

2.3 GRAPH LEARNING

Graph learning covers a broad spectrum of methods that aim to address traditional machine learning tasks on graph data. It is also connected to transforming the nodes in continuous representations of low dimensionality or high sparsity. Such transformations allow to capitalize on methods from the broad literature of statistics and optimization for continuous spaces, on the input graph, a fundamentally discrete methematical object. The main challenge is to retain the relationships between the nodes, meaning the edges, in the space of the new representations. To think of some of the advantages of such representations, one can simply address the problem of comparing two graphs with comparing two vectors. The latter can be computed in linear time by the Euclidean distance, while the former is refered to the graph isomoprhism problem [146], and there is no known polynomial time algorithm to solve it. Hence the need to form continuous representations of graphs, a practice that was already common before the advent of deep learning, either by topology [83] or by kernel mechanisms [159].

Recently, various types of graph learning has resurfaced as a promising subfield of deep learning [107, 171]. The primary goal to learn representations from graphs is to perform these machine learning tasks:

- Node classification/regression: given a graph where a number of nodes are labeled, treat each node as a sample and classify or regress the labels of the rest of the nodes.
- Edge prediction: given a pair of vertices, predict whether there is or should be an edge between them.
- Semi-supervised learning: When the vast majority of the samples on a typical machine learning dataset are unlabeled, the model capitalizes on an underlying graph that forms relationships be-

1 https://github.com/geopanag/awesome-influence-maximization-papers

tween the labeled samples and the rest of the dataset to form accurate predictions.

- Graph classification/regression: given a set of graphs with known labels, classsify or regress the labels for a set of unseen graphs.
- Graph clustering: given a graph, separate the nodes into well-balanced groups, also called communities, such that the within group edge density is large and the between group edge density is small.

Some applications of these tasks include classifying molecules [74, 158], reasoning over knowledge graphs [40], understanding documents [160], recommendations in social networks [65] or solving combinatorial optimization problems [47]. The learning methods can be broken down to two subfields, node representation learning and graph neural networks, both of them comprising of deep neural networks at their core.

2.3.1 *Neural Networks*

The simplest neural network, a multilinear perceptron, is a dot product between the input samples X and a set of parameters W, followed by a sigmoid function such as *tanh* to form a prediction probability for each of the input samples [78]. Deep learning is a sequence of numerous such dot products with W where the output of one serves as input to the other and the activation function between them is commonly a non-linear function f such as *ReLU* [115]. Each pair of parameters and activation function is called a neural layer $f(XW)$. Neural networks have proven to be universal approximators, meaning that theoretically they can represent any type of target function, given enough data and training [93]. The parameters are trained using backpropagation i.e. the derivative of the loss function is computed and the the chain rule dictates the updates of each layer [178]. In practice, the deep network is trained using batches of the data and stochastic gradient descent is utilized to update the parameter with backpropagation, because of its computational efficiency and learning benefits [26].

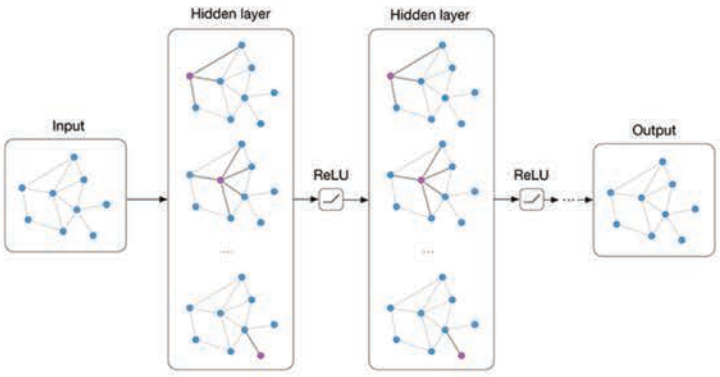


Figure 2.5: Schematic representation of a graph convolutional neural network from https://tkipf.github.io/graph-convolutional-networks/.

2.3.2 *Node Representation Learning*

Inspired by the huge success of word2vec [75], DeepWalk[171] and then node2vec [84] were developed and have proven equally effective in the domain of graphs. Their neural architecture comprises of two layers, one hidden and one output. Each node is represented typically by a one-hot vector. To train the model we start with random walks from a random node, which produces a sequence of nodes. A predetermined window w passes through this sequence and creates pairs of the window's median node u each other node v in the window. These are called node-context pairs and are used to train in an unsupervised manner the neural network. The input to the model is u and the predicted output is v. The model, resembling Skipgram [75] can be summarized as follows:

$$\arg\max_{\mathbf{W}}(\log(pr(v|u))) = \frac{\exp(\mathbf{W}_u^1 \mathbf{W}_v^2)}{\sum_{v' \in V} exp(\mathbf{W}_u^1 \mathbf{W}_{v'}^2)} \tag{2.6}$$

where W^1 is the hidden layer parameters and W^2 is the output. Due to the sum in the denominator of the softmax function being commonly prohibitively big, most architectures utilize negative sampling, i.e. taking a few nodes that do not appear in the neighborhood of u to normalize the probability. Intuitively, the model is trained to bring closer the hidden layer representations of nodes with similar neighborhoods. This allows to translate the relationships between the nodes from the discrete space into the continuous $\mathbb{R}^d$ where d is the size of the embedding.

2.3.3 *Graph Neural Networks*

A graph neural network is essentially a non-linear parameterization of message passing. To be specific, given that the input samples are connected in a graph G and let let $\mathbf{H}_v^{(0)} \in \mathbb{R}^d$ denote the initial feature vector of node v, the aim is to combine the features $\mathbf{H}_u^{(0)}, u \in N(v)$ of a node v's neighbors along with H_v such that the accuracy in the downstream task is enhanced. This combination which resembles message passing, is typically a summation or an average $\mathbf{Z}_u = \mathbf{H}_u^0 + \sum_{v \in N(u) H_v^0}$ and is the input to a simple multilinear perceptron $\mathbf{H}^1 = f(\mathbf{Z}\mathbf{W}_0)$. Each of the GNN layers uses the graph structure and the node feature vectors from the previous layer to generate new representations for the nodes. The feature vectors are updated by aggregating local neighborhood information. Suppose we have a GNN model that contains T neighborhood aggregation layers. In an abstract sence, for the t-th neighborhood aggregation layer ($t > 0$), the hidden state $\mathbf{H}_v^{(t)}$ of a node v is updated as follows:

$$\begin{aligned} \mathbf{M}_v^{(t)} &= \text{AGGREGATE}^{(t)}\left(\left\{\mathbf{H}_u^{(t-1)} | u \in \mathcal{N}(v)\right\}\right) \\ \mathbf{H}_v^{(t)} &= \text{COMBINE}^{(t)}\left(\mathbf{H}_v^{(t-1)}, \mathbf{M}_v^{(t)}\right) \end{aligned} \tag{2.7}$$

By defining different $\text{AGGREGATE}^{(t)}$ and $\text{COMBINE}^{(t)}$ functions, we obtain a different GNN variant. For the GNN to be end-to-end trainable, both functions need to be differentiable. Furthermore, since there

is no natural ordering of the neighbors of a node, the $\text{AGGREGATE}^{(t)}$ function must be permutation invariant. The node feature vectors $\mathbf{H}_v^{(T)}$ of the final neighborhood aggregation layer are usually passed on to a fully-connected neural network to produce the output. The model with an average aggregate mechanism, called Graph Convolutional Network (GCN)[107] can be seen in Figure 2.5 and can be expressed in a vectorized manner as :

$$\mathbf{H}^{i+1} = f(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{H}^i\,\mathbf{W}^{i+1}) \tag{2.8}$$

where $\mathbf{D}$ is the degree matrix and the product $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ resembles the normalized adjacency matrix. Normalization is mainly applied to avoid numerical instabilities and exploding/vanishing gradients associated with deep neural networks. One should note that by normalizing, we capture the distribution of the representations of the neighbors of a node, but not the exact multiset of representations. After the desired depth l is reached, the final representations can be used as an input to a node classifier or as concatenated pairs for edge prediction. An example scheme of a graph convolutional network can be seen in Figure (2.5).

$$\hat{y} = \text{ReLU}(\mathbf{H}^l\mathbf{W}_o)) \tag{2.9}$$

The aggregation of all node representations can be used for graph-level tasks such as graph classification or graph similarity.

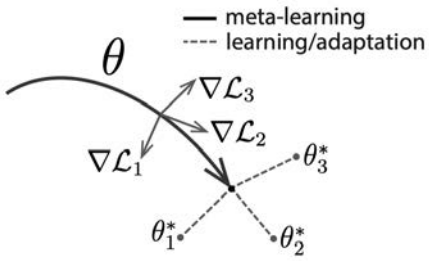


Figure 2.6: Scheme of Model Agnostic Meta Learning (MAML) [68].

2.4 TRANSFER AND MULTI-TASK LEARNING

Transfer learning contains numerous promising methods to address data scarcity. The core concept is to train a model on a specific task where there is enough labeled and unlabeled data, and then utilize this model in a task that suffers from insufficient amount of training or overall data samples. It has gathered increased attention for deep learning models for two reasons. The first is that it is easy to utilize a pretrained set of parameters by simply concetanating the current model with them. The second is deep learning's relentless need for large amount of training data, which makes it unuseable in tasks with insufficient data, thus transfer learning is a promising path to address this. The method is closely related to multi-task learning [61], where the aim is to learn a set of parameters that perform multiple tasks at the same time, to enhance the models capacity to generalizing. Both are

considered part of the general subfield of meta-learning [206] and both have exhibited tremendous success recently. One example is the GPT-3 natural language model that learns to generalize through different languages efficiently [29].

In the context of this thesis, we develop a multi-task neural network [31] i.e. a neural network with two output layers trained on two types of labels. Moreover, we utilized the Model Agnostic Meta Learning (MAML) [68] as a means to improve forecasting in the start of a temporal dataset, similar to addressing a cold start problem. In MAML, we use a set of meta-train datasets (train and test sets) $M_tr = \{D^1, D^2, ..., D^p\}$ to learn an initial set of parameters θ. These are trained regularly on a meta-test training set and tested on the meta-test testing set regularly. MAML takes advantage of the meta-train set by taking small gradient steps over θ instead of regular training. This is achieved by training $\theta*$, a temporary version of θ on some samples from a specific task's train set. Subsequently, the error of the prediction of $\theta*$ with the task's test set creates a gradient that is used to update θ. The purpose is to take small regulated steps towards a better estimate without falling in early local minima, as shown in Figure (2.6). After the metatraining, θ underoges regular training in the meta-tests train set and evaluated in the respective test set.

2.5 REINFORCEMENT LEARNING

In reinforcement learning, the task is to learn a model that can provide sequential decisions. The system can be formalized as a Markov Decision Process (MDP) which consists of a set of states S, a set of actions A, a reward R that is a function of a state-action pair, and a transition probability P that defines the distribution to transition on the next state [192]. The Markov assumption certifies that the probability of the next state depends only on the immediate previous: $P(t+1|(s_t, a_t), ..., (s_0, a_0) = P(t+1|(s_t, a_t))$. The model we learn is the policy $\pi(s_t)$ that maps a state to an action, and is a neural network in the case of Deep Q-learning [205].

For the model to learn, we create a dataset based on simulating different trajectories of the environment. At each time step and until the environment has reached its final state, we typically choose an action based on the policy or based on a random choice. We balance between these two choices using a random threshold e. This allows the training set to include samples from untouched state-action pairs (exploring) and samples predicted by the policy to increase the final reward (exploiting). Each choice yields an action and hence a new state and a reward is retrieved, which are stored in a memory buffer. The outcome of one simulation is called a trajectory $((s_0, s_1, a_0, r_0), (s_1, s_2, a_1, r_1), ...)$ and is stored as separate quadraples in the memory.

In the thesis, we assume that the system does not have transition probabilities i.e. if we choose an action the agent arrives at it in the next step deterministically, we aim thus for a non-stochastic policy. The policy can be learned directy using policy gradient [186] or by learning

the Q-value of a given state-action pair. We will focus on the latter, since we aim for a deterministic policy and we have a discrete action space. To learn the mapping we use a neural network Q^{π} that takes as input the state-action embeddings and outputs a score for each possible action. The action is then chosen as:

$$a' = \arg\max_{a} Q^{\pi}(s, a) \tag{2.10}$$

The Q function corresponds to the cumulative reward up to K steps ahead:

$$Q^{\pi}(s, a) = \mathop{\mathbb{E}}_{\pi}[R_t] = \mathop{\mathbb{E}}_{\pi}[\sum_{k \geq 0}^{K} \gamma^k r_{t+k} | s_t = s, a_t = a] \tag{2.11}$$

where γ represents a discount factor for the future rewards. The reason the Q function does not act solely on myopic rewards ($K = 0$) and instead uses the cumulative reward, is that reinforcement learning aims to perform approximate dynamic programming. It is thus trained to choose the action that will improve the reward in the long term, and not to act greedily. If we assume s' to be the next state from (s, a), we can derive the Bellman equation for the optimum policy π^*, $Q^*(s, a) = max_{\pi} Q^{\pi}(s, a)$ using Equation 2.11:

$$Q^*(s, a) = \mathop{\mathbb{E}}_{\pi}^{*}[r_t + \gamma \sum_{k \geq 0}^{K} \gamma^k r_{t+k+1} | s_{t+1} = s'] = \tag{2.12}$$

$$= \mathop{\mathbb{E}}_{\pi}^{*}[r_t + \gamma \max_{a}' Q^*(s', a')] \tag{2.13}$$

$$= \sum_{s \in S} [R(s, a, s') + \gamma \max_{a}' Q^*(s', a')] \tag{2.14}$$

which clarifies the recurrent nature of the model. We thus aim to train Q^{π} such that it captures the future rewards given by the environment and the value of the future state. The value of the future state is typically computed by the maximum action of $Q^{\pi'}$, a target network updated with the value of Q^{π} after every certain amount of steps, to stabilize training:

$$\Lambda = \mathbb{E}[(Q^{\pi}(s_t, a_t) - (r_{t+n} + \gamma \max_{a} Q^{\pi'}(s_{t+n}, a)))^2] \tag{2.15}$$

where $n > 1$ to capture long term rewards. During training in practice we sample a certain amount of transition quadraples from the memory and use them as samples to evaluate the loss function.

3

MULTI-TASK LEARNING FOR INFLUENCE ESTIMATION AND MAXIMIZATION

We address the problem of influence maximization when the social network is accompanied by diffusion cascades. In prior works, such information is used to compute influence probabilities, which is utilized by stochastic diffusion models in influence maximization. Motivated by the recent criticism on the effectiveness of diffusion models as well as the galloping advancements in influence learning, we propose IMINFECTOR (Influence Maximization with INFluencer vECTORs), a unified approach that uses representations learned from diffusion cascades to perform model-independent influence maximization that scales in real-world datasets. The first part of our methodology is a multi-task neural network that learns embeddings of nodes that initiate cascades (influencer vectors) and embeddings of nodes that participate in them (susceptible vectors). The norm of an influencer vector captures the ability of the node to create lengthy cascades and is used to estimate the expected influence spread and reduce the number of candidate seeds. In addition, the combination of influencer and susceptible vectors form the diffusion probabilities between nodes. These are used to reformulate the network as a bipartite graph and propose a greedy solution to influence maximization that retains the theoretical guarantees. We apply our method in three sizable networks with diffusion cascades and evaluate it using cascades from future time steps. IMINFECTOR outperforms various competitive algorithms and metrics from the diverse landscape of influence maximization in terms of combined efficiency, scalability and seed set quality.

SOURCE CODE The implementation of the proposed model can be found online [1].

3.1 INTRODUCTION

In section 1.3, we briefly described the problems of utilizing diffusion models with random probabilities in the context of influence maximization. We underline the importance of learning these probabilities from historical logs of activity i.e. diffusion cascades. A diffusion cascade is a sequence of events in discrete time. The events correspond to users in the network, who are influenced by the topic of the cascade in time. A tweet and its retweets is the most obvious example, but it can generalize to other systems, including the web and the academic literature, as we will see further in the experimental section. One crucial subtle point here is that we are not aware of who influences whom, which would

1 https://github.com/geopanag/IMINFECTOR

make the diffusion cascade a tree, we are only aware of the sequence the users are influenced at.

The benefits of learning influence probability from real diffusion cascades over random assignments are obvious. However, even when the parameters are learned in an empirical manner from diffusion cascades, in multiple cases the independent cascade (IC) model is utilized [27, 54, 179], which is problematic for two reasons. Apart from severe overfitting due to the massive number of parameters, this approach assumes influence independence throughout related nodes or edges of the same node. This assumption overlooks the network's assortativity, meaning that an influential node is more prone to effect a susceptible node then a less influential node, even when the edge of the latter to the susceptible is stronger than the edge of the former. This effect comes in contrast with the actual mechanics of influence [12].

To address these issues, we propose a method that learns influencer and susceptible embeddings from cascades, and uses them to perform influence maximization without the use of a diffusion model. Initially we propose CELFIE, a variation of the CELF algorithm analyzed in Chapter 2, that utilizes influence probabilities. Our model capitalizes on the information contained in past diffusion cascades using influence and susceptibility representations learnt for this setting. To be specific, we device a modification of the recently proposed influence embedding model INF2VEC [66] in order to improve its scalability. Our focus on the cascades' initiators is justified from empirical evidence on the predominantly tendency of influencers to initiate rather than copy cascades, which we argue based on a data analysis on *Sina Weibo*. As we will see in detail in the next section, INF2VEC computes influence and susceptibility embeddings based on time precedence in diffusion cascade history and follow relationships in the network. We adjust it to focus on initiators of the cascades and compute diffusion probabilities, which are influence probabilities between nodes with any possible distance in the network. Following suit from recent model-independent influence maximization algorithms [113, 207], we overlook the diffusion model and connect each candidate seed (influencer) and every susceptible node with a diffusion probability using the dot product of their respective influencer-susceptibility embeddings, forming a bipartite network. Diffusion probabilities have the advantage of capturing higher-order correlations that diffusion models fail to due to their Markovian nature. Moreover, on this setting there are no higher order paths, which means that each seed directly infects a specific node, thus the stochastic existence of an influence edge effects only the infection of the node in the receiving end, and the need for repeated simulations is eliminated. We take advantage of this to define a new marginal gain based on a sampling strategy similar to the live-edge model, but diminishing the sampling space based on previously influenced nodes and reusing a seed's past samples to compute the new influence spread.

This model however suffers from an overestimation of the influence spread i.e. the network is covered very fast which is computed by sampling repetitively from a node's neighbors and summing the edge's weights, decreases very fast during the seeding rounds. This has as a

result the whole network being covered within a few iterations, meaning that the retrieved seed set can stop at a budget smaller than the one provided, if the method has detected that the whole network is already covered. To address this, we first modify further the influence learning model into INFECTOR (INfluencer Vectors), a multi-task neural network that learns influencer vectors for nodes that initiate cascades and susceptible vectors for those that participate in them. INFECTOR additionally embeds the aptitude of an influencer to create sizable cascades in the norm of her embedding. We use these embeddings and their norms to define an influence spread over this new bipartite network, that uses the estimated aptitude as a prediction for the percentage of the network each seed can cover. Moreover, we can use this estimate of every candidate influencers' spread to diminish the number of the candidate seeds. Finally, we propose IMINFECTOR, a scalable greedy algorithm that uses this submodular influence spread to compute a seed set, retaining the theoretical guarantee of $1 - 1/e$. To evaluate the performance of the algorithm, the quality of the produced seed set is determined by a set of unseen cascades from future time steps, similar to a train and test split in machine learning. We deem this evaluation strategy more reliable than traditional evaluations based on simulations because it relies on actual traces of influence. IMINFECTOR outperforms CELFIE and previous influence maximization methods with learned edge weights, either in quality or speed in three sizable networks with cascades.

3.2 RELATED WORK

The main problem of the aforementioned influence maximization approaches is that the established influence spread may lead to seed sets of poor quality. This can be caused either by the assignment of simplistic influence weights or by the diffusion model's innate assumptions. To address the issue with the random influence weights, some novel influence-maximization approaches assume multiple rounds of influence maximization can take place over the graph, hence multiple simulations can be used to compute the influence probabilities while balancing between the number of nodes influenced in each round and learning influence for non examined parts of the network. Since this is an inherently exploration-exploitation problem these models are based on multi-armed bandits to use the feedback and update of the parameters [191, 219, 224]. Though useful, these algorithms are built based on the diffusion models as well, hence they share their aforementioned deficiencies. Recently more model-independent online learning [113] approaches that utilize regret functions without the use of diffusion models have been proposed [207], but they suffer from scalability issues and would not be able to scale in real-world social networks, such as the ones we examine in this work.

There have been few attempts to address influence maximization with learned influence parameters from real past cascades, which serve as benchmark comparisons for our proposed approach [79, 80]. Influence

learning can also follow a more principled probabilistic perspective, by learning the parameters based on the independent cascade model [77]. These models suffer from overfitting due the number of parameters which is proportional to the number of edges. To address this an array of works utilize representation learning to capture influence between two nodes. A representative example is EMBEDD-IC [27], an adaptation of the independent cascade where the influence probabilities are expressed as a combination of the nodes' embeddings. Each node is associated with two embeddings, a source and a target, which combined construct the probability of influencing and getting influenced. This method however relies on independent cascade, and consequently overlooks the aforementioned higher order influence (Section 1.3). Another important problem with influence learning techniques that are based on diffusion models, is that they assume influence independence between edges. This assumption does not allow the model to capture influence similarities between nodes of similar status, which exist because of the assortative mixing property [155]. Moreover, it causes a substantial depreciation in the estimation of the influence spread [9], which is why in this work, we utilize a model that learns influence and susceptibility embeddings based on the co-occurence in diffusion cascades [66]. More recent influence learning methods are devoid of diffusion models and learn the embeddings based on co-occurrence in cascades [66], similarly to node representation learning. This type of influence learning has not been utilized yet for influence maximization. Although more accurate in diffusion prediction, the input of this model consists of node-context pairs derived from the propagation network of the cascade, a realization of the underlying network (e.g. follow edges) based on time-precedence in the observed cascade (e.g. retweets). This requires looping through each node in the cascade and iterating over the subsequent nodes to search for a directed edge in the network, which has a complexity of $\mathcal{O}(c\bar{n}(\bar{n}-1)/2)$, where c is the number of cascades and $\bar{n}$ is the average cascade size. This search is too time-consuming as we analyze more on the experimental section.

It should be noted that apart from pure influence learning, multiple methods have been developed to predict several aspects of a cascade, such as TOPO-LSTM and HIDAN to predict the next infected node [214, 216], RMTPP, DEEPDIFFUSE and CYANRNN for next node and time of infection [52, 97, 215], DEEPCAS to predict cascade size [123], and FOREST to predict next node and size together [231]. These methods have advanced end-to-end neural architectures focused each on their respective tasks. However, their hidden representations can not be utilized in the context of IM, because they are derived by a sequence of nodes, hence we can not form individual influence relationships between two distinct nodes. In other words, an influence maximization algorithm requires a node-to-node influence relationship, which is not clearly provided by the aforementioned references. Moreover, methods that do learn node-to-node influence but capitalize on other information such as sentiment of the context [130] can not be utilized as well due to our datasets missing content, but it is a promising future approach.

Symbol	Meaning	Type
N	number of nodes	scalar
t_u	time of node' u post	scalar
C_u	cascade initiated by u	set
$\mathbf{x}, \mathbf{y}_t$	one-hot embedding of a node	vector
y_c	cascade length	scalar
X^t	$\mathbf{x}, \mathbf{y}$ pairs extracted from cascade	list
$\mathbf{O}$	origin embeddings of INFECTOR	matrix
$\mathbf{T}$	target embeddings of INFECTOR	matrix
C	constant embeddings of INFECTOR	vector
$p_{u,v}$	probability of u influencing v	scalar
$\mathbf{z}$	hidden layer output	vector
f_t	softmax	function
f_c	sigmoid	function
$\boldsymbol{\varphi}_t$	output for node influenced	vector
φ_c	output for cascade size	scalar
$\boldsymbol{\Phi}$	jacobian of INFECTOR's output	matrix
L	loss of INFECTOR	vector
D	INFECTOR classification probabilities	matrix
λ_u	number of nodes to be influenced by u	scalar
$\hat{D}_s$	D sorted for the row of node s	matrix
$\sigma'(s)$	influence spread of node set s	scalar
S	Set of seed nodes	set

Table 3.1: Table of symbols.

3.3 LEARNING INFLUENCER VECTORS

3.3.1 *Node-Context Extraction*

Our goal in the first part of our methodology is to create a model that learns representations suitable for scalable model-independent influence maximization. Due to the lack of a diffusion model, the network's structure becomes secondary and online methods tend to rely only on the activity of individual nodes [113, 207]. We follow suit and propose an influence learning method that focuses on influencers. We start from the context creation process that produces the input to the network. In

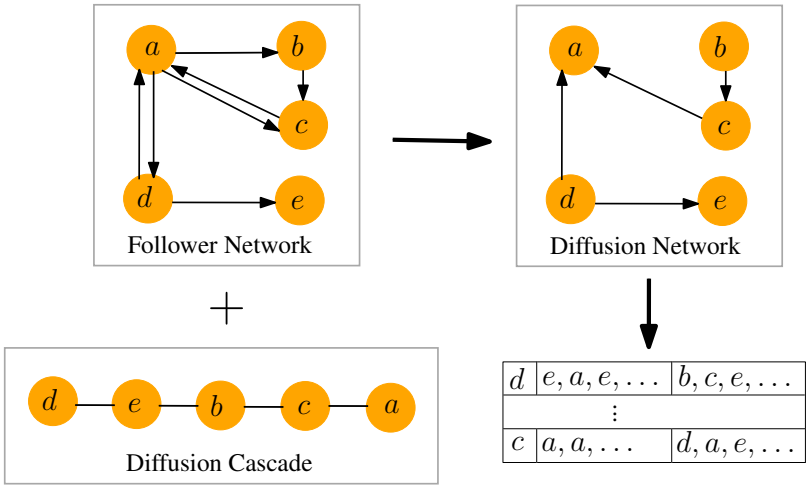


Figure 3.1: Schematic representation of INF2VEC node-context creation [66].

previous node-to-node influence learning models, initiating a cascade is considered equally important with participating in a cascade created by someone else [66], thus the context of a node is derived by the nodes occurring after it in a cascade, as we see in Figure 3.1.

This process requires the creation of the propagation network, meaning going through every node in the cascade and iterating over the subsequent nodes to search for a directed edge in the network. This has a complexity of $\mathcal{O}\Big(c(\bar{n}(\bar{n}-1)/2)\Big)$, where c is the number of cascades and $\bar{n}$ is the average cascade size. Given that the average size of a cascade can surpass 60 nodes, it is a very time consuming for a scalable influence maximization algorithm. To overcome this we focus on the nature of IM, meaning the final chosen seed users will have to exert influence over other nodes, thus we may accelerate the process by capitalizing on the differences between influencers and simple users. Intuitively, we expect the influencers to exhibit different characteristics in sharing content than the simple/susceptible nodes [155]. We argue that an influencer's strength lies on the cascades she initiates, and evaluate this hypothesis through an exploratory analysis.

We utilize the cascades of *Sina Weibo* dataset, a large scale social network accompanied by retweet cascades to validate our hypothesis. The dataset is split in train and test cascades based on their time of occurrence and each cascade represents a tweet and its set of retweets. We keep the 18,652 diffusion cascades from the last month of recording as a test set and the 97,034 from the previous 11 months as a train set. We rank all users that initiated a cascade in the test set based on three measures of success: the number of test cascades they spawn, their cumulative size, and the number of Distinct Nodes Influenced (DNI) [54, 164, 169], which is the set of nodes that participated in these test cascades. We bin the users into three categories based on their success in each metric, and for each category we compute the total cascades the users start in the training set opposed to those they participate in.

Here, we examine the contrast between the behavior of the influencers and normal users, meaning how more probable is for an influencer to start cascades compared to normal users. As we see in Figure 3.2,

users that belong to the top category of the test set are much more prone to create cascades compared to the ones who belong to the mid and low categories. Since our end goal is to find influencers for our algorithm, this observation allows us to focus solely on the initiators of the cascades, rather than every node in the cascade. Moreover, we observe that influencers in Weibo are more prone to create cascades opposed to participating in them. This means that by overlooking their appearances inside the cascades of others, we do not lose too much information regarding their influence relationships, as most of them start the cascades.

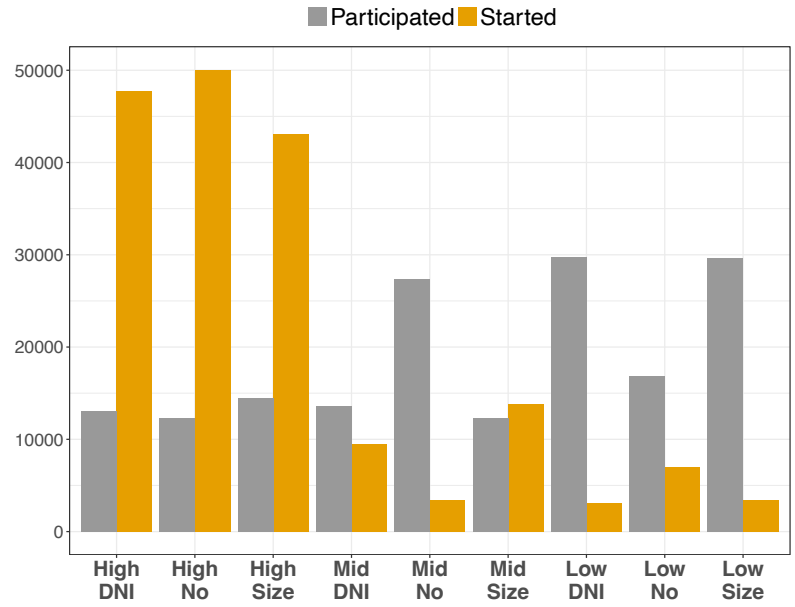


Figure 3.2: Influencer's initiating versus participating in diffusion cascades. **DNI** stands for distinct nodes influenced, **size** stands for cascade size, and **no** stands for number of cascades started.

To this end, we propose a new approach to the creation of the input to the neural network behind INF2VEC. Instead of deriving one node-context pair for each node in the diffusion, we can derive a single one only for the initiator of the cascade. Apart from the significant computational gain, we expect this approach to perform adequately due to the above observation. Moreover, instead of creating the diffusion network, which is time consuming and prone to noise due to the absence of edges [164], we can perform another type of sampling to derive the context of the initiator.

Another important characteristic in the study of social influence is its temporal dynamics. In the case of information diffusions, the time passed between two node's activity is known to play a role in the amount of influence the source node exerts to the target [79]. Yet, the original INF2VEC algorithm does not capitalize over this attribute. In our dataset, we can observe this phenomenon by studying the copying times in the diffusion cascades of the influencers. As mentioned above, we focus on the nodes that have started a cascade in the test set and use as a measure of a node's influence the number of nodes it infects

throughout all these cascades (DNI). We compute the average copying time of a cascade, for all train cascades of these nodes, and average it to get an estimate of how fast these nodes get "copied" during their cascades. Subsequently, we group these nodes based on their DNI and compute the average copying time of each group and plot it opposed to the DNI in Figure 3.3. This plot indicates that nodes with higher influence tend to have much faster cascades than ordinary influencers. Of course, there are much more nodes with smaller DNI than high, but since we take the average of the copying time for each DNI, the result is not affected. Apart from confirming our intuition, this observation buttress several findings that underline the decay of influence as the copying time increases in real-world diffusion cascades [79]. To this end, we guide the sampling of a node's context based on the copying times in its cascades. More specifically, we form the context by performing a random sampling over the nodes in the cascade, where the probability is inversely proportional to the copying time between the candidate node and the initiator. For the current analysis, we do an oversampling of 120% to emphasize the importance of fast resharing in the depiction of influence. This creates node-context pairs that take into account the temporal dynamics of the cascades.

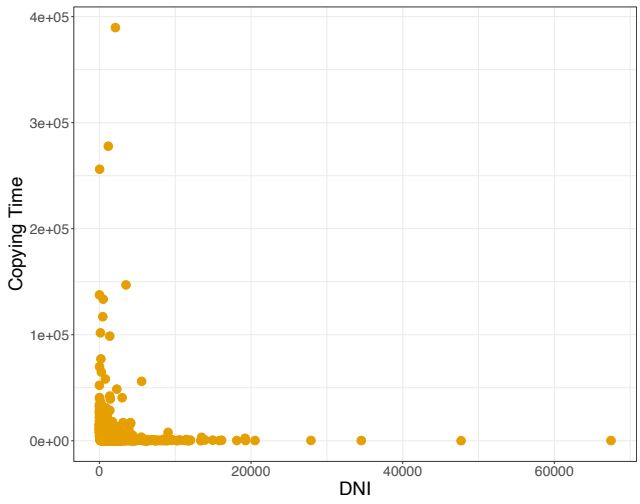


Figure 3.3: Average copying time in the train cascades relative to the number of distinct nodes influenced in the test cascades.

To clarify notation, all cascade initiators are called influencers, an influencer u's context will be created by sampling over all nodes v in a given cascade c that u started, with probability inversely proportional to their copying time. In this way, the faster v's retweet is, the more probable it will appear in the context of u, following this formula

$$P(v|\mathcal{C}_u) \sim \frac{(t_u - t_v)^{-1}}{\sum_{v' \in \mathcal{C}_u} (t_u - t'_v)^{-1}} \tag{3.1}$$

It is a general principle to utilize temporal dynamics in information propagation when the network structure is not used [54, 77, 238] and is based on empirical observation on the effect of influence in time [79]. In our case we do an oversampling of 120% to emphasize the importance of fast copying. This node-context creation has a complexity of $\mathcal{O}(cn)$,

which is linear to the cascade's size and does not require searching in the underlying network. Even more importantly, this type of context sets up the model to compute diffusion probabilities, i.e. influence between nodes with more than one hop distance in the network.

3.3.2 *Diffusion Probabilities*

Intuitively, diffusion probability (DP) is the probability of the susceptible node appearing in a diffusion started by the influencer, independently of the two nodes' distance in the network. This means that the underlying influence paths from the seed to the infected node is included *implicitly*, which changes drastically the computation of influence spread. For example, in a setting with diffusion models and influence probabilities, a node u might be able to influence another node z by influencing node v between them, as shown in Figure 3.4. However, in our case, if u could indeed induce the infection of z in a direct or indirect manner, it would be depicted by the diffusion probability $p_{u,z}$.

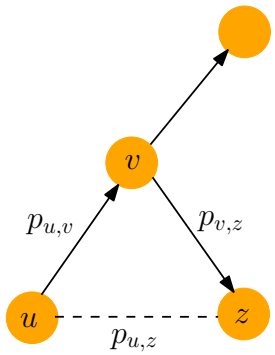


Figure 3.4: Example of higher order influence showing the difference between influence and diffusion probabilities.

In this setting we can capture the case when v appears in the cascades of u and z appears in the cascades of v but not in u's. This is a realistic scenario that occurs when v reshares various types of content, in which case content from u might diffuse in different directions than z. Hence, it would be wrong to assume that u's infection would be able to eventually cause z's. Typical influence maximization algorithms fail to capture such higher order correlations because diffusion models act in a Markovian manner and can spread the infection from u to z.

3.3.3 *INFECTOR*

The diffusion probabilities (DPs) are the basis for our model-independent approach. We use the node context creation of node-context creation analyzed in Section 3.3.1 and change the neural architecture of INF2VEC to come up with the embeddings along with an estimate of an influencers aptitude. To this end, we use a multi-task neural network [31] to learn simultaneously the aptitude of an influencer to create long cascades as well as the diffusion probabilities between her and the reposters. We chose to extend the previous architecture in a multi-task learning

setting because (*i*) the problem could naturally be broken in two tasks, and (*ii*) theoretical and applied literature suggests that training linked tasks together improves overall learning [61, 163]. In our case, given an input node u, the first task is to classify the nodes it will influence and the second to predict the size of the cascade it will create. An overview of our proposed INFECTOR model can be seen in Figure 3.5.

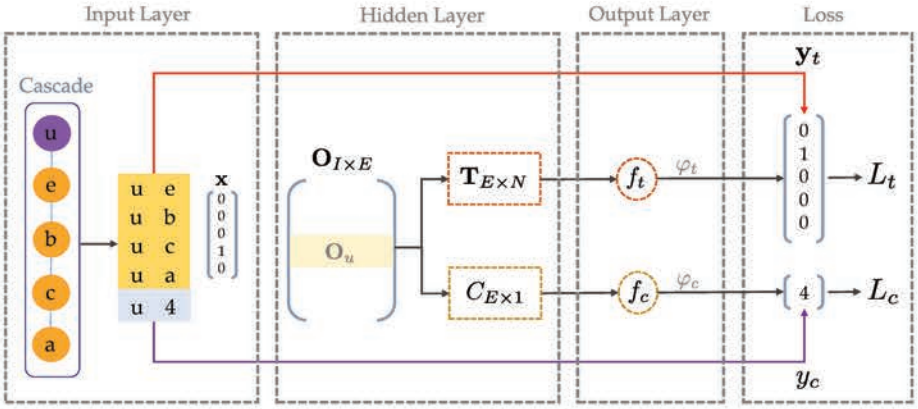


Figure 3.5: Schematic representation of the INFECTOR model. Given the cascade, a sequence of pairs is extracted, each pair consisting of the initiator node $\mathbf{x}$ (one-hot embedding) which is the *input*, and one of the "infected" nodes $\mathbf{y}_t$ i.e. e,b etc. which is the *output*. The last pair is the initiator and the cascade size y_c i.e. 4 as output. After the initator's embedding lookup through the origin embeddings $\mathbf{O}$, the vector passes through $\mathbf{T}$ and f_t if the output is another node or through $\mathbf{C}$ and f_c if the output is scalar. The loss functions are log-loss L_t for node output or mean squared error L_c for scalar output.

There are two types of inputs. The first is the training set comprised of the node-context pairs. Given a cascade t with length m, we get a set $\mathbf{X}^t = \{(\mathbf{x}^1, \mathbf{y}_t^1), (\mathbf{x}^1, \mathbf{y}_t^2), ..., (\mathbf{x}^m, \mathbf{y}_t^m)\}$, where $\mathbf{x} \in \mathbb{R}^I$ and $\mathbf{y}_t \in \mathbb{R}^N$ are one hot encoded nodes, with I the number of influencers in the train set and N the number of nodes in the network. The second is a similar set X^c, where instead of a vector e.g. $\mathbf{y}_t^1$, there is a number $\mathbf{y}_c^1$ denoting the length of that cascade, initiated by the $\mathbf{x}^1$. To perform joint learning of both tasks we mix the inputs following the natural order of the data; given a cascade, we first input the influencers-context pairs extracted from it and then the influencers-cascade length pair, as shown in Figure 3.5. $\mathbf{O} \in \mathbb{R}^{I\times E}$, with E being the embeddings size, represents the source embeddings, $\mathbf{O}_u \in \mathbb{R}^{1\times E}$ the embeddings of cascade initiator u, $\mathbf{T} \in \mathbb{R}^{E\times N}$ the target embeddings and $C \in \mathbb{R}^{E\times 1}$ is a constant vector initialized to 1. Note that $\mathbf{O}_u$ is retrieved by the multiplication of the one-hot vector of u with the embedding matrix $\mathbf{O}$.

The first output of the model represents the diffusion probability $p_{u,v}$ of the source node u for a node v in the network. It is created through a softmax function and its loss function is the cross-entropy. The second output aims to regress the cascade length, which has undergone min-max normalization relative to the rest of the cascades in this set, and hence a sigmoid function is used to bound the output at $(0, 1)$. The respective equations can be seen in Table 3.2. Here y_t is a one-hot

representation of the target node and y_c is the normalized cascade length. We employ a non-linearity instead of a simple regression because without it, the updates induced to the hidden layer from the second output would heavily overshadow the ones from the first. Furthermore, we empirically observed that the update of a simple regression would cause the gradient to explode eventually. Moreover, variable C is a constant vector of ones that is untrainable, in order for the update of the regression loss to change only the hidden layer. This change was motivated by experimental evidence.

	Classify Influenced Node	*Regress Cascade Size*
Hidden	$\mathbf{z}_{t,u} = \mathbf{O}_u \mathbf{T} + b_t$	$\mathbf{z}_{c,u} = \mathbf{O}_u C + b_c$
Output	$\boldsymbol{\varphi}_t = \frac{e^{(\mathbf{z}_{t,u})}}{\sum_{u' \in G} e^{\mathbf{z}_{t,u'}}}$	$\boldsymbol{\varphi}_c = \frac{1}{1+e^{-(\mathbf{z}_{c,u})}}$
Loss	$L_t = \mathbf{y}_t \log(\boldsymbol{\varphi}_t)$	$L_c = (y_c - \varphi_c)^2$

Table 3.2: The layers of INFECTOR.

The training happens in an alternating manner, meaning when one output is activated the other is idle, thus only one of them can change the shared hidden layer at a training step. For example, given a node u that starts a cascade of length 4 like in Figure 3.5, $\mathbf{O}_u$ will be updated 4 times based on the error of $\frac{\partial L_t}{\partial \mathbf{O}}$, and one based on $\frac{\partial L_c}{\partial \mathbf{O}}$, using the same learning rate e for the training step τ:

$$\mathbf{O}_{u,\tau+1} = \mathbf{O}_{u,\tau} - e\frac{\partial L}{\partial \mathbf{O}} \tag{3.2}$$

The derivatives from the chain rule are defined as:

$$\frac{\partial L}{\partial \mathbf{O}} = \frac{\partial L}{\partial \varphi}\frac{\partial \varphi}{\partial z}\frac{\partial z}{\partial \mathbf{O}}. \tag{3.3}$$

The derivatives in Eq. (3.3) differ between regression and classification. For the loss function of classification L_c, we have that

$$\frac{\partial \mathbf{z}_c}{\partial \mathbf{O}} = \mathbf{T}^\top \tag{3.4}$$

$$\frac{\partial \boldsymbol{\varphi}_t}{\partial \mathbf{z}_t} = \begin{cases} \boldsymbol{\varphi}_t^u(1 - \boldsymbol{\varphi}_t^v), & u = v \\ -\boldsymbol{\varphi}_t^u \boldsymbol{\varphi}_t^v, & u \neq v \end{cases} \tag{3.5}$$

where u is the input node and v is each of all the other nodes, represented in different dimensions of the vectors $\boldsymbol{\varphi}_t$. If we create a matrix consisting of N replicates of $\boldsymbol{\varphi}_t$, we can express the Jacobian as this dot product:

$$\frac{\partial \boldsymbol{\varphi}_t}{\partial \mathbf{z}_t} = \boldsymbol{\Phi}_t \cdot (I - \boldsymbol{\Phi}_t)^\top, \quad \boldsymbol{\Phi}_t = \begin{bmatrix} \boldsymbol{\varphi}_t \\ \vdots \\ \boldsymbol{\varphi}_t \end{bmatrix} \in \mathbb{R}^{N \times N} \tag{3.6}$$

$$\frac{\partial L_t}{\partial \boldsymbol{\varphi}_t} = \mathbf{y_t}\left(-\frac{1}{\boldsymbol{\varphi_t}}\right) \tag{3.7}$$

The derivatives for the regression task are:

$$\frac{\partial \mathbf{z}_c}{\partial \mathbf{O}} = C^\top \tag{3.8}$$

$$\frac{\partial \varphi_c}{\partial \mathbf{z}_c} = \frac{1}{1+e^{-\mathbf{z}_c}}\left(1-\frac{1}{1+e^{-\mathbf{z}_c}}\right) = \varphi_c(1-\varphi_c) \tag{3.9}$$

$$\frac{\partial L_c}{\partial \varphi_c} = 2(y_c-\varphi_c)\frac{(-\varphi_c)}{\partial \varphi_c} = -2(y_c-\varphi_c) \tag{3.10}$$

From Eqs. (3.3), (3.4), (3.6), (3.7), (3.8), (3.9),(3.10) we have:

$$\frac{\partial L}{\partial \mathbf{O}} = \begin{cases} -\frac{y_t}{\varphi_t}(\boldsymbol{\Phi}_t \cdot (I-\boldsymbol{\Phi}_t)^\top)\mathbf{T}^\top \\ -2(y_c-\varphi_c)(\varphi_c(1-\varphi_c))\mathbf{C}^\top \end{cases} \tag{3.11}$$

As mentioned above, the aim here is for the embedding of an influencer (hidden layer $\mathbf{O}$) to not only capture who she influences but also her overall aptitude to create lengthy cascades. To be specific, for each node v inside the cascade, O_u and T_v will undergo updates using the upper formula from Eq. (3.11) to form the diffusion probabilities, depending on $\boldsymbol{\varphi_t}$ and $\boldsymbol{y_t}$. We see that the upper update is formed based on a combination of the gradients, hence each dimension of the update vector might be positive or negative. In contrast, when updating the parameters for the regression task, we multiply a constant vector C with a scalar value. Thus, the update will increase or decrease the overall norm of the embedding analogously to the difference of the predicted and the actual cascade size.

The main difference between INFECTOR and similar node-to-node influence learning methods is that it computes diffusion probabilities and does not require the underlying social network, in contrast to influence probabilities which are assigned to edges of the network [27, 66, 179]. Apart from capturing the aforementioned higher order correlations, DPs will allow us to overcome the problems induced in influence maximization by the diffusion models as well as surpass the computational bottleneck of the repeated simulations. Please note that our method's final purpose is influence maximization, which is why we focus so much on the activity of influencers and overlook the rest of the nodes. If we aimed for a purely prediction task, such as cascade size or next infected node prediction, it is not clear that our learning mechanism would be effective.

3.4 INFLUENCE MAXIMIZATION WITH INFLUENCER VECTORS

3.4.1 *Diffusion with Influencer Vectors*

In the second part of the methodology, we aim to perform fast and accurate influence maximization using the representations learnt by adapted INF2VEC or INFECTOR and their properties. Initially, we will use the combination of the representations which form the *diffusion probability* of every directed pair of nodes. The DPs derived from the network can define a *matrix*:

$$\mathbf{D} = \begin{bmatrix} f_t(\mathbf{O}_1 T) \\ \vdots \\ f_t(\mathbf{O}_I T) \end{bmatrix} \tag{3.12}$$

which consists of the nodes that initiate train cascades (influencers) in one dimension and all susceptible nodes in the other. Even though influencers are fewer than the total nodes, $\mathbf{D}$ can still be too memory-demanding for real-world datasets. To overcome this, we can keep the top $P\%$ influencers based on the norm of their influencer embedding $|\mathbf{O}_u|$ to reduce space, depending on the device. Recall that, the embeddings are trained such that their norm captures the influencers' potential to create lengthy cascades, because $\mathbf{O}_u C = \sum_i^E \mathbf{O}_u(i) = |O|$, since C is constant.

Given the diffusion matrix $\mathbf{D}$, formally, the probability of a node v getting influenced by a seed set S is the complementary probability of not getting influenced by each node $u \in S$. Summing this over all non-seed nodes gives the number of nodes the S can influence:

$$\sigma(S) = \sum_{v \in G/S} \left(1 - \prod_{u \in S} (1 - p_{u,v})\right) \tag{3.13}$$

To transform this into a set function that computes the infected nodes, we can use a threshold such that a node will get infected if its probability of getting influenced by the seed set at this step is equal to or more than 0.5, which is the value used in classifiers with softmax output. Unfortunately, it is easy to see that this function is not submodular. A toy example with two source nodes a, b that can influence three other nodes with probabilities $[0.6, 0.3, 0.5]$ and $[0.3, 0.4, 0.4]$, respectively. In the first step, the algorithm will choose a as it gives 2 infected nodes opposing to b that gives 0. In the second step, following our definition, the addition of b will infect the second and final node, thus the property of diminishing returns does not stand for this influence spread and we can not utilize the greedy algorithm. Moreover the optimization of this non-convex function is non-trivial.

Alternatively, $\mathbf{D}$ can be interpreted as a bipartite network where the left side nodes are the candidate seeds for influence maximization, and each of them can influence every node in the right side, where the rest of the network resides. Since all edges are directed from left to right, no paths with length more than 1 exist. This means that the probability of an edge can only define the infection of one node, and it is independent

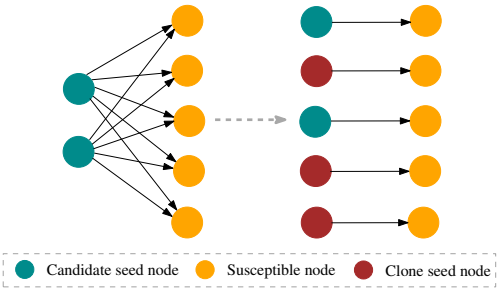


Figure 3.6: Example of cloning candidate seeds for perfect bipartite matching.

of the infection of the rest—hence, we do not require a diffusion model to estimate the spread. If we view the diffusion probabilities as edge weights, a simplification used extensively in similar context [38, 81], we can transform the problem into weighted bipartite matching. However, in this case, each candidate seed (left side) can be assigned to more than one nodes (right side). The matching can become perfect by creating a number of clone seeds for each candidate seed, and assigning them to the nodes the initial seed would be assigned to, in a one-to-one fashion, as can be seen in Figure 3.6.

The number of clones is equal to the expectation of a candidate seed u's influence spread λ_u. Cloning each node u for λ_u times transforms $\mathbf{D}$ into a balanced, weighted bipartite graph, for which we could use algorithms such as the Munkres assignment [153]. Unfortunately, it has a complexity of $\mathcal{O}(n^4)$ and requires constructing a non-sparse $N \times N$ matrix, which renders it prohibitive for real-world datasets. Though we can not use it directly, this interpretation remains useful for future directions.

To be specific regarding our assumptions and the difference with diffusion models, we revisit the source of the computational demand, which we briefly discussed in Chapter 2. Simulations are performed in order to account for all 2^e possible combinations of edges. This happens because an edge might produce the infection of more than the target node e.g. when it is a bridge between two communities. This can not happen in our setting by the definition of diffusion probability: if a node can influence another node, no matter how far that is in the original network, it will be depicted in their edge weight. That is clear in the first iteration, where the infection of a seed can only cause the infection of its neighbors hence simulations are not required. To keep this property in the subsequent iterations (where the candidate's influenced spread overlaps with previous seeds) we make the assumption that each node will influence for sure a certain number of nodes, based on INFECTOR's or adapted INF2VEC's representations. Hence these nodes can be removed, as they should not be recalculated in the influence spread of subsequent candidate seeds. This pertains to the adaptive influence maximization which is also known to exhibit similar theoretical guarantees [76].

3.4.2 *Cost Effective Lazy Forward with Influence Embeddings (CELFIE)*

One can observe that a seed's influence spread in the bipartite network is related only to its edges and the edges of the rest of seed set defined by matrix D. This characteristic can induce a drastic change to the computation of the influence spread in Eq. (3.13). In the typical live-edge model, edges are randomly sampled from the network and their joint probability in Eq. (2.3) is used to weigh the influence spread. During this process, edges that are unrelated to the seeds can be sampled, because they may form a new path from a seed to multiple uninfected nodes, which causes the computation of the joint probability of the whole network in Eq. (2.3). However, in our case sampling an edge that is unrelated to the seed set will never produce an infected node, because we do not have any paths. It should be noted here that we assume the diffusion probabilities are independent, and the amount of nodes a seed can influence is not fixed in the sense that if a node that would be influenced by a seed is already influenced, does not interfere with the rest of the nodes that a seed can influence.

Thus, in the sampling process of this simulation model, we do not have to take into account the probability of the whole network, but only of the seed set under consideration. We can update $P(g)$ in Eq. (2.2) to depict this effect as follows:

$$P(g) = \prod_{s \in S} \prod_{e \subseteq \tilde{N}} p_e \prod_{e' \in N(s) \setminus \tilde{N}} (1 - p_{e'}), \tag{3.14}$$

where $N(s)$ are the neighbors of the seed s, and $\tilde{N} \sim N(s)$ is a sample of them of size ϵ. This not only allows to restrain our sampling to the seed at hand, but also separate the sampling of the seed set with the sampling of the candidate seed. In this way instead of sampling from the whole network, computing the sample's joint probability and finding the live paths from the seed set in each simulation, we can sample solely for each candidate seed and combine it with the sampling of the seed set in the respective simulation from the previous iterations. To this end, we can redefine the computation of the marginal gain and the algorithm itself.

To measure the marginal gain of a candidate seed c, we use Algorithm 4. We perform a fixed number of simulations, where each one consists of sampling a fixed number of c's edges (line 4), according to their probabilities. Subsequently, we unite these nodes with the ones that have been infected by the seed set in that respective simulation (line 5), which gives the candidates influenced set. The cumulative quantity from all simulations subtracted by the same quantity of the seed set up to now defines the marginal gain. The influence spread is basically the same with the traditional σ [106], the only difference being sampling from fewer edges. We can thus contend that it is submodular and monotonic increasing.

The CELF with Influence Embeddings (CELFIE) method can be seen in Algorithm 5. It is an adapted implementation of CELF [121]. We use Algorithm (4) as a subroutine to estimate the marginal gain c in

Algorithm 4 CELFIE MARGINAL INFLUENCE SET (MIS)

Input: Influence likelihood matrix D, index of candidate seed c, influenced set up to now inf_set,number of simulations l, number of edges to sample ϵ

Output: Updated influenced set inf_set

procedure MIS($D, c, inf\_set, l, \epsilon$)
  set gain = 0, n = size(D)[0]
  for $i \leftarrow 0; i < l; i++$ **do**
    $c\_inf \leftarrow Sample(\epsilon, n, D[c, :])$
    $inf\_set[i] = inf\_set[i] \cup c\_inf$
  return inf_set

Algorithm 5 CELFIE

Input: Influence likelihood matrix D , number of simulations l, number of edges to sample ϵ, seed set size $size$

Output: Seed set S

procedure CELFIE($D, l, \epsilon, size$)
  set $q = [], S = []$
  for $i \leftarrow 0; i < l; i++$ **do**
    $inf\_set[i] \leftarrow \emptyset$
  for $c \leftarrow 0; c < N[1]; c++$ **do**
    $infs \leftarrow MIS(D, c[0], inf\_set, l, \epsilon)$
    $q.append([c, infs, 0])$
  $q \leftarrow Sort(q, 1, Desc = True)$
  while $S.size() < size$ **do**
    $c \leftarrow q[0]$
    if $c[2] == size(S)$ **then**
      $S.add(c[npos])$
      $inf\_set = c[1]$
      $q.delete(c)$
    else
      $infs \leftarrow MIS(D, c[0], inf\_set, l, \epsilon)$
      $mg \leftarrow 0$
      for $i \leftarrow 0; i < l; i++$ **do**
        $mg+ = |infs[i]| - |inf\_set[i]|$
      $c \leftarrow [mg, infs, |S|]$
      $q \leftarrow Sort(q, 1, Desc = True)$
  return S

line 18 and 19. Variables l is the number of simulations and ϵ is the number of edges to sample during each simulation. The list q is used to store a node's id and influence set, as well as the iteration it was last updated. We can use CELF because the influence spread is submodular and monotonic as mentioned above. This algorithm is an adaptation of the algorithm introduced in Section 2.2.2.

Note that, CELFIE utilizes the diffusion probabilities computed from an INF2VEC with the context creation introduced in 3.3.1. The main problems with this model is that the estimated influence spread is

unbounded. Since influencers tend to be copied many times in real cascades, the diffusion probabilities can become quite substantial. This leaves room for significant overestimations for influencers with very strong connections in the bipartite network, which in turn means that the inf_set parameters may include the whole V without S having reached its budget. We thus have to find a way to normalize the amount of influence a node can exert. To do this we will use INFECTOR's representations and Eq. (3.16).

3.4.3 *Influence Maximization with Influencer Vectors (IMINFECTOR)*

Since INFECTOR embeds in $|\mathbf{O}_u|$ of a candidate seed u's how potentially big is the cascade it initiates, will utilize it to compute the fraction of all N nodes u is expected to influence:

$$\lambda_u = \left\lceil N \frac{||\mathbf{O}_u||_2}{\sum_{u' \in \mathcal{I}} ||\mathbf{O}_{u'}||_2} \right\rceil \tag{3.15}$$

where $\mathcal{I}$ is the set of candidate seeds. The term resembles the norm of u relative to the rest of the influencers and the size of the network. In other words it is computing the amount of the network u will influence. Since we know a seed s can influence a certain number of nodes, we can use the diffusion probabilities to identify the top λ_s nodes it connects to. Moreover, we have to take into account the values of the DPs, to refrain from selecting nodes with big λ_s but overall small probability of influencing nodes. Consequently, the influence spread is defined by the sum of the top λ_s DPs:

$$\sigma'(s) = \sum_{j}^{\lambda_s} \hat{\mathbf{D}}_{s,j} \tag{3.16}$$

where $\hat{\mathbf{D}}_s$ is the DPs of seed s sorted in descending order. Once added to the seed set, as mentioned above the seed's influence set is considered infected and is removed from $\mathbf{D}$. This means that a seed's spread will never get bigger in two subsequent rounds, and we can employ CELF to accelerate our solution.

To retain the theoretical guarantees of the greedy algorithm $(1 - 1/e)$, we need to prove the monotonicity and submodularity of the influence spread that we optimize in each iteration. To do this, we need to separate the influence spread of the seed set S throughout the different steps of the algorithm. Let $i(s)$ be the step before node s was inserted in S, S^i be seed set at step i, $\mathbf{D^i}$ the DP matrix at step i and u the current candidate seed.

Corollary 3.4.0.1. *The influence spread is monotonic increasing.*

Proof.

$$\sigma'(S^{i(u)} \cup \{u\}) = \sum_{s \in S^{i(u)} \cup \{u\}} \sum_{j}^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} = \sum_{s \in S^{i(u)}} \sum_{j}^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} + \sum_{j}^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)} = \sigma'(S^{i(u)}) + \sum_{j}^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)} \geq \sigma'(S^{i(u)}) \tag{3.17}$$

□

Corollary 3.4.0.2. *The influence spread is submodular.*

Proof.

$$\sigma'(S^{i(u)-1} \cup \{u\}) - \sigma'(S^{i(u)-1}) = \sum_{s \in S^{i(u)-1} \cup \{u\}} \sum_{j}^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} - \sum_{s \in S^{i(u)-1}} \sum_{j}^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} = \sum_{j}^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)-1} \tag{3.18}$$

$$\sigma'(S^{i(u)} \cup \{u\}) - \sigma'(S^{i(u)}) = \sum_{j}^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)} \tag{3.19}$$

$\hat{\mathbf{D}}_{u,j}^{i(u)}, \forall j \in \lambda_u$ are the top DPs of u for nodes that are left uninfected from the seeds in $S^{i(u)}$. Note here that, $\hat{\mathbf{D}}_{u,j}^{i(u)}, \forall j \in \lambda_u$ are the top DPs of u for nodes that are left uninfected from the seeds in $S^{i(u)}$. So, the influence spread of u is reverse analogous to the influence spread of the seed set up to that step:

$$S^{i(u)-1} \subseteq S^{i(u)} \Leftrightarrow \sum_{s \in S^{i(u)-1}} \sum_{j}^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} \leq \sum_{s \in S^{i(u)}} \sum_{j}^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} \Leftrightarrow \sum_{j}^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)-1} \geq \sum_{j}^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)} \tag{3.20}$$

□

The algorithm is given at Algorithm 6 and is an adaptation of CELF using the proposed influence spread and updates on $\mathbf{D}$. We keep a queue with the candidate seeds and their attributes (line 2) and the uninfected nodes (line 3). The attributes of a candidate seed are its influence set (line 5) and influence spread (line 6), as defined by Eq. (3.16). We sort based on the influence spread (line 8) and proceed to include in the seed set the candidate seed with the maximum marginal gain, which is Ω after the removal of the infected nodes. Once a seed is chosen, the nodes it influences are marked as infected (line 14). The DPs of the candidate seeds are reordered after the removal of the infected nodes (line 17) and the new marginal gain is computed (line 18).

An alternative to Eq. (3.15) would be to keep the actual percentage of nodes influenced by each seed, i.e. $||\mathbf{O}_u||_2/N$. This approach, however,

Algorithm 6 IMINFECTOR

Input: Probability diffusion matrix D, expected influence spread λ, seed set size ℓ

Output: Seed set S

procedure IMINFECTOR(D, λ, ℓ)
set $q \leftarrow [\,], S \leftarrow \emptyset$
$F = 0 : \dim(D)[1]$
for $s = 0; s < \dim(D)[0]; s++$ **do**
$R \leftarrow \text{argsort}(D[s, F])[0 : \lambda[s]]$
$\Omega \leftarrow \text{sum}(D[s, R])$
$q.\text{append}([s, \Omega, R, 0])$
$q \leftarrow \text{sort}(q, 1)$
while $|S| < \ell$ **do**
$u \leftarrow q[0]$
if $u[3] == |S|$ **then**
$S.\text{add}(u[0])$
$R \leftarrow u[2]$
$F \leftarrow F - R$
$q.\text{delete}(u)$
else
$R \leftarrow \text{argsort}(D[u[0], F])[0 : \lambda[u[0]]]$
$\Omega \leftarrow \text{sum}(D[u[0], R])$
$q[0] \leftarrow [u[0], \Omega, R, |S|]$
$q \leftarrow \text{sort}(q, 1)$
return S

suffers from a practical problem. In real social networks, influencers can create cascades with tens or even hundreds of thousands of nodes. In this case, $|\lambda_u|$ can be huge, so when our algorithm is asked to come up with a seed set of e.g. 100 nodes, matrix $\mathbf{D}$ will be emptied in the first few iterations that will include seeds covering the whole network. This would constrain our algorithm to small seed set sizes, in which case simple ranking metrics tend to be of equal strength with influence maximization, as we argue further on the evaluation methodology. Hence, we use a normalization that alleviates moderately the large differences between influence spreads and allows for bigger seed set sizes.

To give a concrete example of how the algorithm works, let's imagine a simple network with 8 nodes, as depicted in Figure 3.7, with the respective weights from each candidate seed S to each susceptible node N. In the first step, λ of a seed S defines how many of the S's top susceptible nodes should be taken into account in the computation of σ'. In this case, we keep the top 3 for $S3$ and top 2 for $S2$ and $S1$. The top 3 for $S3$ correspond to $\{N1, N3, N4\}$, as indicated by the green edges in the bipartite network, whose sum gives a $\sigma' = 0.9$. For $S2$, we could either take $\{N1, N2\}$ or $\{N1, N3\}$, both giving a $\sigma' = 0.6$, which is lower than $S3$, and the same applies for $S1$. Thus $S3$ is chosen as the seed for step 1 and the influenced nodes are removed.

In the next step, matrix $\mathbf{D}$ is updated to remove the influenced nodes and σ' is recomputed. In this case, there are only two nodes left for

Seed	$N1$	$N2$	$N3$	$N4$	$N5$	λ	σ'
S1	0.1	0.3	0.2	0.2	0.2	2	0.5
S2	0.4	0.2	0.2	0.1	0.2	2	0.6
S3	0.5	0.1	0.2	0.2	0	3	0.9

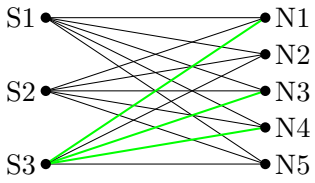


Figure 3.7: Example of IMINFECTOR: step 1.

Seed	$N2$	$N5$	λ	σ'
S1	0.3	0.2	2	0.5
S2	0.2	0.2	2	0.4

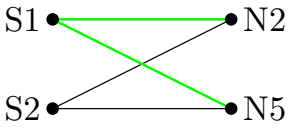


Figure 3.8: Example of IMINFECTOR: step 2.

each seed, so their influence spread is computed by the sum of all their edges, since their $\lambda = 2$. An important point here is that while in the first step $S2$ had a stronger σ' than $S1$, $S1$ is stronger now, due to the removal of $N1$, which was strongly susceptible to the influence of $S2$, but infected in the previous round by $S3$. Thus, the chosen seed in step 2 is $S1$.

Running time complexity. The first step of the algorithm is to compute the influence spread for all candidate seeds I in $\mathbf{D}$, which takes $I \cdot N \log N$, because we sort the DPs of each candidate seed to take the top λ. The sorting of q which has $\mathcal{O}(I \cdot \log I)$ before the algorithm's iterations start, adds constant time to the total complexity. With that in mind, and taking into acount the seed set size ℓ, the complexity is analogous to $\mathcal{O}(\ell \cdot I(I \log I) \cdot (N \log N))$, similar to CELF, but with sorting N in every evaluation of the influence spread. In practice, in every iteration, N is diminished by an average of λ because of the removed influence set, so the final logarithmic term is much smaller. Finally, due to the nature of CELF, much fewer influence spread evaluations than I take place in reality (line 16 in the algorithm), which is why our algorithm is fast in practice.

3.5 EXPERIMENTAL EVALUATION

3.5.1 *Datasets*

It is obvious that we can not use networks from the influence maximization literature because we require the existence of ground truth diffusion cascades. Although such open datasets are still quite rare, we managed to assemble three, two social and one bibliographical, to provide an estimate of performance in different sizes and information types. Table 6.4 gives an overview of the datasets.

	Digg	*MAG*	*Sina Weibo*
Nodes	279,631	1,436,158	1,170,689
Edges	2,251,166	15,928,078	225,877,808
Cascades	3,553	181,020	115,686
Avg Cascade size	847	29	148

Table 3.3: Summary of the datasets used.

- *Digg*: A directed network of a social media where users follow each other and a vote to a post allows followers to see the post, thus it is treated as a retweet [118].
- *MAG Computer Science*: We follow suit from [175] and define a network of authors with undirected co-authorship edges, where a cascade happens when an author writes a paper and other authors cite it. In other words, a co-authorship is perceived as a friendship, a paper as a post and a citation as a repost. In this case, we employ Microsoft Academic Graph (MAG) [187]. To construct the network and diffusion cascades from MAG, we utilized the tables PaperReferences, Author, PaperAuthorAffiliation,PaperFields and FieldsofStudy from the official from the official site[2]. Initially, we removed some one-paper authors based on a pattern we analyze in Chapter 6. Subsequently, we kept only papers with fields that are included in the Computer Science "field of study". To create the coauthorship network, we formed a graph based on the authors' coappearence in these papers using the 'PaperAuthorAffiliation' table. To form the diffusion cascades between authors, we had to derive a sequence of papers (citing other papers) using 'Paper References'. Subsequently, we transform it into a cascade over our network by substituting the papers with their authors, and forming one separate cascade for each author of the initial paper (as all of them can be considered initiators). The authors of a certain paper that is included in the cascade appear in a random sequence

2 https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/

accompanied with the same timestamp, which corresponds to the time between the initial paper's publication and the time this paper was published. We remove cascades with length less than 10.

- *Sina Weibo*: A directed follower network where a cascade is defined by the first tweet and the list of retweets [236]. We remove from the network nodes that do not appear in the cascades i.e. they are practically inactive, in order to make more fair the comparison between structural and diffusion-based methods, since the evaluation relies on unseen diffusions.

3.5.2 *Baseline Methods*

Most traditional influence maximization algorithms, such as greedy [106] and CELF [121], do not scale to the networks we have used for evaluation. Thus, we employ only the most scalable approaches:

- K-CORES: The top nodes in terms of their core number, as defined by the undirected k-core decomposition. This metric is extensively used for influencer identification and it is considered the most effective structural metric for this task [138, 139].
- AVG CASCADE SIZE: The top nodes based on the average size of their cascades in the train set. This is a straightforward ranking of the nodes that have proven effective in the past [13].
- SIMPATH: A heuristic that capitalizes on the locality of influence pathways to reduce the cost of simulations in the influence spread [81]. (The *threshold* is set to 0.01).
- CREDIT DISTRIBUTION: This model uses cascade logs *and* the edges of the network to assign influence credits and derive a seed set [80]. (Parameter λ is set to 0.001).
- IMINFECTOR: Our final model with embedding size equal to 50, trained for 5 epochs with a learning rate of 0.1. The reduction percentage P is set to 10 for *Weibo* and *MAG*, and 40 for *Digg*. CELFIE is also employed with similar hyperparameters.

The parameters are the same as in the original papers. Comparing network-based methods such as SIMPATH and IMM with methods that use both the network and cascades, is not a fair comparison as the latter capitalize on more information. To make this equitable, each network-based method is coupled with two influence learning (IL) methods that provide the influence maximization methods with influence weights on the edges, using the diffusion cascades:

- DATA-BASED (DB): Given that the follow edge $u \to v$ exists in the network, the edge probability is set to $A_{u \to v} / A_u$, i.e. the number of times v has copied (e.g. retweeted) u, relative to the total activity (e.g. number of posts) of u [79]. Any edge with zero

probability is removed, and the edges are normalized such that the sum of weighted out-degree for each node is 1.

- INF2VEC: A shallow neural network performing influence learning based on the co-occurence of nodes in diffusion cascades and the underlying network [66]. It was proven more effective than similar influence learning methods [27, 179] in the tasks of next node prediction and diffusion simulation.

3.5.3 *Evaluation Methodology*

We split each dataset into train and test cascades based on their time of occurrence. The methods utilize the train cascades and/or the underlying network to define a seed set. The train cascades amount for the first 80% of the whole set and the rest is left for testing. The evaluation is twofold: computational time and seed set quality, similar to previous literature in influence maximization. We evaluate the quality of the predicted seed set using the number of *Distinct Nodes Influenced* (DNI)[54, 164, 169], which is the combined set of nodes that appear in the test cascades that are initiated from each one of the chosen seeds. To be specific, each predicted seed in the seed set has started some cascades in the test set. We compute the set of all infected nodes (DNI) by simply adding every node appearing in these test cascades, in a unified set. The size of this set indicates a measure of how successful was that seed set on the test set. To understand the choice of DNI we need to take into consideration that each seed can create several test cascades. For example, another metric we could use is the sum of the average cascade length of each seed's test cascades [225]. With this approach though, we fail to counter for the overlap between different seeds' spreads, and hence we can end up with influential seeds whose influence spreads overlaps a lot, resulting in an eventually smaller number of infected nodes. As mentioned above, DNI maintains a set of all distinct nodes participating in cascades started by each of the seeds, hence tackling the potential overlap issue. Overall, the most significant advantage over other standard influence maximization evaluations is that it relies on actual spreading data instead of simulations. Although not devoid of assumptions, it is the most objective measure of a seed set's quality, given the existence of empirical evidence.

Since our datasets differ significantly in terms of size, we have to use different seed set size for each one. For *MAG* which has 205,839 initiators in the train set, we test it on 10,000, *Weibo* with 26,158 is tested on 1,000, and *Digg* with 537 has a seed set size of 50. This modification is crucial to the objective evaluation of the methods because small seed sets favor simpler methods. For example, the top 100 authors based on K-CORES in *MAG* would unavoidably work well because they are immensely successful. However, increasing the seed set size allows the effect of influence overlap to take place, and eventually, simplistic methods fall short. The experiments were run on a machine with Intel(R) Xeon(R) W-2145 CPU @ 3.70GHz, 252GB ram, and an Nvidia Titan V. Any

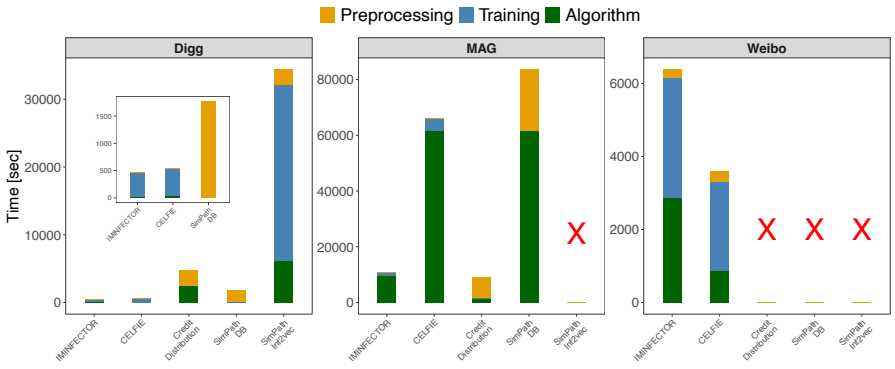


Figure 3.9: Time comparison between the methods. Methods that could not scale are marked with X.

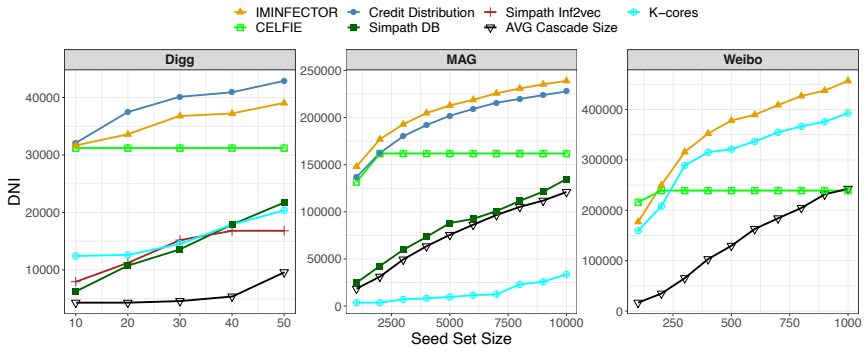


Figure 3.10: The quality of the seed set derived by each method in different sizes, measured by the seed set's number of Distinct Nodes Influenced (DNI) in the test cascades. Algorithms that failed to scale for the minimum seed set size, are not included.

method that required more memory or more than seven days to run, it is deemed unable to scale.

3.5.4 *Results*

Figure 3.9 shows the computational time of the examined methods, separated based on different parts of the model, and Figure 3.10 shows the estimated quality of each seed set. In terms of quality, we can see that IMINFECTOR surpasses the benchmarks in two of the three datasets. In *Digg*, CREDIT DISTRIBUTION performs better but is almost 10 times slower, because of the average cascade size in the dataset. All methods could scale in *MAG* except for SIMPATH with INF2VEC weights, because in contrast to DB, INF2VEC retains all the edges of the original *MAG* network, for which SIMPATH takes more than one week to run. Moreover, SIMPATH with both types of weights is not able to scale in the *Weibo* dataset, due to the number of edges. CREDIT DISTRIBUTION also fails to scale in *Weibo* due to the network's and cascade size. IMM with INF2VEC can not scale due to the high demand in memory. It can still scale with DB weights (close to 1M nodes), but it performs purely. The only method that scales successfully in Weibo was ranking by the k-core

decomposition, which took about 650 seconds. IMM's performance is lower than expected, highlighting the difference between data-driven means of evaluation and the traditional simulations of diffusion models, which have been used in the literature due to the absence of empirical data. IMM optimizes diffusion simulations as part of its solution, so it is reasonable to outperform other methods in this type of evaluation. However, it performance is not equal in a metric based on real influence traces. This can be attributed largely to the aforementioned problems of diffusion models and their miscalculation of the influence spread.

Compared to CELFIE, the previous version of our algorithm, IMINFECTOR has superior performance as the seed set size increases. One of the CELFIE's core disadvantages is that the marginal influence spread, which is computed by sampling repetitively from a node's neighbors and summing the edge's weights, decreases very fast during the seeding rounds. This has as a result the whole network being covered with few rounds, and hence the retrieved seed set size to always be bounded. In IMINFECTOR, each node has an expected influence spread as shown in Eq. (3.15), which we have defined based on a node's influencer vector, compared to the rest of the initiator's representations. In other words, this formulation imposes the constraint that the network is shared among all candidate seeds, depending on their influencer vector, rendering infeasible to cover the whole graph without putting all initiators in the seed set. Moreover, it allows for a fixed computation of a node's influence spread (e.g. without resampling), which renders it faster the CELFIE which retrieves only 127 seeds for Weibo, 2059 for MAG and 4 for Digg.

In terms of influence learning methods, DB outperformed INF2VEC because it removes edges with no presence in the cascades, diminishing significantly the size of the networks, some even close to 90%. However, it also had a severe shortcoming most notable in the social networks. Due to the general lack of consistent reposters and the huge amount of posts, the total activity of a node (the denominator in edge weight) was much larger than the number of reposts by a follower (numerator). Hence, the edge weights were too small with insignificant differences, alleviating the computation of the simulated influence spread. This effect is not so strong in the bibliographical network, because authors tend to cite the same popular authors more consistently. A side observation is the heavy computational burden of INF2VEC. It accounts for most of the computational time (blue color), which justifies the context creation mechanism of INFECTOR, while the accuracy validates our hypothesis that influencers' success lies more on the cascades they initiate rather their reposts.

In general, we see that IMINFECTOR provides a fair balance between computational efficiency and accuracy. Most importantly, it exhibits such performance using *only the cascades*, while the rest baselines use both, the network and cascades. Being unaffected by the network size, IMINFECTOR scales with the average cascade size and the number of cascades, which makes it suitable for real world applications where the networks are too big and scalability is of utmost importance.

3.6 CONCLUSION

In this chapter, we have proposed IMINFECTOR, a model-independent method to perform influence maximization using representations learned from diffusion cascades. The machine learning part learns pairs of representations from diffusion cascades, while the algorithmic part uses the representations' norms and combinations to extract a seed set. The algorithm outperformed several methods based on a data-driven evaluation in three large scale datasets.

It should be noted that our method is not devoid of assumptions. Firstly, we assume that the optimum seed set is comprised solely of nodes that initiate cascades. Given the task of influence maximization, this seems a valid assumption, with the exception of cases where very popular people in the real world enter the social network and get hyperconneccted without sharing content. Apart from this assumption, our method is highly scalable; a system though with too many cascade initiators (close to the number of users), will increase the overall complexity. A simple solution to this is to remove initiators with minuscule activity, which, given the exponential distribution characterizing the users' activity, will diminish severely the number. Finally, we tested our method in three datasets of varying sizes and characteristics, but there are further types of social networks that we need to experiment with to certify our method's effectiveness to the fullest.

A potential future direction is to examine deeper and more complex architectures for INFECTOR to capture more intricate relationships. From the algorithmic part, we can experiment with bipartite matching algorithms, given sufficient resources. Overall, the main purpose of this work is primarily to examine the application of representation learning in the problem of influence maximization and secondarily to highlight the importance of data-driven evaluation (e.g. DNI). In addition, we want to underline the strength of model-independent methods and evaluations, given the recent findings on the severe shortcomings of diffusion models. We hope this will pave the way for more studies to tackle influence maximization with machine learning means.

4

LEARNING GRAPH REPRESENTATIONS FOR INFLUENCE MAXIMIZATION

In the previous chapter, we developed an influence maximization method for datasets with diffusion cascades. However, numerous real-world networks are not accompanied with ground truth cascades. We thus develop a neural method for influence maximization based on diffusion models. In this chapter, we develop GLIE, a Graph Neural Network (GNN) that inherently parameterizes an upper bound of influence estimation and train it on small simulated graphs. Experiments show that GLIE provides accurate influence estimation for real graphs up to 10 times larger than the train set. More importantly, it can be used for influence maximization on considerably larger graphs, as the predictions ranking is not affected by the drop of accuracy. We develop a version of Cost Effective Lazy Forward optimization with GLIE instead of simulated influence estimation, surpassing the benchmark for influence maximization, although with a computational overhead. To balance the time complexity and quality of influence, we propose two different approaches. The first is a Q-network that learns to choose seeds sequentially using GLIE's predictions. The second defines a provably submodular function based on GLIE's representations to rank nodes fast while building the seed set. The latter provides the best combination of time efficiency and influence spread, outperforming SOTA benchmarks.

SOURCE CODE The implementation of the proposed model can be found online [1].

4.1 INTRODUCTION

Several real-world problems can be cast as a combinatorial optimization problem over a graph. From distributing packages [144] to improving the general health [221] and vehicles' management [201], optimization on graphs lies in the core of many real-world applications that are vital to our way of living. Unfortunately, the majority of these problems are NP-hard, and hence we can only approximate their solution in a satisfactory time limit that matches the real world requirements. Recent machine learning methods have emerged as a promising solution to develop heuristic methods that provide fast and accurate approximations [18]. The general idea is to train a supervised or unsupervised learning model to infer the solution given an unseen graph and the problem constraints. The models tend to consist of Graph Neural Networks (GNNs) to encode the graph and the nodes, Q-learning [192, 217] to produce sequential predictions, or a combination of both. The practical motivation behind learning to solve combinatorial optimization problems, is that inference

1 https://github.com/geopanag/graph_rep_for_im

time is faster than running an exact combinatorial solver [100]. That said, specialized combinatorial algorithms like CONCORDE for the *Traveling Salesman Problem* (TSP) or GUROBI in general, cannot be surpassed yet [108].

Though many such methods have been proposed for a plethora of problems, influence maximization has not been addressed yet extensively. Influence maximization is defined in Chapter (2), where we underline that it is NP-hard and is approximated through submodular maximization. Moreover, the influence estimation problem that is embedded in influence maximization, is also mentiond in Chapter 2) to be #P-hard and approximated using repetitive Monte-Carlo simulations of the chosen diffusion model. As mentioned above, due to the inherent computational problems of both problems, several scalable algorithms[22, 196] and heuristics [38, 101] were developed capitalizing on sketches or the structure of the graph to produce more efficient solutions. Influence maximization can be applied on a plethora of real-world tasks, such as epidemic containment [222], diminishing fake news [30], and running viral marketing campaigns [49].

We address influence maximization using neural networks, to capitalize on the aforementioned advantages as well as their ability to easily incorporate contextual information such as user profiles and topics [199], a task that remains unsolvable for non-specialized influence maximization algorithms and heuristics. We propose GLIE, a GNN that provides efficient influence estimation for a given seed set and a graph with influence probabilities. It can be used as a standalone influence predictor with competitive results for graphs up to 10 times larger than the train set. Moreover, we leverage GLIE for influence maximization, combining it with CELF [121], that typically does not scale beyond networks with thousands of edges. The proposed method runs in networks with millions of edges in seconds, and exhibits better influence spread than a state-of-the-art algorithm and previous GNN-RL methods for influence maximization. In addition, we develop GRIM, a Q-learning architecture that utilizes GLIE's representations and predictions to obtain seeds sequentially, while minimizing the number of influence estimations throughout steps. Finally, we propose PUN, a method that uses GLIE's representations to compute the number of neighbors predicted to be uninfluenced and uses it as an approximation to the marginal gain. We prove PUN's influence spread is submodular and monotone, and hence can be optimized greedily with a guarantee, in contrast to prior learning-based methods. The experiments indicate that PUN provides the best balance between influence quality and computational efficiency.

The chapter is organized as follows. Section 4.2 presents an overview of relevant approaches and clarifies the advantage of the proposed models. Section 4.3 describes the proposed methods, starting with influence estimation and advancing progressively towards faster methods for influence maximization. Section 4.4 exhibits and interprets the experimental results for influence estimation and influence maximization. Finally, Section 4.5 summarizes the contribution and presents future steps.

4.2 RELATED WORK

The first approach to solving combinatorial optimization (CO) using neural networks was based on attention-based NNs for discrete structures, PointerNets [209], followed by an architecture that combines PointerNets with an actor-critic training to find the best route for TSP [17]. The first architecture that utilized graph-based learning was S2n-Dqn [47], usingStruct2Vec to encode the states of the nodes and the graph, and training a Q-learning model that chooses the right node to add in a solution given the current state.

Based on S2v-Dqn, a Dqn [150] for the network dismantling problem was recently proposed [64] [125]. The model, named Finder, uses a deep Q-learning architecture where the representations are derived by three GraphSage layers. The reward is based on size of the giant connected component size, i.e. every new node (seed) chosen, aims to dismantle the network as much as possible. Some of the main advantages of Finder is that it is trained on small synthetic data, which are easy to make, and can extrapolate to relatively large graphs. On the other hand, one of the core disadvantages is that it can not work with directed graphs and weighted edges. Another recent supervised deep learning approach on influence maximization, Gcomb [142], utilizes a probabilistic greedy to produce scores on graphs and trains a GNN to predict them. A Q-network receives the scores along with an approximate calculation of the node's neighborhood correlation with the seed set, to predict the next seed. This approach, though scalable and comparable to SOTA in accuracy, has to be trained on a large random subset of the graph (30% of it) and tested on the rest. This makes the model graph-specific, i.e. it has to be retrained to perform well on a new graph. This imposes a serious overhead, considering the time required for training, subsampling and labeling these samples using the probabilistic greedy method with traditional influence estimation. As shown in [142] Appendix G, it takes at least hundreds of minutes and is thus out of our scope. Finally, recent works on learning approximations to submodular policies [8] require a large number of ground truth evaluation to create the training trajectories, which rendering the training too time consuming. Moreover, such methods require a novel neural network encoding to capture the state of influence maximization, which has not been developed yet.

In this chapter, we propose an approach that combines the advantages of the aforementioned methods, in that it is only trained on small simulated data once and generalizes to larger graphs, and it addresses the problem of influence maximization in weighted directed networks. Furthermore, the approach can be broken down to a GNN for influence estimation and three influence maximization methods. The former can act alone as influence predictor and be competitive with relevant methods, such as Dmp [131] for graphs up to one scale larger than the train set. Glie is used to propose Celf-Glie, Celf [121] with Glie as influence estimator, Grim, a Q-network that learns how to choose seeds using the Glie's estimations and hidden representations, and Pun, an adaptive influence maximization method that optimizes greedily a submodular influence spread using Glie's representations.

Symbol	Meaning	Type
N	number of nodes	scalar
$\mathbf{A}$	column stochastic transition matrix	matrix
$\mathbf{X}$	input features	matrix
$\mathbf{H_t}$	activations of the hidden layer t	matrix
$\mathbf{W_t}$	learnable parameters of the hidden layer t	matrix
$p_{u,v}$	probability of u influencing v	scalar
S	Set of seed nodes	set
L_S, L'_S	influence set representations	vector
m_S	approximation to the marginal gain	vector
$\hat{\sigma}(s), \sigma^m(s)$	influence spread of node set s	scalar

Table 4.1: Table of symbols.

We note here that the majority of the relevant literature on DL for CO address small graphs [47, 108, 174, 209] which makes them not applicable to our task. More scalable, unsupervised methods citepkaralias2020erdos are tailored to specific problems and is non-trivial to adjust them to our problem, with the exception of [128] which was found significantly worse than the SOTA algorithm we compare with in [142].

4.3 METHODOLOGY

4.3.1 *Graph Learning for Influence Estimation (GLIE)*

In this section, we introduce GLIE (**G**NN **L**earning **I**nfluence **E**stimation), which aims to learn how to estimate the influence of a seed set S over a graph $G = (V, E)$. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a transition matrix and $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the features of nodes, representing which nodes belong to the seed set by 1 and 0 otherwise:

$$\mathbf{X}_u = \begin{cases} \{1\}^d, & u \in S \\ \{0\}^d, & u \notin S \end{cases} . \tag{4.1}$$

For the following analysis, we set $d = 1$. More dimensions will become meaningful when we parameterize the problem. If we normalize $\mathbf{A}$ by each row, we form a row-stochastic transition matrix, as:

$$\mathbf{A}_{uv} = p_{vu} = \begin{cases} \frac{1}{\deg(u)}, & v \in \mathcal{N}(u) \\ 0, & v \notin \mathcal{N}(u) \end{cases} , \tag{4.2}$$

where $\deg(u)$ is the in-degree of node u and $\mathcal{N}(u)$ is the set of neighbors of u. Based on the weighted cascade [106], each row u stores the probability of node u being influenced by each of the other nodes that are

connected to it by a directed link $v \to u$. Note that, in case of directed influence graphs, $\mathbf{A}$ should correspond to the *transpose* of the adjacency matrix. The influence probability $p(u|S)$ resembles the probability of a node u getting influenced if its neighbors belong in the seed set, i.e. during the first step of the diffusion. We can use message-passing to compute a well-known upper bound $\hat{p}(u|S)$ of $p(u|S)$ for node u:

$$\hat{p}(u|S) = \mathbf{A}_u \cdot \mathbf{X} = \sum_{v \in \mathcal{N}(u) \cap S} \frac{1}{\deg(u)} = \tag{4.3}$$

$$\sum_{v \in \mathcal{N}(u) \cap S} p_{vu} \geq 1 - \prod_{v \in \mathcal{N}(u) \cap S} (1 - p_{vu}) = p(u|S), \tag{4.4}$$

where the second equality stems from the definition of the weighted cascade and the inequality from the proof in [240], App. A. As the diffusion covers more than one-hop, the derivation requires repeating the multiplication to approximate the total influence spread. To be specific, computing the influence probability of nodes that are not adjacent to the seed set requires estimating recursively the probability of their neighbors being influenced by the seeds. If we let $\mathbf{H}_1 = \mathbf{A} \cdot \mathbf{X}$, and we assume the new seed set S^t to be the nodes influenced in the step $t-1$, their probabilities are stored in $\mathbf{H}_t$, much like a diffusion in discrete time. We can then recompute the new influence probabilities with $\mathbf{H}_{t+1} = \mathbf{A} \cdot \mathbf{H}_t$,

Corollary 4.3.0.1. *The repeated product $\mathbf{H}_{t+1} = \mathbf{A} \cdot \mathbf{H}_t$ computes an upper bound to the real influence probabilities of each infected node at step $t+1$.*

Corollary 4.3.0.2. *The repeated product $\mathbf{H}_{t+1} = \mathbf{A} \cdot \mathbf{H}_t$ computes an upper bound to the real influence probabilities of each infected node at step $t+1$.*

Proof. We have:

$$\begin{aligned}
\hat{p}^t(u|S^t) &= \mathbf{A}_u \cdot \mathbf{H}_t && (4.5)\\
&= \sum_{v \in \mathcal{N}(u) \cap S^t} \hat{p}_v p_{vu} && (4.6)\\
&\geq \sum_{v \in \mathcal{N}(u) \cap S^t} p_v p_{vu} && (4.7)\\
&\geq 1 - \prod_{v \in \mathcal{N}(u) \cap S^t} (1 - p_v p_{vu}) && (4.8)\\
&= p^t(u|S^t) && (4.9)\\
& && (4.10)
\end{aligned}$$

- (4.7) stems from Eq. 4.4 in the manuscript:

$$\hat{p}(u|S) = \mathbf{A}_u \cdot \mathbf{X} = \sum_{v \in \mathcal{N}(u) \cap S} \frac{1}{\deg(u)} = \tag{4.11}$$

$$\sum_{v \in \mathcal{N}(u) \cap S} p_{vu} \geq 1 - \prod_{v \in \mathcal{N}(u) \cap S} (1 - p_{vu}) = p(u|S). \tag{4.12}$$

- (4.8) can be proved by induction similar to [240]. For every $p_v \leq 1$, the base case $\sum_{v\in\mathcal{X}} p_v p_{vu} \geq 1 - \prod_{v\in\mathcal{X}}(1 - p_v p_{vu})$ is obvious for $|\mathcal{X}| = 1$. For $|\mathcal{X}| > 1$, we have:

$$\begin{aligned}
1 - \prod_{v\in\mathcal{X}}(1 - p_v p_{vu}) &= 1 - (1 - p_x p_{xu}) \prod_{v\in\mathcal{X}\setminus x}(1 - p_v p_{vu}) \\
&= 1 - \prod_{v\in\mathcal{X}\setminus x}(1 - p_v p_{vu}) + p_x p_{xu} \prod_{v\in\mathcal{X}\setminus x}(1 - p_v p_{vu}) \\
&\leq \sum_{v\in\mathcal{X}\setminus x} p_v p_{vu} + p_x p_{xu} \prod_{v\in\mathcal{X}\setminus x}(1 - p_v p_{vu}) \\
&\leq \sum_{v\in\mathcal{X}\setminus x} p_v p_{vu} + p_x p_{xu} \\
&= \sum_{v\in\mathcal{X}} p_v p_{vu}. \qquad (4.13)
\end{aligned}$$

- (4.9) we have $p(u|v) = p_v p_{vu}$ per definition of the independent cascade, and consequently $p(u|S) = 1 - \prod_{v\in\mathcal{N}(u)\cap S}(1 - p_v p_{vu})$, where $p_v = 1$ for $v \in S^1$, which are the initial seed set that are activated deterministically. We can thus contend that Eq. (4.10) stands, and the computed probabilities are an upper bound of the real influence probabilities. Hence the influence spread, which is computed as $\hat{\sigma}(S) = \sum_{(u,v)\in E} p_{uv}$ is also an upper bound to the real $\sigma(S)$.

□

In reality, due to the existence of cycles, two problems arise with this computation. Firstly, if the process is repeated the influence of the original seeds may increase again, which comes in contrast with the independent cascade model. This can be controlled by minimizing the repetitions, e.g., four repetitions cause the original seeds to be able to reinfect other nodes in a network with triangles. To this end, we leverage up to three neural network layers. Another problem due to cycles pertains to the probability of neighbors influencing each other. In this case, the product of the complementary probabilities in Eq. (4.4) does not factorize for the non-independent neighbors. This effect was analyzed extensively in Lokhov and Saad [131], App. B, and proved that the influence probability computed by $p(u|S)$ is itself an upper bound on the real influence probability for graphs with cycles. Intuitively, the product that represents non-independent probabilities is larger than the product of independent ones. This renders the real influence probability, which is complementary to the product, smaller than what we compute.

We can thus contend that the estimation $\hat{p}(u|S)$ provides an upper bound on the real influence probability—and we can use it to compute an upper bound to the real influence spread of a given seed set i.e. the total number of nodes influenced by the diffusion. Since message-passing can compute inherently an approximation of influence estimation, we can parameterize it to learn a function that tightens this approximation based on supervision. In our neural network architecture, each layer consists of a GNN with a batchnorm and dropout omitted here, and starting from $\mathbf{H}_0 = \mathbf{X} \in \mathbb{R}^{n\times d}$ we have:

$$\mathbf{H}_{t+1} = \mathrm{ReLU}([\mathbf{H}_t, \mathbf{A}\mathbf{H}_t]\mathbf{W}_0). \qquad (4.14)$$

The readout function that summarizes the graph representation based on all nodes' representations is a summation over all the final representations with skip connections:

$$\mathbf{H}_S^G = \sum_{v \in V} [\mathbf{H}_0^v, \mathbf{H}_1^v, \ldots, \mathbf{H}_t^v]. \tag{4.15}$$

This representation captures the probability of all nodes being active throughout each layer. The output that represents the predicted influence spread is derived by:

$$\hat{\sigma}(S) = \text{ReLU}(\mathbf{H}_S^G \mathbf{W}_o). \tag{4.16}$$

Note that, the derived representations of each layer H_t^i, untrained, are the upper bound of the influence probability of seed set's the t-hop neighbors. The parameters of the intermediate layers W_t are trained such that the upper bound is reduced and the final layer W_o can combine the probabilities to derive a cumulative estimate for the total number of influenced nodes. We empirically verify this by examining the layer activations which can be seen in Figure 4.2 and the heatmaps indicate a difference between columns (nodes) expected to be influenced, meaning we could potentially predict not only the number but also who will be influenced. However, since $\hat{\sigma}$ is derived by multiple layers, the relationships and thresholds to determine the exact influenced set is not straight forward. Below we experiment with different such sets extracted from H for the purpose of influence maximization.

4.3.2 *Cost Effective Lazy Forward with GLIE (CELF-GLIE)*

CELF is analyzed in Chapter 2. In our case, we propose a straight forward adaptation where we substitute the original CELF Influence Estimation based on Monte Carlo Independent Cascade with the output of GLIE. Initially we design an analysis to empirically prove that GLIE's output is submodular and monotonous. For the real datasets, we use the seed set retrieved by GLIE-CELF and a random seed set to quantify the differences between subsequent estimations. To be specific, we have a sequence S that represents the seed set and a sequence R that represents the random nodes, with S_j being the seed set up to j^{th} element and s_j being the j^{th} element, and similarly r_j for R. We compute the marginal gain to check for monotonicity:

$$m_{ss} = \hat{\sigma}(S_j \cup s_{j+1}) - \hat{\sigma}(S_j) \tag{4.17}$$

$$m_{sr} = \hat{\sigma}(S_j \cup r_{j+1}) - \hat{\sigma}(S_j), \tag{4.18}$$

and for submodularity, we have , with $i = j - 1$:

$$s_{ss} = (\hat{\sigma}(S_i \cup s_{j+1}) - \hat{\sigma}(S_i)) - (\hat{\sigma}(S_j \cup s_{j+1}) - \hat{\sigma}(S_j)) \tag{4.19}$$

$$s_{sr} = (\hat{\sigma}(S_i \cup r_{j+1}) - \hat{\sigma}(S_i)) - (\hat{\sigma}(S_j \cup r_{j+1}) - \hat{\sigma}(S_j)). \tag{4.20}$$

In Figure 4.1, we plot m and s for some of our datasets. Regarding s, since we require a constant node, we randomly sample one of the seeds s_j and a random node r_j and visualize the sequences of both s with regard to adding them in every step. The values of these functions correspond to nodes, and range from tens to thousands, depending on the datasets. For monotonicity and submodularity, we verify that m and s are always more than zero. Moreover, we see that they decrease with the size of the seed set, as well as the observation that adding a random seed provides worst marginal gains (in monotonicity plots) than adding the chosen seed.

Since GLIE are monotonous and submodular in practice, $\hat{\sigma}$ is suitable for optimizing with CELF, though we can not obtain any theoretical guarantee. CELF-GLIE has two main computational bottlenecks. First, although it alleviates the need to test every node in every step, in practice it still requires Influence Estimation for more than one nodes in each step. Second, it requires computing the initial Influence Estimation for every node in the first step. We will try to alleviate both with the two subsequent methods.

4.3.3 *Graph Reinforcement Learning for Influence Maximization (GRIM)*

We develop a method that computes only one influence estimation in every step, along with the initial influence estimation for all nodes. We first utilize the activations mentioned above to define the influence set representation $L_S \in \{0,1\}^n$, which can be computed by adding the activations of each layer $\mathbf{H}_t$, summing along the axis of the hidden layer size, and thresholding to get a binary vector:

$$L_S = \mathbb{1}\left\{\sum_{t=0}^{T} \frac{\sum_{i=0}^{d_t} \mathbf{H}_t^i}{d_t} \geq 0\right\}, \tag{4.21}$$

where T is the number of layers, and $\mathbf{H}_t^i \in \mathbb{R}^{n\times 1}$ is a column from $\mathbf{H}_t$. This vector contains a label for each node whose sign indicates if it is predicted to be influenced. We compute the average representation because d_t varies throughout layers, and since we add all layer's outputs, we need an equal contribution from each layer's dimension to the final output. We utilize this to compute the difference between the influence of the current seed set and the initial influence of each other node.

We aim to build a method that learns how to pick seeds sequentially. The model needs to receive information from GLIE regarding the state (graph and seed set), and decide on the next action (seed). Note that, GLIE can not provide a direct estimate of a new candidate's s marginal gain without rerunning GLIE$(S \cup s, G)$, which is what we try to avoid. To this end, we utilize a double Q-network [205] and the model is depicted in Figure 4.2 (middle and right part). During the first step, GLIE provides an influence estimation for all candidate seeds, and the node with the highest is added to the seed set, similar to GLIE-CELF. We also keep a list of each node's initial influence set L_s. Subsequently, the Q-network produces a Q-value for each node s using as input the

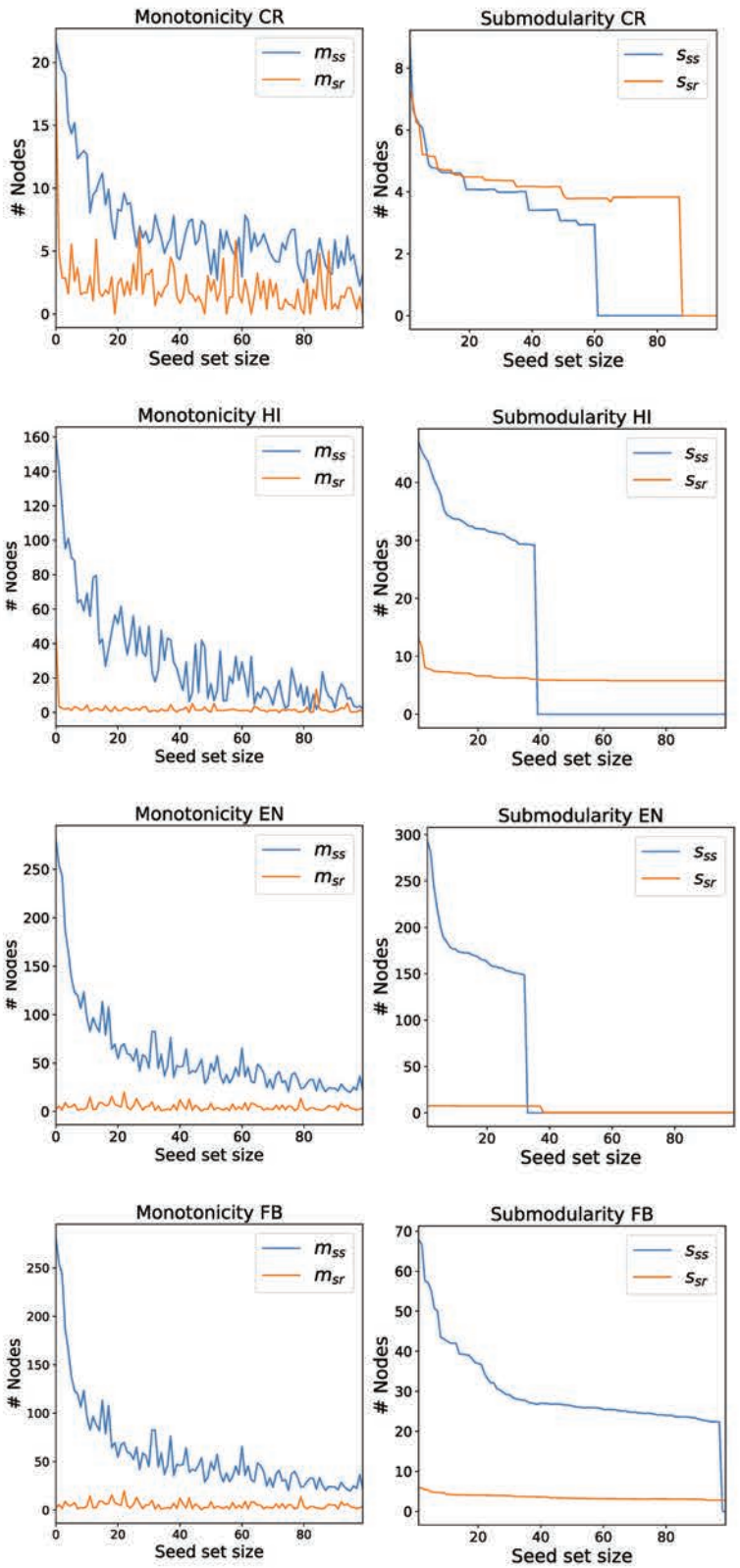
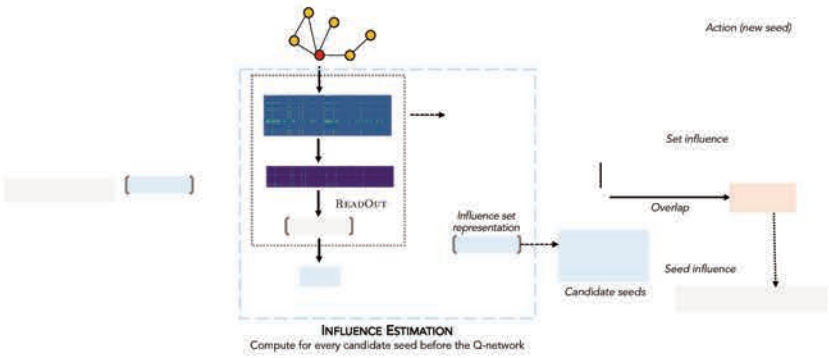


Figure 4.1: Monotonocity and submodularity in our datasets.

estimated influence of the current seed set $\hat{\sigma}(S)$, the initial influence of the node $\hat{\sigma}(s)$, and the interaction between them. The interaction is defined as the difference between their corresponding influence sets $O(S, s) = \sum_{i=0}^{n} \mathbb{1}\{L_s^i - L_S^i \geq 0\}$, as predicted by GLIE. The latter aims


Action (new seed)
Set influence
Overlap
READOUT
Influence set representation
Seed influence
Candidate seeds
INFLUENCE ESTIMATION
Compute for every candidate seed before the Q-network

a simpler influence set representation then in 4.21. Let $\hat{L}_S, L'_S \in \{0,1\}^n$ be the binary vectors with 1s in nodes predicted to be uninfluenced and nodes predicted to be influenced respectively:

$$\hat{L}_S = \mathbb{1}\left\{\sum_{i=0}^{d_1} \mathbf{H}_1^i \leq 0\right\} \qquad L'_S = \mathbb{1}\left\{\sum_{i=0}^{d_1} \mathbf{H}_1^i \geq 0\right\} \tag{4.24}$$

L'_S is simpler than L_S defined in Eq. (4.21) and provides a more rough estimate, but it allows for a simpler influence spread which we can optimize greedily

$$\sigma^m(S) = |L'_S| \tag{4.25}$$

We can use $\hat{L}_S$ and message-passing to predict the amount of a node's neighborhood that remains uninfluenced, i.e. the **P**otentially **U**ninfluenced **N**eighbors (PUN), weighted by the respective probability of influence For a node u,

$$m_S[u] = \sum_{v \in N(u)} A_{u,v}\hat{L}_v = A_u^\top \cdot \hat{L}_S \in \mathbb{R}^{n\times 1} \tag{4.26}$$

For efficiency, we can compute $m_S = A^T\hat{L}$ which can be considered an approximation to all nodes marginal gain on their immediate neighbors. We can thus optimize this using $\arg\max_S(m_S)$, as shown in Figure (4.2). In order to establish that σ^m can be optimized greedily with a theoretical guarantee of $(1-\frac{1}{e})$OPT, we prove its monotonocity and submodularity.

Theorem 4.3.0.1. *The influence spread σ^m is submodular.*

For the purposes of the proof, $X_i \in \{0,1\}^{n\times d}$ is the input and $H_i \in \mathbb{R}^{n\times hd}$ is the output of the first neural layer for the input seed set S_i, and $P \in \{1\}^{hd\times 1}$. Moreover we define the support function $supp(v) = \{i \in [1,d], v_i \neq 0\}$ [99] as the set of indices of non zero elements in vector $v \in \mathbb{R}^d$. Finally let R represent ReLU and b_{tr}, st_{tr} the mean and standard deviation computed by the batchnorm. Each step is justified further below.

Proof. Monotonocity, $\forall i < j, S_i \subset S_j$:

$$supp(X_j) \supset supp(X_i) \tag{4.27}$$
$$supp(X_jW) \supseteq supp(X_iW) \tag{4.28}$$
$$supp(AX_jW) \supseteq supp(AX_iW) \tag{4.29}$$
$$supp(R(AX_jW)) \supseteq supp(R(AX_iW)) \tag{4.30}$$
$$supp\left(\frac{R(AX_jW) - b_{tr}}{st_{tr}}\right) \supseteq supp\left(\frac{(R(AX_iW) - b_{tr})}{st_{tr}}\right) \tag{4.31}$$
$$supp(H_j) \supseteq supp(H(S_i)) \tag{4.32}$$
$$supp(H_jP) \supseteq supp(H_iP) \tag{4.33}$$
$$|\mathbb{1}_{>0}\{H_jP\}| \geq |\mathbb{1}_{>0}\{H_iP\}| \tag{4.34}$$
$$|L'_j| \geq |L'_i| \tag{4.35}$$
$$\sigma^m(S_j) \geq \sigma^m(S_i) \tag{4.36}$$
$$\tag{4.37}$$

1. (4.27) stems by the definition of X in Eq. (4.1).
2. (4.28) X_j is a convex hull that contains X_i [28]. We multiply both sides by a real matrix $W \in \mathbb{R}^{d \times hd}$ which can equally dilate both convex hulls in terms of direction and norm. This equal transformation cannot change the sign of the difference between the elements of X_i and X_j and hence cannot interfere with the support of X_j over X_i. The statement becomes more obvious for $X \in \{0,1\}^{n \times 1}$ and $W \in \mathbb{R}^{1 \times 1}$. Note that both can result in zero matrices so we use subset or equal.
3. (4.29) A is a non-negative matrix.
4. (4.30) ReLU is a non negative monotonically increasing function.
5. (4.31) Subtract by the same number and divide by the same positive number.
6. (4.32) Definition in Eq. (4.14).
7. (4.33) P is positive.
8. (4.34) By definition of the support.
9. (4.35) By definition of L'_S.

□

For the proof of submodularity we have to define $X_{iu} = X_{S_i \cup u}, u \in V$ and note by the definition of the input that $|X_{ju} - X_j| = |X_{iu} - X_i|$ for the l_1 norm (sum of all elements):

Proof. Submodularity $\forall i < j, S_i \subset S_j, ,$:

$$|X_{ju} - X_j| = |X_{iu} - X_i| \tag{4.38}$$

$$A|X_{ju} - X_j| = A|X_{iu} - X_i| \tag{4.39}$$

$$|A(X_{ju} - X_j)| = |A(X_{iu} - X_i)| \tag{4.40}$$

$$|AX_{ju} - AX_j| = |AX_{iu} - AX_i| \tag{4.41}$$

$$|AX_{ju}W - AX_jW| = |AX_{iu}W - AX_iW| \tag{4.42}$$

$$R(|AX_{ju}W - AX_jW|) - 2b_{tr} = R(|AX_{iu}W - AX_iW| - 2b_{tr}) \tag{4.43}$$

$$|R(AX_{ju}W) - R(AX_jW) - 2b_{tr}| = |R(AX_{iu}W) - R(AX_iW) - 2b_{tr}| \tag{4.44}$$

$$supp(R(AX_{ju}W) - R(AX_jW) - 2b_{tr}) = supp(R(AX_{iu}W) - R(AX_iW) - 2b_{tr}) \tag{4.45}$$

$$supp(R(AX_{ju}W - b_{tr})) - supp(R(AX_jW) - b_{tr}) \subseteq supp(R(AX_{iu}W - b_{tr})) - supp(R(AX_iW) - b_{tr}) \tag{4.46}$$

$$supp(H_{ju}) - supp(H_j) \subseteq supp(H_{iu}) - supp(H_i) \tag{4.47}$$

$$\sigma^m(S^j \cup \{u\}) - \sigma^m(S^j) \leq \sigma^m(S^i \cup \{u\}) - \sigma^m(S^i) \tag{4.48}$$

$$\tag{4.49}$$

1. (4.41) Distributive property
2. (4.42) Similar to multiplication by A.
3. (4.46) The norm of the difference is distributed equally, but the right hand difference has as least the same or more positive elements because the norm of A ,which is stochastic ,is bounded by V hence X_u can give up to the same gain to AX_j and AX_i, the same number b_{tr} is subtracted, and more elements are activated by X_j then X_i as shown in Eq. 4.35.
4. (4.47) We skipped dividing by st_{tr} for brevity.
5. (4.48) Arrive with similar steps as 4.33 - 4.36.

□

Regarding the approximation of the marginal gain we first show that choosing the node corresponding to the maximum m_S will give the maximum L'_{ju}: $A'_u\hat{L}_j \geq A'_v\ \hat{L}_i \Rightarrow L'_{ju} \geq L'_{iv}$.

$$A'_u\hat{L}_j = \sum_{v \in N(u)} A'_{uv}\hat{L}_j[v] = \sum_{v \in N(u)} A_{uv}L'_j[u] = \sum_{v \in N(u)} A_{uv}X_{ju} \tag{4.50}$$

$$\tag{4.51}$$

This means that m_S gives the node u that improves the biggest number of rows in AX_{ju} that are not already considered influenced. Since we know from Eq. 4.36 that a $AX_{iu} \geq AX_{iv} \Rightarrow |L'_{iu}| \geq |L'_{iv}|$, the claim concludes. Hence choosing the best node using the marginal gain approximation is as good as the real influence spread. Now we prove the submodularity of the proposed marginal gain.

Proof. Submodularity for the approximation of the marginal gain, $\forall i < j, S_i \subset S_j$, starting from (4.34):

$$|\mathbb{1}_{>0}\{H_j P\}| \geq |\mathbb{1}_{>0}\{H_i P\}|$$
$$|\mathbb{1}_{\leq 0}\{H_j P\}| \leq |\mathbb{1}_{\leq 0}\{H_i P\}| \tag{4.52}$$
$$A'_u \hat{L}_j \leq A'_u \hat{L}_i \tag{4.53}$$
$$m_{S_j}[u] \leq m_{S_i}[u] \tag{4.54}$$
$$(|L'_j| + m_{S_j}[u]) - |L'_j| \leq (|L'_i| + m_{S_i}[u]) - |L'_i| \tag{4.55}$$
$$\sigma^m(S^j \cup \{u\}) - \sigma^m(S^j) \leq \sigma^m(S^i \cup \{u\}) - \sigma^m(S^i) \tag{4.56}$$

1. (4.36) Complementarity between elements that are ≤ 0 and elements > 0.
2. (4.38) Definition in Eq. (4.24) and multiply with non-negative row u from matrix A'.
3. (4.39) Definition in Eq. (4.26).
4. (4.40) Adding and subtracting $|L_j|$ and $|L_i|$.
5. (4.41) By definition of σ^m in Eq. (4.25) and the marginal gain of u, we arrive at submodularity in Eq. (4.41).

□

PUN can be seen in the left part of Figure 4.2. We start by setting the first seed as the node with the highest degree, which can be considered a safe assumption as in practice it is always part of seed sets. We use GLIE(S, G) to retrieve $\hat{L}_S$, which we use to find the next node based on $\arg\max_{v \in G \setminus S} m_S[v]$ and the new $\hat{L}_{S \cup \{v\}}$. One disadvantage of PUN is that σ^m is an underestimation of the predicted influence, as can be seen in Figure 4.3. Contrasted with the upper bound, DMP, σ^m is not as accurate as $\hat{\sigma}$, but allows us to compute efficiently a submodular proxy for the marginal gain. This underestimation means that a part of the network considered uninfluenced in $\hat{L}_S$ is measured as potential gain for their neighbors, hence the ranking based on m_S can be effected negatively. As we observed in Figure 4.3, the divergence of σ^m increases with the size of the seed set.

4.4 EXPERIMENTS

All the experiments are performed in a PC with an NVIDIA GPU TITAN V (12GB RAM), 256GB RAM and an Intel(R) Xeon(R) W-2145 CPU @ 3.70GHz. Regarding the reproducibility of our experiments, we have attached all python codes in the supplementary files along with

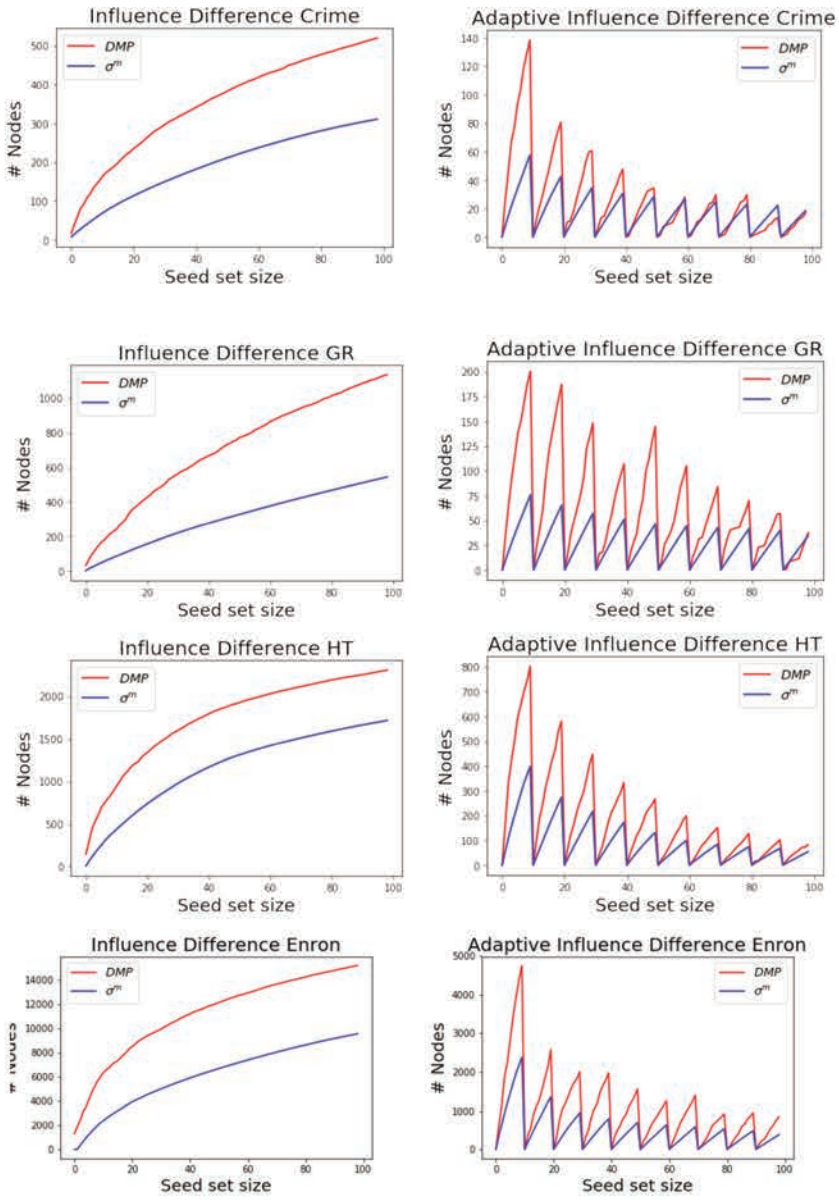


Figure 4.3: Difference between DMP influence estimate and σ^m in standard influence maximization and adaptive influence maximization with full feedback every 10 seeds, in two datasets (Crime and GR).

detailed instructions on how to reproduce the results. We also point to the python codes we utilized to run the benchmarks. The datasets utilized are given in Table (4.2). The real graphs are:

- Crime a bipartite criminal network where each node denotes a person and an edge represents that two persons are involved in the same crime [112].
- General Relativity Collaboration A coauthorship network derived from papers submited on e-print arxiv under the General Relativity and Quantum Cosmology category [120].
- HI-II-14 a protein-protein interaction network, where each node represents a protein and each edge is the interaction between these proteins [7].

	Graph	# Nodes	# Edges
Sim	Test/Train	$100 - 500$	$950 - 4,810$
	Large	$1,000 - 2,000$	$11,066 - 19,076$
Small	Crime (CR)	829	2,946
	HI-II-14 (HI)	4,165	26,172
	GR Colab (GR)	5,242	28,980
Large	Enron (EN)	33,697	361,622
	Facebook (FB)	63,393	1,633,660
	Youtube (YT)	1,134,891	5,975,246

Table 4.2: Graph datasets.

- Enron an email network that covers all the email communications in Enron. Each node is an email address, and an edge denotes at least one communication between these addresses [122].
- Facebook is a subset of the friendsship networks from Facebook users. A node represents a user and an edge represents friendship between two users [112].
- Youtube the friendship network of the video-sharing site Youtube. Nodes are users and an edge between two nodes indicates friendship [112].

4.4.1 *Influence Estimation*

For training in the influence estimation task, we create a set of labeled samples, each consisting of the seed set S and the corresponding influence spread $\sigma(S)$. We create 100 Barabasi-Albert [16] and Holme-Kim [92] undirected graphs ranging from 100 to 200 nodes and 30 from 300 to 500 nodes. 60% are used for training, 20% for validation and 20% for testing. We have used these network models because the degree distribution resembles the one of real world networks. The influence probabilities are assigned based on the weighted cascade model, i.e. a node u has equal probability $1/\deg(u)$ to be influenced by each of her $\mathcal{N}(u)$ nodes. This model requires a directed graph, hence we turn all undirected graphs to directed ones by appending reverse edges. Though estimating influence probabilities is a problem on its own [53, 165], in the absence of extra data, the weighted cascade is considered more realistic than pure random assignments [106].

We observed that $1,000$ has similar results to $10,000$, due to the small size of the networks. The training takes 500 up to $1,200$ seconds, depending on when the validation error converges. We have used a small scale grid-search to find the optimum batch size 64, dropout 0.4, number of layers 2, hidden layer size 64, and feature dimension 50. More importantly, we observed that it is beneficial to decrease the hidden layer size (by a factor of 2) as the depth increases, i.e. go from 32 to

16. This means that the 1-hop node representations are more useful compared to the 2-hop ones and so on—validating the aforementioned conclusion that the approximation to the influence estimation in Eq. (4.4), diverges more as the message-passing depth progresses.

We use two different ways to come up with the seed sets S for varying sizes of S from 1 to 5. We use random seed sets in order to capture the average influence spread expected for a seed set of about that size. This creates "average samples" which would constitute the whole dataset in other problems. In influence maximization however, the difference in σ between an average seed set and the optimum seed set can be significant, hence training solely on the random sets would render our model unable to predict larger values that correspond to the optimum. We thus add in the samples the optimum seed set for each size, taken using **Celf** and MC ICs for influence estimation. For each size, we have the 30 random seed sets that serve as negative samples and the optimum, which is a more balanced form of supervision, as you expect the crucial majority of the seed sets to have an average σ. This amounts to a total of $20,150$ training samples.

Regarding the training procedure, we have used a small scale grid-search using the validation set to find the optimum batch size 64, dropout 0.4, number of layers 2, hidden layer size 64, and feature dimension 50. More importantly, we observed that it is beneficial to decrease the hidden layer size (by a factor of 2) as the depth increases, i.e. go from 32 to 16. This means that the 1-hop node representations are more useful compared to the 2-hop ones and so on—validating the aforementioned conclusion that the approximation to the influence estimation in Eq. (4.4), diverges more as the message-passing depth increases. The training then proceeds for 100 epochs with an early stopping of 50 and learning rate of 0.01.

We evaluate the models in three different types of graphs. The first is the test set of the dataset mentioned above. The second is a set of 10 power-law large graphs ($1,000$ – $2,000$ nodes) to evaluate the capability of the model to generalize in networks that are larger by one factor. The third is the set of small real-world graphs in Table 4.2. The real graphs are evaluated for varying seed set sizes, from 2 to 10, to test our model's capacity to extrapolate to larger seed set sizes. Due to the size of the latter two graphs (HI and GR), we take for each seed set size the top nodes based on the degree as the optimum seed set along with a 30 random seed sets for the large simulated graphs and 3 for the real graphs, to validate the accuracy of the model in non-significant sets of nodes.

To quantify the potential of GLIE for larger seed sets, we sample 9 random seed sets and 1 with the highest degree nodes and compute the error of DMP and GLIE, with the ground truth influence divided by the average influence in Table (4.4). We see that the error does not increase significantly as the seed set increases, and that GLIE outperforms DMP in GR while the reverse happens in CR and HT.

We have compared the accuracy of influence estimation with DMP [131]. We could not utilize the influence estimation of UBLF [240] because its central condition is violated by the weighted cascade model

Graph (seeds)	DMP		GLIE	
	MAE	**Time**	**MAE**	**Time**
Test (1 – 5)	0.076	0.05	0.046	0.0042
Large (1 – 5)	0.086	0.44	0.102	0.0034
CR (1 – 10)	0.009	0.11	0.044	0.0029
HI (1 – 10)	0.041	2.84	0.056	0.0034
GR (1 – 10)	0.122	4.32	0.084	0.0042

Table 4.3: Average mean absolute error (MAE) divided by the average influence i.e. relative MAE and time (in seconds) throughout all seed set sizes and samples, along with the real average influence spread.

Graph	**Seeds**	DMP	GLIE
CR	20	0.005	0.031
CR	50	0.006	0.059
CR	100	0.017	0.152
GR	20	0.161	0.029
GR	50	0.125	0.042
GR	100	0.093	0.082
HT	20	0.010	0.105
HT	50	0.004	0.062
HT	100	0.002	0.113

Table 4.4: Relative MAE for diffusion prediction of larger seed sets.

and the computed influence is exaggerated to the point it surpasses the nodes of the network. The average error throughout all datasets and the average influence can be seen in Table 4.3, along with the average time.We evaluate the retrieved seed set using the independent cascade, and the results are shown in Table 4.5. We should underline here that this task would require more then 3 hours for the *Crime* dataset and days for *GR* using the traditional approach with $1,000$ MC IC. As we can see in Table 4.5, GLIE-CELF allows for a significant acceleration in computational time, while the retrieved seeds are more effective. Moreover, in CELF, the majority of time is consumed in the initial computation of the influence spread, i.e. the overhead to compute 100 instead of the 20 seeds shown in Table 4.5, amounts to 0.11, 0.22 and 0.19 seconds for the three datasets respectively.

4.4.2 *Influence Maximization*

GRIM is trained on a dataset that consists of 50 random BA graphs of $500 - 2,000$ nodes. It is trained by choosing 100 seeds sequentially, to maximize the reward (delay = 2 steps) for each network. Since the

Graph (seeds)	Seed Overlap	DMP-CELF		GLIE-CELF	
		Influence	**Time**	**Influence**	**Time**
CR(20)	14	221	83	229	1.0
HI(20)	13	1,235	8,362	1,281	5.49
GR(20)	12	295	16,533	393	7.01

Table 4.5: Influence maximization for **20** seeds with CELF, using the proposed (GLIE) substitute for influence estimation and evaluating with 10,000 MC independent cascades (IC).

immediate reward corresponds to the marginal gain, the sum of these rewards at the end of the "game" corresponds to the total influence of the seed set. An episode corresponds to completing the game for all 50 graphs; we play 500 episodes, taking roughly 40 seconds each. The exploration is set to 0.3 and declines with a factor of 0.99. The model is optimized using ADAM, as in GLIE. We store the model that has the best average influence over all train graphs in a training episode. In order to diminish the computational time of the first step in GLIE-CELF and GRIM, we focus on candidate seeds that surpass a certain degree threshold based on the distribution, a common practice in the literature [38, 142].

For comparison, we use a state-of-the-art influence maximization method, IMM [196] which capitalizes on reverse reachable sets [22] to estimate influence. Specifically, it produces a series of such influence sketches and uses them to approximate the influence spread without any simulation when building the seed set. This results in remarkable acceleration while retaining a theoretical guarantee with high probability. Note that, IMM is considered state-of-the-art and has similar influence spreads with [198], while surpassing various heuristics [101]. We set $e = 0.5$ as proposed by the authors. We also compare with FINDER, which is analyzed in Section 6.2.1, and with the most well known heuristic methods for the Independent Cascade PMIA [212], DEGREEDISCOUNT [38] and K-CORES [137] .

Graph	GLIE-CELF	GRIM	PUN	K-CORE	PMIA	DEGDISC	IMM	FINDER
CR	228	**232**	227	126	224	227	228	**232**
GR	**402**	208	374	184	342	272	387	379
HI	1,300	1,304	1,307	866	1,274	1,234	**1,316**	1,110
EN	8,189	8,185	8,205	5741	8,041	7,988	**8,208**	3,964
FB	**4,487**	4,480	4,263	1594	2491	4,489	4,481	1,823
YT	**105,897**	104,824	104,106	67,084	97,622	104,706	105,888	3,799

Table 4.6: Influence spread computed by 10,000 MC ICs for 20.

The results for the influence spread of 20,50,100 and 200 seeds as computed by simulations can be seen in Tables 4.6 to 4.9, while the time results are shown in Table 4.10 and Table 4.11. The top result is in bold and the second best is underlined. One can see that GLIE-CELF exhibits overall superior influence quality compared to the rest of the methods,

Graph	GLIE-CELF	GRIM	PUN	K-CORE	PMIA	DEGDISC	IMM	FINDER
CR	**381**	371	378	263	378	368	375	367
GR	**738**	553	700	303	654	556	725	635
HI	**1,914**	1,905	1,908	1,407	1,899	1,824	1,589	1,904
EN	**11,819**	11,114	11,757	9,796	11,686	11,183	10,698	7,133
FB	6,631	5,879	6,329	2,974	6,574	4,489	**6,724**	5,649
YT	147,631	148,250	145,796	78,575	145,863	143,161	**148,597**	9,152

Table 4.7: Influence spread computed by 10,000 MC ICs for 50.

Graph	GLIE-CELF	GRIM	PUN	K-CORE	PMIA	DEGDISC	IMM	FINDER
CR	**522**	509	521	455	520	512	516	502
GR	**1,102**	997	1,076	421	1,013	919	1,085	897
HI	2,307	1,302	**2,308**	2,024	2,291	2,229	2,290	2,274
EN	**14,920**	14,022	14,912	10,918	14,855	13,808	14,848	12,596
FB	**8,710**	7,418	8,409	4,174	5,613	8,247	8,625	5,746
YT	189,515	187,808	187,808	89,546	189,746	**194,834**	194,521	34,941

Table 4.8: Influence spread computed by 10,000 MC ICs for 100 seeds.

Graph	GLIE-CELF	GRIM	PUN	K-CORE	PMIA	DEGDISC	IMM	FINDER
CR	**661**	650	657	647	656	644	650	642
GR	1,617	1,502	**1,626**	701	1,566	1415	1,617	1,286
HI	2,685	2,631	**2,688**	2,540	2,685	2,614	2,668	2,625
EN	17,601	16,642	**17,614**	13,015	17,534	16,500	17,497	17,244
FB	10,981	9,406	10,626	6,434	7,688	10,309	**11,007**	10,801
YT	**246,439**	241,000	244,579	110,409	242,057	236,726	247,178	50,435

Table 4.9: Influence spread computed by 10,000 MC ICs for 200 seeds.

	100 **seeds**					200 **seeds**				
Graph	GLIE-CELF	GRIM	PUN	IMM	FINDER	GLIE-CELF	GRIM	PUN	IMM	FINDER
CR	1.25	0.91	0.15	**0.13**	0.41	2.00	2.03	0.25	**0.19**	0.41
GR	3.41	0.69	**0.17**	0.57	2.36	4.55	1.79	**0.26**	0.95	2.36
HI	1.20	2.59	**0.17**	0.56	1.01	2.19	0.60	**0.27**	1.29	1.01
EN	5.89	4.85	**0.52**	4.78	9.30	15.49	5.49	**0.97**	10.47	9.30
FB	120.6	100.00	**1.42**	69.90	56.8	287.7	123.95	**3.1**	171.25	56.80
YT	119.00	48.00	**13.20**	55.40	191.00	151.33	100.00	**28.92**	82.13	191.00

Table 4.10: Computational time in seconds.

	100 **seeds**				200 **seeds**			
Graph	PMIA	DEGDISC	K-CORE	PUN	PMIA	DEGDISC	K-CORE	PUN
CR	0.13	**0.04**	**0.04**	0.15	0.21	0.06	**0.04**	0.25
GR	0.70	**0.12**	1.5	0.17	0.80	**0.13**	1.5	0.26
HI	1.24	0.13	**0.12**	0.17	1.36	0.14	**0.12**	0.27
EN	24.83	1.96	2.17	**0.52**	26.74	2.06	2.17	**0.97**
FB	21.2	8.86	10.62	**1.42**	22.77	9.29	10.62	**3.1**
YT	3838.5	52.39	74.91	**13.2**	4006.29	54.38	74.91	**28.92**

Table 4.11: Computational time of heuristic approaches compared to PUN.

but is quite slower. GRIM is slightly faster than GLIE-CELF but is the second slowest method. This quantifies the substantial overhead caused by computing the influence spread of all candidate seeds in the first step.

Their time difference amounts to how many more influence estimations GLIE-CELF performs in every step compared to GRIM, which performs only one. This is more obvious with PUN, which requires only one influence estimation in every step and no initial computation. It is from 3 to 60 times faster than IMM while its computational overhead moving from smaller to larger graphs is less than linear to the number of nodes. In terms of influence quality, PUN is first or second in the majority of the datasets and this effect becomes more clear as the seed set size increases. DEGDISC is faster than PUN in smaller graphs but slower in larger and overall worse in seed set quality. PMIA provides medium seed set quality but is computationally ineficient. IMM is clearly not the fastest method, but it is very accurate, specially for smaller seed set sizes. FINDER exhibits the least accurate performance, which is understandable given that it solves a relevant problem and not exactly influence maximization for IC. The computational time presented is the time required to solve the node percolation, in which case it may retrieve a bigger seed set than 100 nodes. Thus, we can hypothesize it is quite faster for a limited seed set, but the quality of the retrieved seeds is the least accurate among all methods. Overall, we can contend that PUN provides the best accuracy-efficiency tradeoff from the examined methods.

Moreover we performed experiments to compare PUN without the use of GPU in Table 4.12, where it is visible that GPU provides a substantial acceleration, but PUN remains the faster option even without it. Finally, we juxtapose PUN and IMM on the same graphs with uniform influence probabilities $p = 0.01$, (a common alternative to WC) in Figure 4.4. We clearly observe that PUN outperforms IMM in this case as well.

Graph	PUN GPU	PUN CPU	IMM
CR	0.15	0.17	0.13
GR	0.17	0.27	0.57
HT	0.17	0.20	0.56
EN	0.52	2.44	4.78
FB	1.42	17.5	69.9
YT	13.2	97.5	55.4

Table 4.12: Comparison between PUN CPU and GPU computational times for 100 seeds.

4.4.3 *Comparison with IMINFECTOR*

Here we perform a comparison between IMINFECTOR and PUN in the datasets of Chapter 3. PUN requires a GPU with larger then 24 GB RAM to run for Weibo, so we kept the comparison in Digg and MAG. The results at Figure 4.5 validate our intuition that PUN, which is trained on the network based on the independent cascade, performs better than IMINFECTOR in traditional evaluation based on monte

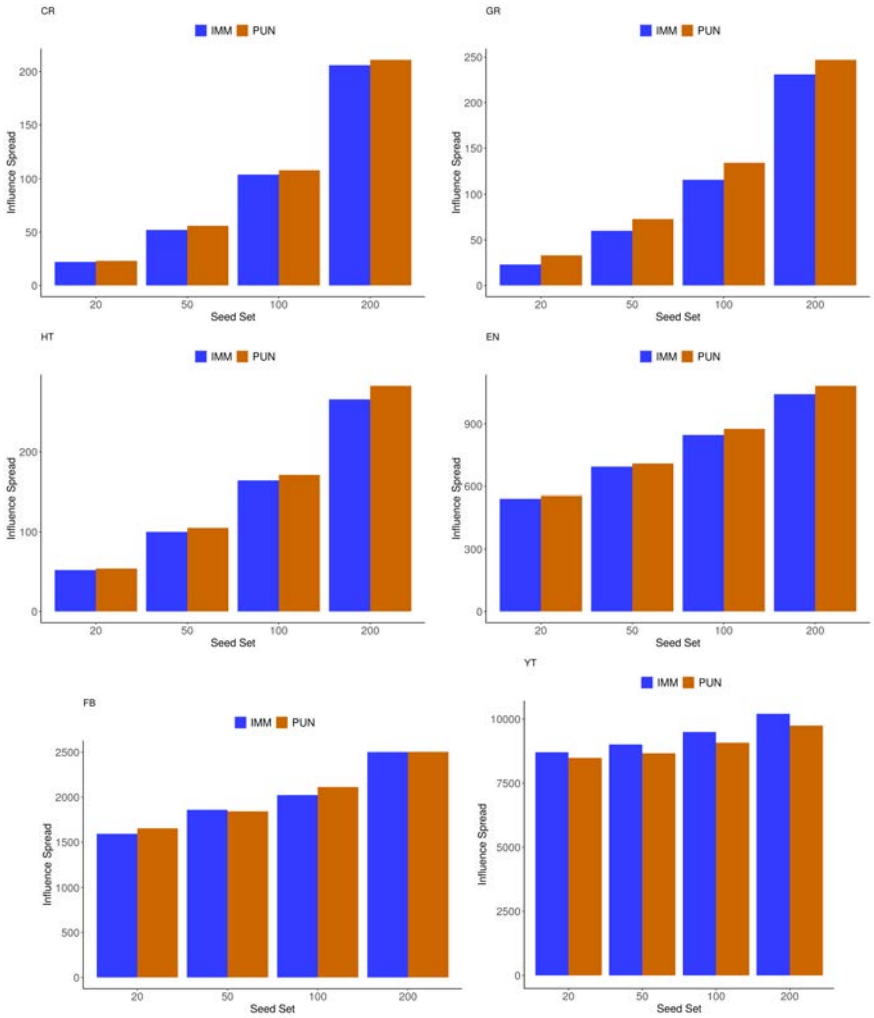


Figure 4.4: PUN vs. IMM for IC with $p = 0.01$.

carlo simulations. On the contrary, IMINFECTOR outperforms PUN significantly in the DNI introduced in Section 3.3.1, as it is based on the unseen cascades. This stark contrast underlines the lack of combined methods for evaluation. An influence maximization seed set should aim to have a balanced performance on both diffusion and network based evaluations. We thus contend that both types of evaluations should be considered, and support the need for future hybrid approaches that address both.

4.5 CONCLUSION

We have proposed GLIE, a GNN-based solution for the problem of influence estimation. We showcase its accuracy in that task and utilized it to address the problem of influence maximization. We developed three methods based on the representations and the predictions of GLIE. GLIE-CELF, an adaptation of a classical algorithm that surpasses SOTA but with significant computational overhead. GRIM, a Q-learning model that learns to retrieve seeds sequentially using GLIE's predictions and representations. And PUN, a submodular function that acts as proxy for

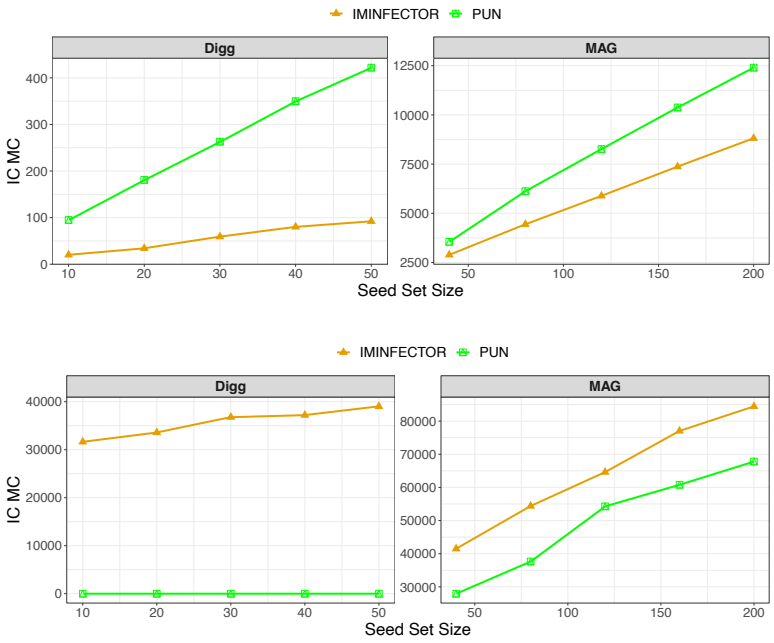


Figure 4.5: PUN vs. IMINFECTOR spreading computed by a) by the independent cascade and b) the distinct nodes influenced in the test set cascades.

the marginal gain and can be optimized adaptively, striking a balance between efficiency and accuracy.

A typical influence maximization algorithm needs a significant contribution in order to take into account the topic of the information shared or the user's characteristics [36] i.e. conditional diffusion. An important practical advantage of a neural network approach is the easy incorporation of such complementary data by adding the corresponding embeddings in the input, as has been done in similar settings [199]. We thus deem an experiment with contextual information a natural next step, given a proper dataset. Our approach can also be utilized to address the minimum vertex cover in large graphs, as it is a problem related to influence maximization and there exists models that work well in both [142]. Finally, we also plan to examine the potential of training online the reinforcement learning, i.e. receiving real feedback from each step of the diffusion that could update both, the Q-NET and GLIE. This would allow the model to adjust its decisions based on the partial feedback received during the diffusion.

Moreover influence maximization methods can be used for battling such large scale manipulation in social media [202], such as targeting the right susceptible users with specific political advertisements might maximize the effect of the campaign on the general intent to vote . There has been extensive research on producing counter campaigns to the fake ones, that are equally effective and hence balancing each other [19]. More importantly, GLIE can be used as part of the mitigation strategy [202], as a black box that substitutes the method's RR-set based influence estimation, to actually increase the balance of political exposure in the given social network for the running campaigns. Moreover, our methods

can inherently be utilized in the context of limiting fake news spreading similar to other influence maximization algorithms [30].

5

TRANSFER GRAPH NEURAL NETWORKS FOR PANDEMIC FORECASTING

The recent outbreak of COVID-19 has affected millions of individuals around the world and has posed a significant challenge to global healthcare. From the early days of the pandemic, it became clear that it is highly contagious and that human mobility contributes significantly to its spread. In this paper, we study the impact of population movement on the spread of COVID-19, and we capitalize on recent advances in the field of representation learning on graphs to capture the underlying dynamics. Specifically, we create a graph where nodes correspond to a country's regions and the edge weights denote human mobility from one region to another. Then, we employ graph neural networks to predict the number of future cases, encoding the underlying diffusion patterns that govern the spread into our learning model. Furthermore, to account for the limited amount of training data, we capitalize on the pandemic's asynchronous outbreaks across countries and use a model-agnostic meta-learning based method to transfer knowledge from one country's model to another's. We compare the proposed approach against simple baselines and more traditional forecasting techniques in 4 European countries. Experimental results demonstrate the superiority of our method, highlighting the usefulness of GNNs in epidemiological prediction. Transfer learning provides the best model, highlighting its potential to improve the accuracy of the predictions in case of secondary waves, if data from past/parallel outbreaks is utilized.

SOURCE CODE The implementation of the proposed model can be found online[1].

5.1 INTRODUCTION

In late 2019, a highly infectious new virus, SARS-CoV-2, started spreading in Wuhan, China. In early 2020, the virus had spread to most countries around the world causing the pandemic of the COVID-19 disease. As of December 7, 2020, a total of $1,532,418$ deaths and $66,422,058$ cases of COVID-19 were confirmed worldwide[2]. In many domains, data admits a natural graph representation. For instance, to predict the spread of COVID-19, ideally, we would like to have access to the social network of all individuals and to make predictions based on the interactions between them. However, in the absence of such data, we consider a similar problem; predicting the development of the disease based on mass mobility data, i.e. how many people moved from one place to another. Mobility inside a region can be regarded as

1 https://github.com/geopanag/pandemic_tgnn
2 https://covid19.who.int

a proxy of the interaction i.e. the more people move, the higher the risk of transmission inside the region. Interestingly, mobility between different regions is known to play a crucial role in the growth of the pandemic, especially for long range travels [46, 190]. Mobility gives rise to a natural graph representation allowing the application of recent relational learning techniques such as graph neural networks (GNNs).

GNNs who have been introduced in Chapter 2 have been applied to a wide variety of tasks, including node classification [107], graph classification [152] and text categorization [160]. Since GNNs have been successfully applied to several real-world problems, in this paper, we also investigate their effectiveness in forecasting COVID-19. We focus on the problem of predicting the number of confirmed COVID-19 cases in each node. We propose a model that captures both spatial and temporal information, thus combining mobility data with the history of COVID-19 cases.

To investigate if the model can learn the underlying complex dynamics associated with COVID-19, we evaluate it on recent data where we predict the number of new cases. Our results demonstrate that GNNs on mobility exhibit a substantial potential in predicting the disease spread. Furthermore, since the availability of data is limited at the start of the outbreak in a country, we employ a transfer learning method based on Model-Agnostic Meta-Learning (MAML) to capitalize on knowledge from other countries' models.

The rest of this chapter is organized as follows. Section 5.2 provides an overview of the related work and elaborates our contribution. Section 5.3 provides an overview of the dataset and the relationship between mobility and COVID-19 cases. Section 5.4 provides a detailed description of the proposed model. Section 5.5 evaluates the proposed model on data from the first COVID-19 wave in 4 EU countries. Finally, section 5.6 summarizes the work and presents potential future work.

5.2 RELATED WORK

As mentioned above, many recent studies have leveraged machine learning and artificial intelligence to make predictions about the spread of COVID-19. For instance, Lorch et al. [133] propose a compartmental SEIR model which is based on a parameterized counting process. The parameteres of the model are estimated using bayesian optimization. and was evaluated on regions of Germany and Switzerland. Flaxman et al. [69] study the effect of major non-pharmaceutical interventions across 11 European countries with a bayesian model whose parameters are estimated based on the observed deaths in those countries. Their results indicate that the interventions have had a large effect on reducing the spread of the disease. Time-series based models have also been utilized and will serve as our baselines. For instance, Chimmula and Zhang [43] employed an LSTM to predict the number of confirmed COVID-19 cases in Canada, while Kufel [111] investigated the effectiveness of the ARIMA model in predicting the dynamics of COVID-19 in certain European countries.

A GNN for epidemic forecasting, ColaGNN, was recently developed by Deng et al. [48]. ColaGNN learns a hidden state for each location using an RNN, and then an attention matrix is derived from these representations that captures how locations influence each other. This matrix forms the graph that is passed on to a GNN to generate the outputs. This work was evaluated on influenza-like illness (ILI) prediction in US and Japan, without the use of an underlying graph. More recent works on predicting COVID-19 using the graph of US counties include a static [103] and a temporal GNN [71]. The former forms a supergraph using the instances of the mobility graph, where the spatial edges capture county-to-county movement at a specific date, and a county is connected to a number of past instances of itself with temporal edges. The node features include demographics, number of deaths and recoveries. In STAN [71] on the other hand, the edges are determined based on demographic similarity and geographical proximity between the counties. STAN takes advantage of the nature of the pandemic and predicts the parameters of an epidemic simulation model together with the infected and recovered cases, using multiple outputs in the neural network. These are used to produce long-term predictions based on the simulation and to penalize the original long-term predictions of the model. The main difference between these approaches and our work lies in the size of the constructed graph and the amount of available data for training. To be specific, in both these approaches the size of the training data ranges from 50 to 60 days. In our case, this is not feasible as the pandemic is already at its peak before the 30^{th} day, and has already cost too many lives. This is why we utilize transfer learning to account for the limited training samples in the initial stages of the pandemic. Moreover, our graphs are relatively small compared to the graph of US counties. Finally, our open data lacks in many cases the number of recovered cases, deaths and population demographics required for training these models. This kind of data may not always be available at the regional level for a real-life pandemic, especially for smaller, less developed countries.

Transfer learning for disease prediction has been used in the past in Zou et al. [242] who have mapped a disease model trained on online google searches obtained from one location, where the virus spreading is available, to another location, where the virus has not spread widely yet. More recently, this approach was utilized in the context of COVID-19 [114]. In the context of graph representation learning, transfer learning has only been used to the best of our knowledge for classifying textual documents represented as graphs [116], for traffic prediction [141], for semi-supervised classification [233] and for designing GNNs that are robust to adversarial attacks [195].

5.3 DATASET

Facebook has released several datasets in the scope of Data For Good program[3] to help researchers better understand the dynamics of the COVID-19 and forecast the spread of the disease [134]. We use a dataset

3 https://dataforgood.fb.com/tools/disease-prevention-maps/

Country	Time	Regions	Avg new case
Italy	24/2-12/5	105	25.65
England	13/3-12/5	129	16.7
Spain	12/3-12/5	35	61
France	10/3-12/5	81	7.5

Table 5.1: Summary of the available data for the 3 considered countries.

that consists of measures of human mobility between administrative NUTS3[4] regions. The data is collected directly from mobile phones that have the Facebook application installed and the Location History setting enabled. The raw data contains three recordings per day (i.e. midnight, morning and afternoon), indicating the number of people moving from one region to another at that point of day. We compute a single value for each day and each pair of regions by aggregating these three values. We focus on 4 European countries: Italy, Spain, France and England.

The number of cases in the different regions of the 4 considered countries were gathered from open data listed in the github page[5] along with the code and the aggregated mobility data. An overview of the preprocessed data can be found in Table 5.1. The start date is the earliest date for which we have both mobility data and data related to the number of cases available. We apply text mining techniques to preprocess and map the regions of the mobility data to those of the open data, as well as quality control for noisy time series where the recordings seemed infeasible. This led us into neglecting a 2 in Italy, and 10 (including islands) in Spain. We also do not take into consideration regions that had less then 10 confirmed cases in total throughout the pandemic, which corresponds to 14 regions in Spain and 3 in Italy, as they were luckily not very affected by the pandemic.

We should stress here that in the considered set of data, case reporting is often not consistent, while there are also very large variations in the number of tests performed in each region/country. This is the main reason behind the large differences in the number of reported cases from day to day, as illustrated in Figure 5.1 for the different regions of Italy and France. Specifically, we see that for almost all regions, the maximum difference encountered between consecutive days is multiple times the average value of the time series, signifying the burstiness and the difficulty of predicting exact samples of such time series.

In order to evaluate the relationship between mobility and COVID-19 cases, we compute the pearson correlation for the examined time shifts in forecasting i.e. ranging from 1 to 14 days ahead. Specifically, for a region u, its mobility is the total number of people moving in and out of it each day, represented by m^t starting from time 0 to time t. The sequence of confirmed COVID-19 cases is represented by c^t, and c^{t+1}

4 https://en.wikipedia.org/wiki/Category:NUTS_3_statistical_regions_of_the_European_Union

5 https://github.com/geopanag/pandemic_tgnn

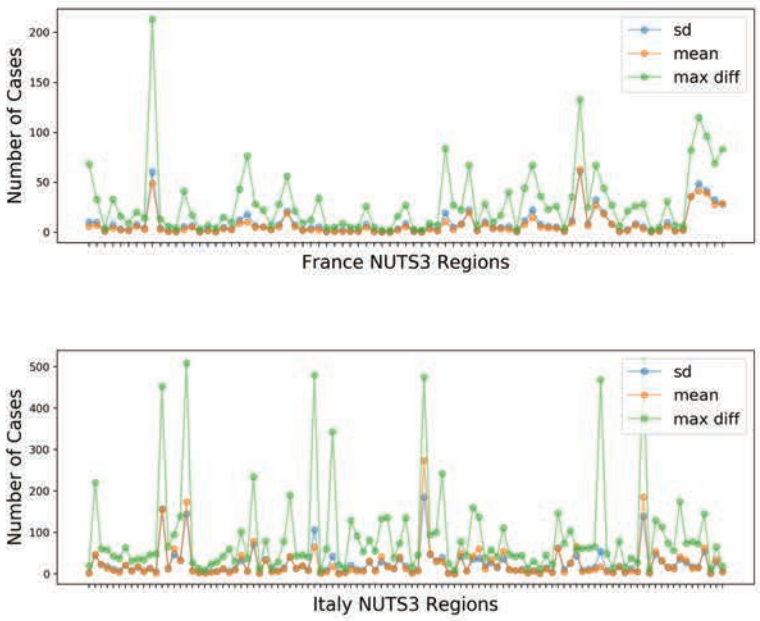


Figure 5.1: Mean, standard deviation and maximum difference of confirmed cases per day.

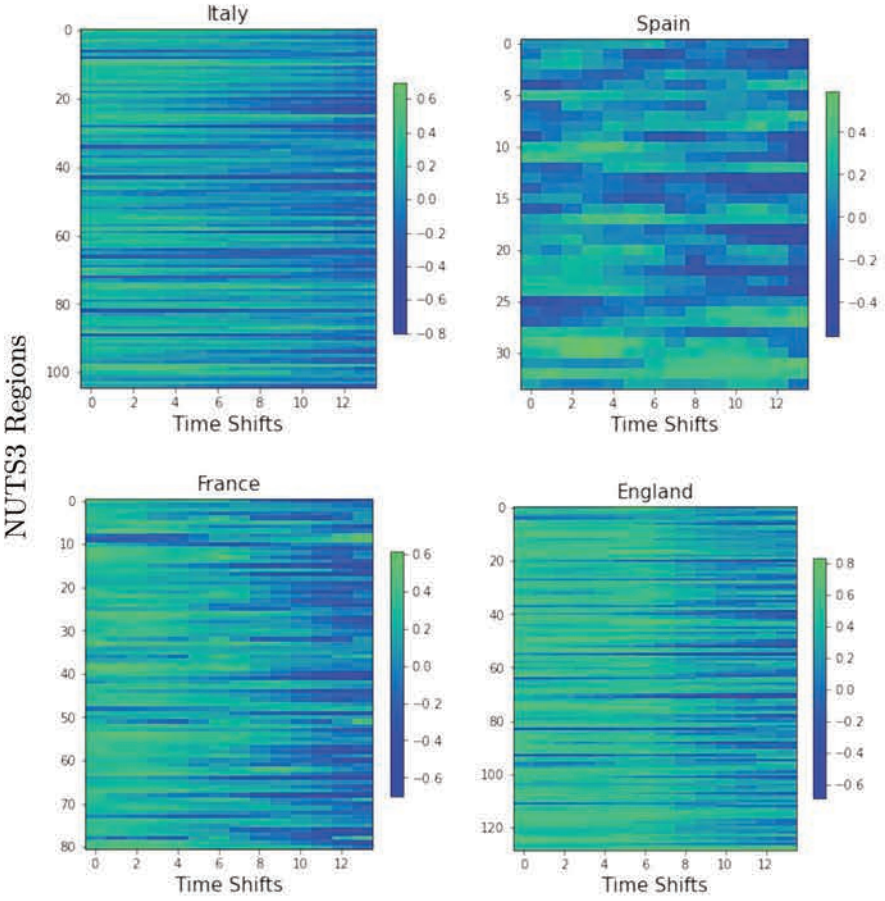


Figure 5.2: Pearson correlation between mobility and number of confirmed cases in the future for examined countries's regions.

represents the vector starting from time 1 to $t + 1$. So for a shift of 1,

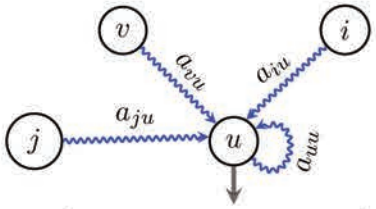
$t = T - 1$ and $Corr^1 = \frac{\sum_i (m_i^t - \mu(m_i^t))(c_i^{t+1} - \mu(c_i^{t+1}))}{\sigma(m^t)\sigma(c^{t+1})}$. In Figure 5.2 we can see the correlation for all regions and all time shifts. Overall, we can see that for most of the regions, mobility is correlated positively with the short term number of cases and vice versa for the long term ones. Overall this pattern pertains throughout most of the regions with the exception of Spain. Hence we expect mobility to be a usefull predictor.

5.4 METHODOLOGY

In this section, we present the proposed neural network architecture for predicting the course of the COVID-19 disease. It should be mentioned that our analysis involves a series of assumptions. First, we assume that people that use Facebook on their mobile phones with Location History enabled constitute a uniform random sample of the general population. Second, we assume that the number of cases in a region reported by the authorities is a representative sample of the number of people that have been actually infected by the virus. Finally, we hypothesize that the more people move from one region to another or within a region, the higher the probability that people in the receiving region are infected by the virus. This is a well-known observation in the field of epidemics[46, 190], and motivates the use of a message-passing procedure as we delineate below.

Symbol	Meaning	Type
N	number of nodes	scalar
$G^{(t)}$	graph at time t	graph
$w_{u,v}^{(t)}$	mobility from region u to v at t	scalar
$\mathbf{x}_u^{(t)}$	number of cases in region u at time t	vector
$\mathbf{H_i^t}$	activations of the hidden layer i at time t	matrix
$\mathbf{W_i}$	learnable parameters of the hidden layer i	matrix
y_u	prediction for region u	scalar
θ	the set of the model's parameters	set
M	the set of datasets	set
$T_{i,j}^k$	the samples for country k starting from day i and predicting j days ahead	set

Table 5.2: Table of symbols.



$$Z_u = (x_j a_{j,u} + x_i a_{i,u} + x_v a_{v,u}) + x_u a_{u,u}$$

Figure 5.3: Example of the message-passing.

5.4.1 *Graph Construction*

We chose to represent each country as a graph $G = (V, E)$ where $N = |V|$ denotes the number of nodes. Specifically, given a country, we create a series of graphs, each corresponding to a specific date t, i.e. $G^{(1)}, \ldots, G^{(T)}$. A single date's mobility data is transformed into a weighted, directed graph whose nodes represent the NUTS3 regions and edges capture the mobility patterns. For instance, the weight $w_{v,u}^{(t)}$ of the edge (v, u) from node v to node u denotes the total number of people that moved from region v to region u at time t. Note that these graphs can also contain self-loops which correspond to the mobility behavior within the regions. The mobility between administrative regions u and v at time t forms an edge which, multiplied by the number of cases $c_u^{(t)}$ of region u at time t, provides a relative score expressing how many infected individuals might have moved from u to v. To be more specific, let $\mathbf{x}_u^{(t)} = (c_u^{(t-d)}, \ldots, c_u^{(t)})^\top \in \mathbb{R}^d$ be a vector of node attributes, which contains the number of cases for each one of the past d days in region u. We use the cases of multiple days instead of just the day before the prediction because case reporting is highly irregular between days, especially in decentralized regions. Intuitively, message-passing over this network computes a feature vector for each region with a combined score from all regions, as illustrated below:

$$\mathbf{A}^{(t)}\,\mathbf{X}^{(t)} = \begin{bmatrix} w_{1,1}^{(t)} & w_{2,1}^{(t)} & \ldots & w_{n,1}^{(t)} \\ w_{1,2}^{(t)} & w_{2,2}^{(t)} & \ldots & w_{n,2}^{(t)} \\ \vdots & \vdots & \vdots & \vdots \\ w_{1,n}^{(t)} & w_{2,n}^{(t)} & \ldots & w_{n,n}^{(t)} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1^{(t)} \\ \mathbf{x}_2^{(t)} \\ \vdots \\ \mathbf{x}_3^{(t)} \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_3 \end{bmatrix}$$

where $\mathbf{A}^{(t)}$ is the adjacency matrix of $G^{(t)}$ and $\mathbf{X}^{(t)}$ is a matrix whose rows contain the attributes of the different regions. In this case, $\mathbf{z}_u \in \mathbb{R}^d$ is a vector that combines the mobility within and towards region u with the number of reported cases both in u and in all the other regions. Here, we would like to stress the importance of the mobility patterns $w_{u,u}$ within a region u which correspond to good indicators of the evolution of the disease, especially during lockdown periods. To visualize concretely how the representations are extracted from the message-passing, Figure 5.3 contains a toy example with a region u receiving a people from different regions, x containing a vector of past cases in that region. $Z_u \in \mathbb{R}^d$ represents an estimate of the number of new latent cases in u,

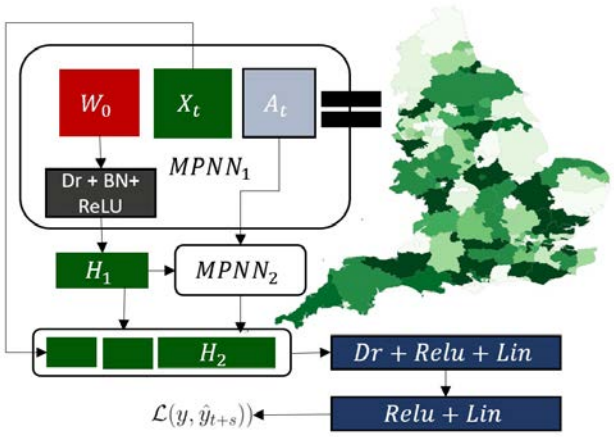


Figure 5.4: Overview of the proposed MPNN architecture.

and is broken down to the cases received from other regions and the new cases caused due to mobility inside u ($x_u a_{uu}$).

5.4.2 *Models*

To model the dynamics of the spreading process, we use two different instances of GNNs [74]. As analyzed in chapter 2, these neural networks consist of a series of neighborhood aggregation layers i.e. message-passing, so we will call them Message Passing Neural Network (MPNN) for the rest of the chapter.

MESSAGE PASSING NEURAL NETWORK To update the representations of the node in each of the input graphs, we use the message-passing scheme of Eq. 2.8 in Chapter 2:

$$\mathbf{H}_{i+1} = f(\tilde{\mathbf{A}}\,\mathbf{H}_i\,\mathbf{W}_{i+1})$$

Note that for simplicity of notation, we have omitted the time index. The above model is applied to all the input graphs $G^{(1)}, \ldots, G^{(T)}$ separately. Given a model with K neighborhood aggregation layers, the matrices $\tilde{\mathbf{A}}$ and $\mathbf{H}_0, \ldots, \mathbf{H}_K$ are specific to a single graph, while the weight matrices $\mathbf{W}_1, \ldots, \mathbf{W}_K$ are shared across all graphs. As the number of neighborhood aggregation layers increases, the final node features capture more and more global information. However, retaining local, intermediary information might be useful as well. Thus, we concatenate the matrices $\mathbf{H}_0, \mathbf{H}_1, \ldots, \mathbf{H}^K$ horizontally, i.e. $\mathbf{H} = \text{CONCAT}(\mathbf{H}_0, \mathbf{H}_1, \ldots, \mathbf{H}_K)$, and the rows of the emerging matrix $\mathbf{H}$ can be regarded as node representations that encode multi-scale structural information, including the initial features of the node. In other words, we utilize skip connections from each layer to the output layer which consists of a sequence of fully-connected layers. Note that we apply the ReLU function to the output of the network since the number of new cases is a nonnegative integer. We choose the mean squared error as our loss function, as shown below:

$$\mathcal{L} = \frac{1}{nT} \sum_{t=1}^{T} \sum_{v \in V} \left(y_v^{(t+1)} - \hat{y}_v^{(t+1)} \right)^2 \tag{5.1}$$

where $y_v^{(t+1)}$ denotes the reported number of cases for region v at day $t+1$ and $\hat{y}_v^{(t+1)}$ denotes the predicted number of cases. An overview of the MPNN is given in Figure 5.4.

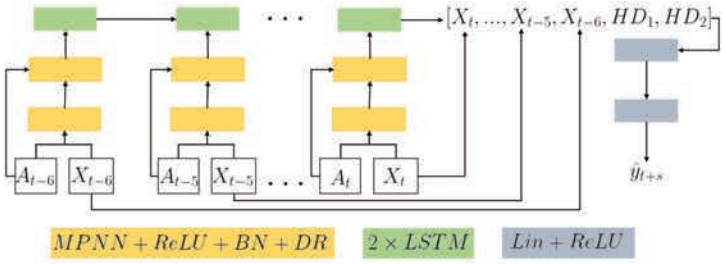


Figure 5.5: Overview of the MPNN LSTM architecture.

MPNN+LSTM. In order to take advantage of the temporal correlation between the target and the confirmed cases in the past, we build a time-series version of our model using the different snapshots of the mobility graph. Given a sequence of graphs $G^{(1)}, G^{(2)}, \ldots, G^{(T)}$ that correspond to a sequence of dates, we utilize an MPNN at each time step, to obtain a sequence of representations $\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \ldots, \mathbf{H}^{(T)}$. These representations are then fed into a Long-Short Term Memory network (LSTM) [82] which can capture the long-range temporal dependencies in time series. We expect the hidden states of an LSTM to capture the spreading dynamics based on the mobility information encoded into the node representations. We use a stack of two LSTM layers. The new representations of the regions correspond to the hidden state of the last time step of the second LSTM layer. These representations are then passed on to an output layer similar to the MPNN, along with the initial features for each time step. Note that this model resembles other attempts to spatio-temporal prediction, but instead of convolutional [232], it employs message-passing layers, similar to [182].

MPNN+TL. Note that the different countries were hit by the pandemic at different times. Indeed, there are cases where once the epidemic starts developing in one country, it has already stabilized in another. Furthermore, a new wave of COVID-19 is very likely to share fundamental characteristics with the previous ones, as it is the same virus. This additional information may prove rather important in case of insufficient training data. In our setting, as discussed in Subsection 5.5.1, the model starts predicting as early as the 15^{th} day of the dataset. In such a scenario, the model has access to a few samples to learn from (split into validation and training sets), while it is used to make predictions for as far as 14 days ahead. Given the inherent need for data in neural networks, this setting is rather challenging. Moreover, our intuition is that a model trained in the whole cycle or an advanced stage of the epidemic can capture patterns of its different phases, which is missing from a new model working in a country at the start of its infection.

To incorporate past knowledge from models running in other countries, we separate our data into tasks and propose an adaptation of MAML [68], which was introduced in Chapter 2. Assuming that the Meta Train

set $M_{\text{tr}} = \{D^{(1)}, \ldots, D^{(p)}\}$, corresponds to the data sets of p countries that we can use to obtain a set of parameters θ. The learnt parameters can then be employed to initialize the model for the country left out in the Meta Test M_{te}. In reality, each dataset $D^{(k)}, k \in \{1, \ldots, p\}$ is divided into subtasks itself. More specifically, each country has different training sets of increasing size (as the train days increase) as well as shorter- and longer-term targets (next day, two days ahead and so on). For each combination of these two, we train a different model. Hence, the set of tasks for a country k is $D^{(k)} = \left\{ \left(Tr_{i,j}^{(k)}, Te_{i,j}^{(k)} \right) : 14 \geq i \geq T_{\max}, 1 \geq j \geq dt \right\}$ where $\left(Tr_{i,j}^{(k)}, Te_{i,j}^{(k)} \right)$ is a dataset (train and test set) associated with country k where the train set comprises of the first i days of the data and the task is to predict the number of cases in the j-th day ahead.

The set of parameters θ corresponds to the weight matrices and biases of all layers in the MPNN model. As mentioned above, in MAML, θ is randomly initialized and underoges gradient descent steps during the metatrain phase. The algorithm is shown in Algorithm 7.

Algorithm 7 MPNN+TL

Input: M_{tr}, M_{te}, α , α_m, n_epochs
Output: θ

1: Initialize θ randomly
2: **for** $D \in M_{\text{tr}}$ **do**
3: **for** $(Tr, Te) \in D$ **do**
4: **for** Batch $b \in Tr$ **do**
5: $\theta_t = \theta - \alpha \nabla_\theta \mathcal{L}(f_\theta(b))$
6: $\theta = \theta - \alpha_m \nabla_\theta \mathcal{L}(f_{\theta_t}(Te))/|M_{\text{tr}}|$
7: **for** $(Tr, Te) \in M_{\text{te}}$ **do**
8: **for** Epoch $e \in n\_epochs$ **do**
9: **for** Batch $b \in Tr$ **do**
10: $\theta = \theta - \alpha \nabla_\theta \mathcal{L}(f_\theta(b))$
11: $error+ = E(f_\theta(Te))$
12: **return** $error/|M_{\text{te}}|$

Note that in the Algorithm 7 , E is our error function (i.e. Eq. (5.5)), $\mathcal{L}$ is the loss function defined in Eq. (5.1), and f is an MPNN.

In each task, we minimize the loss on the task's train set towards a task-specific θ_t, as shown in Eq. (5.2) and on lines 3-5 of the algorithm. Then, we use the emerging θ_t to compute the gradient with respect to θ in the task's test set as illustrated in Eq. (5.3) below and on line 6. This gradient is normalized by the total number of tasks in the set to refrain from taking too big steps:

$$\theta_t = \theta - \alpha \nabla_\theta \mathcal{L}\left(f_\theta(Tr_{i,j}^{(k)}) \right) \tag{5.2}$$

$$\theta = \theta - \alpha_m \nabla_\theta \mathcal{L}\left(f_{\theta_t}(Te_{i,j}^{(k)}) \right) \tag{5.3}$$

The standard update includes the gradient of θ_t and that of θ, which is in fact the hessian matrix, as shown in Eq. (5.4).

$$\frac{\partial \mathcal{L}_T(f_{\theta_t}(Te_{i,j}^{(k)}))}{\partial \theta} = \nabla_{\theta_t}\mathcal{L}(f_{\theta_t}(Te_{i,j}^{(k)}))(I - a\nabla_{\theta}^2\mathcal{L}f_{\theta}(Tr_{i,j}^{(k)})) \quad (5.4)$$

We are dropping the term that contains the hessian, as it was shown to have insignificant contribution in practice [68], possibly due to the vanishing gradient. Finally, we train θ on the train set of M_{te} and test on its test set (lines 7-10 and 11 respectively).

5.5 EXPERIMENTS

In this section, we first describe the experimental setting and the baselines used for comparison. We last report on the performance of the proposed models and the baselines.

5.5.1 *Experimental setup*

In our experiments, we train the models using data from day 1 to day T, and then use the model to predict the number of cases for each one of the next dt days (i.e. from day $T+1$ to day $T+dt$). We are interested in evaluating the effectiveness of the model in short-, mid- and long-term predictions. Therefore, we set dt equal to 14. We expect the short-term predictions (i.e. small values of dt) to be more accurate than the long-term predictions (i.e. large values of dt). Note that we train a different model to predict the number of cases for days $T+i$ and $T+j$ where $i, j > 0$ and $i \neq j$. Therefore, each model focuses on predicting the number of cases after a fixed time horizon, ranging from 1 day to 14 days. With regards to the value of T, it is initially set equal to 14 and is gradually increased (one day at a time). Therefore, the size of the training set increases as time progresses. Note that a different model is trained for each value of T. Furthermore, for each value of T, to identify the best model, we build a validation set which contains the samples corresponding to days $T-1$, $T-3$, $T-5$, $T-7$ and $T-9$, such that the training and validation sets have no overlap with the test set.

With regards to the hyperparameters of the MPNN, we train the models for a maximum of 500 epochs with early stopping after 50 epochs of patience. Early stopping starts to occur from the 100^{th} epoch and onward. We set the batch size to 8. We use the Adam optimizer with a learning rate of 10^{-3}. We set the number of hidden units of the neighborhood aggregation layers to 64. Batch normalization and dropout are applied to the output of every neighborhood aggregation layer, with a dropout ratio of 0.5. We store the model that achieved the highest validation accuracy in the disk and then retrieve it to make predictions about the test samples. For the MPNN+LSTM model, the dimensionality of the hidden states of the LSTMs is set equal to 64. All the models are implemented with Pytorch [168]. We evaluate the performance of a model by comparing the predicted total number of

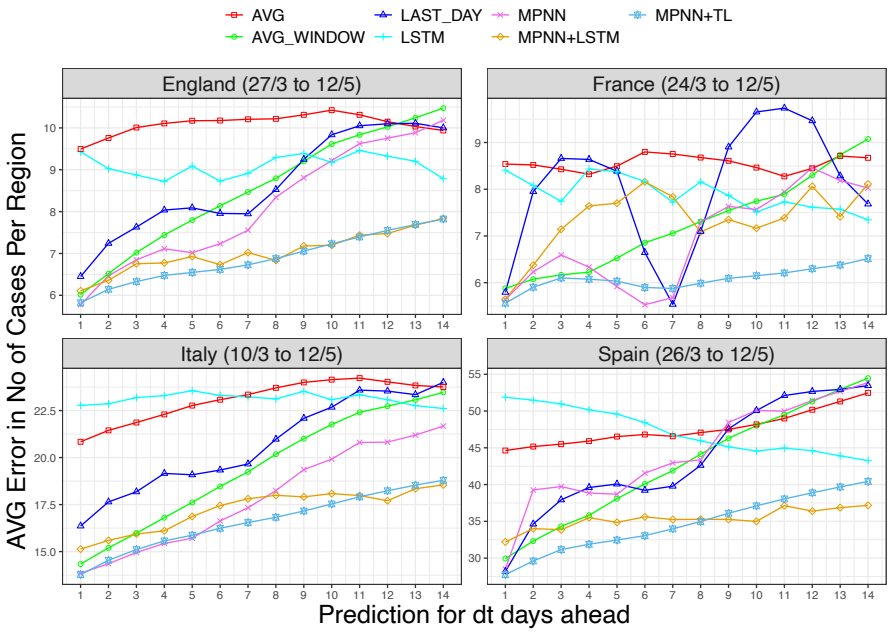


Figure 5.6: Average number of cases lost per region for each target shift. PROPHET and ARIMA are omitted and shown in the table, because they effected the legibility of the plot.

Model	Up to next 3 Days				Up to next 7 Days				Up to next 14 Days			
	England	France	Italy	Spain	England	France	Italy	Spain	England	France	Italy	Spain
AVG	9.75	8.50	21.38	45.10	9.99	8.55	22.23	45.87	10.09	8.55	23.09	47.63
LAST_DAY	7.11	7.47	17.40	33.58	7.62	7.37	18.49	37.06	8.66	8.03	20.69	43.63
AVG_WINDOW	6.52	6.04	15.17	32.19	7.34	6.40	16.81	36.06	8.54	7.24	19.45	42.79
LSTM	9.11	8.08	22.94	51.44	8.97	8.13	23.17	49.89	9.10	7.91	23.12	47.26
ARIMA	13.77	10.72	35.28	40.49	14.55	10.53	37.23	41.64	15.65	10.91	39.65	46.22
PROPHET	10.58	10.34	24.86	54.76	12.25	11.56	27.39	62.16	16.24	14.61	33.07	79.42
TL_BASE	9.65	7.67	19.12	42.25	12.30	9.21	23.44	52.29	13.48	12.27	24.89	59.68
MPNN	6.36	6.16	14.39	35.83	6.86	5.99	15.47	38.51	8.13	6.93	17.88	44.25
MPNN+LSTM	6.41	6.39	15.56	33.35	6.67	7.21	16.41	34.47	7.02	7.36	17.25	35.31
MPNN+TL	**6.05**	**5.83**	**14.08**	**29.61**	**6.33**	**5.90**	**14.61**	**31.55**	**6.84**	**6.13**	**16.69**	**34.65**

Table 5.3: Average error for $dt = 1-3, 1-7$ and $1-14$, in number of cases per region.

cases in each region versus the corresponding ground truth, throughout the test set:

$$\text{error} = \frac{1}{n\,dt} \sum_{t=T+1}^{T+dt} \sum_{v \in V} |\hat{y}_v^{(t)} - y_v^{(t)}| \tag{5.5}$$

5.5.2 *Baselines*

We compare the proposed models against the following baselines and benchmark methods, which have been applied to the problem of COVID-19 forecasting:

- AVG: The average number of cases for the specific region up to the time of the test day.
- AVG_WINDOW: The average number of cases in the past d for the specific region where d is the size of the window.

Country	Difference	Total Cases
England	10,088	99,087
France	4,004	29,767
Italy	24,872	169,674
Spain	13,176	100,345

Table 5.4: Cumulative difference in terms of correctly identified cases between the proposed method and the next best baseline.

- LAST_DAY: The number of cases in the previous days is the prediction for the next days.
- LSTM [43]: A two-layer LSTM that takes as input the sequence of new cases in a region for the previous week.
- ARIMA [111]: A simple autoregressive moving average model where the input is the whole time-series of the region up to before the testing day.
- PROPHET [135]: A forecasting model for various types of time series[6].The input is similar to ARIMA.
- **TL_BASE**: An MPNN that is trained on all data from the three countries and the train set of the fourth (concatenated), and tested on the test set of the fourth. This serves as a baseline to quantify the usefulness of MPNN+TL.

We should note here that since we rely solely on the number of confirmed cases, we can not utilize models that work with recovery, deaths and policies, such as SEIR. That said, a simple approach is to run SI at every given T with a parameter β taken from the COVID-19 literature, along with the number of infected people at T and the population. In some preliminary experiments, however, this provided errors in a different scale then the ones mentioned here, similar to Gao et al. [71], which is why we have not experimented further.

5.5.3 *Results and Discussion*

The average error per region for each one of the next 14 days is illustrated in Figure 5.6. We observe that in all cases, the proposed models yield lower average errors compared to those of the baselines. Among the three variants, MPNN+TL is the best-performing model. It initially outperforms the other approaches by a small margin that increases further after the second day. Even simple baselines can be competitive at predicting the next day's number of cases since proximal samples for the same region from the same phases of the pandemic tend to have a similar number of cases. However, a prediction that goes deeper in time requires the identification of more persistent patterns. In the case of our model, as mentioned above, we aim to capture unregistered cases moving

6 https://github.com/facebook/prophet

from one region to the other or spreading the disease in their new region. These cases would inevitably take a few days to appear, due to the delay of symptoms associated with COVID-19. This is why MPNN performs well throughout the 14-days window. The results also demonstrate the benefit of transfer learning techniques since MPNN+TL outperforms MPNN and its baseline **TL_BASE** in all cases. We expect MPNN to perform similar towards the end of the dataset, when the training of both models has become similar due to the number of epochs and. The main difference occurs when T is small, where the training samples are scarce and MPNN is unable to capture the underlying dynamics. One way to see this is again the accuracy of MPNN+TL in the long term predictions. Due to the size of the prediction window, long term tasks have diminished train set, meaning if the task is to predict t+14 and the set ends at day 60, t+14 training will stop at day 46 while the t+1 will stop at 59. Thus MPNN performs similarly at the short-term predictions but fails compared to MPNN+TL in the long term.

Note that for clarity of illustration, we chose not to visualize the performance of PROPHET and ARIMA in Figure 5.6 as their error was distorting the plot. However, we present in Table 6.2 the average error for the predictions where dt takes three values: 3, 7 and 14. Overall, it is clear that the time series methods (i.e. LSTM, PROPHET, and ARIMA) and the temporal variant of our method MPNN+LSTM yield quite inaccurate predictions. Apart from the inherent difficulty of learning with time-series data, which we analyze further below, this might also happen because of the nature of the epidemic curve. Specifically, sequential models, that are trained with values that tend to increase, are impossible to predict decreasing or stable values. For instance, in our dataset, once the models have enough samples to learn from, there is a transition in the phase of the epidemic due to lockdown measures. The same applies when the epidemic starts to recede at the start of May.

Our error function treats all regions equivalently, independent of the region-specific population and the number of cases, i.e. a region with 10 cases per day should not be treated the same as a region with 1000. We need a measure to take into account the region-specific characteristics as well as the time. Towards this end, we computed the deviation between the average predicted and the actual average number of cases over the next 5 days (not to be confused with the average error for the next x days in Table 5.3) and 5.4). The relative error of each region is defined below and is illustrated in Figure 5.7.

$$r = \frac{1}{n\,(T-5)} \sum_{t=1}^{T-5} \sum_{v \in V} \frac{\left| \sum_{i=0}^{5} \hat{y}_v^{(t+i)} - \sum_{i=0}^{5} y_v^{(t+i)} \right|}{\sum_{i=0}^{5} y_v^{(t+i)}} \tag{5.6}$$

One can see that the regions with high relative error are the ones with the fewest cases. On the other hand, the regions with the highest number of cases tend to have much smaller relative error, less than 20% to be exact, with the exception of one region in Spain. This indicates that our model indeed produces accurate predictions that could be useful in resource allocation and policy-making during the pandemic.

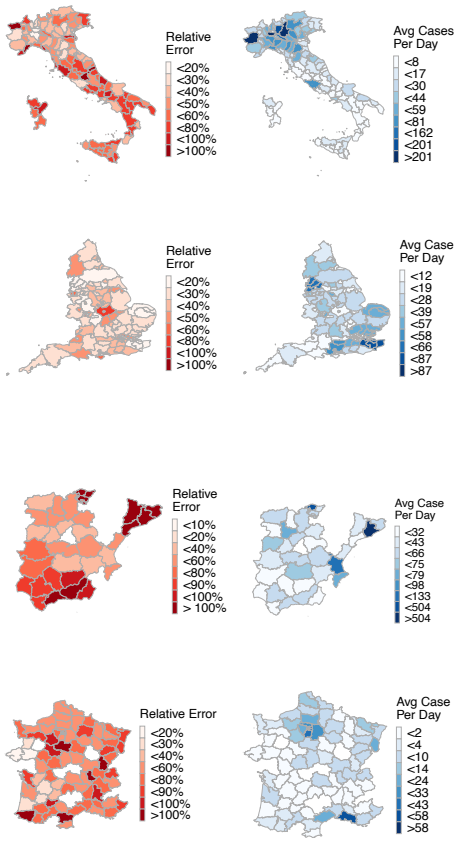


Figure 5.7: Plot of the relative test error and average number of cases per day for each available region.

Figure 5.7 also allows us to evaluate the method more objectively. From a machine learning perspective, one may argue that even though the MPNN+TL outperforms the baselines, their predictions are not very accurate in terms of average error. This is partially explained due to the inherent problems of the dataset mentioned at Section 5.3 as well as the assumption that case reporting is standard throughout the regions. We expect a large improvement in performance in case a standard methodology for case reporting is adopted and the number of tests per region remains constant and proportional to the population. Having said that, utilizing such a model in practice is more than feasible, as the difference in scale is more useful at the regional level. In other words, a region predicted to have 200 new cases total in the next 5 days will have similar needs with a region with 240 or 160 real cases (20% error). Contrary to that, a prediction of 200 cases with a real value of 300 would be a more significant misclassification, which is not possible looking at the results at Figure 5.7. Moreover, regions with big relative error tend to have a small number of cases e.g. the model may predict 20 cases in a region with 10 (100% error), which is also acceptable from a real-world perspective.

5.6 CONCLUSION

In this paper, we presented a model for COVID-19 forecasting which could provide useful insights to policymakers and allow them to make informed decisions on appropriate interventions and resource allocation. The proposed model builds upon the recent work on GNNs. We use mobility data as a graph where nodes correspond to regions and edge weights to measures of human mobility between their endpoints. Then, we derive variants of the family of MPNNs to generate representations for the regions based on their interactions. Furthermore, since different countries might be in different phases of the epidemic, we propose to transfer a well-performing disease spreading model from one country to another where limited data is available. Experiments conducted on data from 4 European countries show that our architectures outperform traditional and more recent approaches in predicting the number of daily new COVID-19 cases.

In terms of future directions, we plan to provide our model with e. g.demographics regarding the the age/gender distribution and features related to the weather. Furthermore, we will include additional data from Facebook such as the intensity of connectedness between regions measured by the friendship relationships between two regions. Our final goal is to evaluate the model on the second wave of COVID-19, based on the first.

6

ANALYSIS OF ACADEMIC INFLUENCE

Measuring and evaluating an author's influence or her overall impact over the academic world has been a withstanding challenge with profound effects on society. Apart from its practical usage for academic evaluation from governmental and industrial organizations, it enhances scientific transparency and reinforces excellence. Moreover, due to the democratization of science through social media, the public and mass media require concise and reliable evaluations to follow a scientist's opinion in pressing matters, since they lack the expertise to assess the validity of an authority's opinion. In this chapter, we present our efforts to address this problem capitalizing on different types of academic networks built based on the Microsoft Academic Graph (MAG), to our knowledge the largest heterogeneous database of scientometric measurements. We first attempt to predict the author's scientific impact using graph neural networks on the coauthorship network with node representations extracted from the author's papers' abstracts. The proposed method is compared with standard data mining techniques based on graph mining and machine learning models. Subsequently, in an effort to provide more qualitative results, we refrain from machine learning and take a more traditional approach to build an interactive web application based on network science and large-scale analytics. We first create the citation network between authors i.e. who-cites-whom, and then separate the authors impact into two different scientometrics: success in the specific field and influence over the academic network The first is the distribution of the h-index for specific scientific fields and a search engine to visualize an authors' position in it as well as the top percentile she belongs to. The second is recomputing the D-core influence metric on this huge network and presenting authority/integration of the authors in the form of D-core frontiers. As a means of evaluation, we present interesting insights on the densest scientific domains and the most influential authors. We believe the proposed analytics highlight under-examined aspects in the area of scientific evaluation and pave the way for more involved scientometrics.

SOURCE CODE The implementation of the proposed model can be found online [1].

6.1 MICROSOFT ACADEMIC GRAPH

Microsoft Academic Graph (MAG) is the biggest openly available bibliographical corpus, including more than 250 million authors and 219 million papers [188]. We employed a snapshot version[2] provided by

1 https://github.com/geopanag/MAG-Author-Influence-Scientometrics
2 created at 9/2/2019

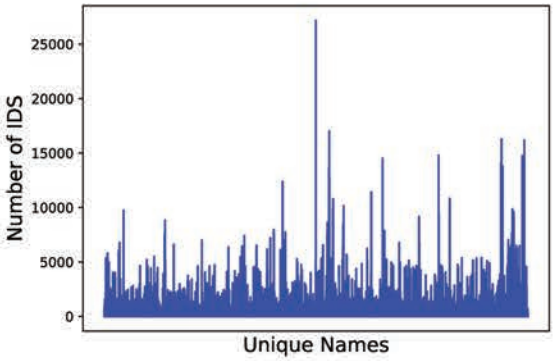


Figure 6.1: Number of ids for the ambiguous names in MAG.

Microsoft Academic Services, but before that, we have examined thoroughly other options such as DBLP, Aminer, and the openly available MAG through the Open Academic Graph [194]. The reason we decided to go with the official version of MAG is i) MAG h-index estimates were closer to other services like Google Scholar and Scopus, ii) open data were missing conference names, scientific fields, etc. That said, MAG required significant preprocessing before performing the analysis.

As a first step, we removed names that do not include English characters, which diminishes the initial 83 to about 63 million. Afterwards, we deal with the issue of multiple IDs referring to the same name. This problem refers to the name ambiguities created by noisy registers in the dataset, such as 'A. H. Tang', 'Arthur Tang', 'A. Tang' etc. which refer to the same person but are interpreted as different[239]. In our case, a name is ambiguous if it has more than 1 IDs, which is very common. To be more specific, out of the 63 million names, roughly one-third $(23,427,411)$ has more than one IDs, amounting to $176,631,328$ ambiguous ID assignments. One can see in Figure 6.1 that the number of possible IDs for the same ambiguous name can reach more than $27,000$. Roughly $3,500$ names had more than $1,000$ IDs each, revealing the scale of the problem.

More than 80% $142,851,013$ of these ambiguous IDs share a peculiar characteristic: they have only one paper. For example, a name that has 27,169 author IDs from Figure 6.1, has $27,126$ IDs with only one paper each. We can hypothesize that these IDs were created specifically for each paper. These IDs are not useful for our purpose since by default they will have an h-index of 1. These 142 million IDs are assigned to $22,503,619$ ambiguous names, which is more than 95% of all ambiguous names. Since we lack an efficient way to perform name disambiguation at this scale, we remove them from the analysis, ending up with 40 million IDs. Regarding the paper's abstracts and titles, we remove stopwords from the text [132].

In the following sections, we describe how we utilized MAG in two different applications pertaining to ranking scientists. Section 6.2 describes a machine learning pipeline that combines textual data from papers with coauthorships exhibiting influence relationships, to regress the h-index of a scientist. Section 6.3 provides a set of plots for the h-index over different scientific fields and a visualization of an author's

influence based on graph degeneracy. Finally, Section 6.4 concludes with a summary and potential future steps.

6.2 GRAPH NEURAL NETWORKS FOR h-INDEX PREDICTION

Citation counts is undoubtedly the most widely used indicator of a paper's impact and success. It is commonly believed that the more citations a paper has received, the higher its impact. When it comes to authors, measuring impact becomes more complicated since an author may have published multiple papers in different journals or conferences. Still the publication record of an author is in many cases the most important criterion for hiring and promotion decisions, and for awarding grants. Therefore, institutes and administrators are often in need of quantitative metrics that provide an objective summary of the impact of an author's publications. Such indicators have been widely studied in the past years [25, 59, 223, 235]. However, it turns out that not all aspects of an author's scientific contributions can be naturally captured by such single-dimensional measures. The most commonly used indicator is perhaps the h-index, a measure that was proposed by Jorge Hirsch [91]. The h-index measures both the productivity (i.e. the number of publications) and the impact of the work of a researcher. Formally, the h-index is defined as the maximum value of h such that the given author has published h papers that have each been cited at least h times. Since its inception in 2005, the h-index has attracted significant attention in the field of bibliometrics. It is not thus surprising that all major bibliographic databases such as Scopus, the Web of Science and Google Scholar compute and report the h-index of authors. One of the main appealing properties of the h-index is that it is easy to compute, however, the indicator also suffers from several limitations which have been identified in the previous years [23, 176, 211].

It is clear from the definition of h-index that it mainly depends on the publication record of the author. On the other hand, there is no theoretical link between the h-index of an author and the collaborations the author has formed. For instance, it is not necessary that the h-index of an author that has collaborated with many other individuals is high. However, the above example is quite reasonable, and empirically, there could be some relation between h-index and co-authorship patterns [105]. In fact, it has been reported that co-authorship leads to increased scientific productivity (i.e. number of papers published) [6, 55, 95, 117, 244]. For instance, some studies have investigated the relationship between productivity and the structural role of authors in the co-authorship network, and have reported that authors who publish with many different co-authors bridge communication and tend to publish larger amounts of papers [58, 110]. Since research productivity is captured by the h-index, co-authorship networks could potentially provide some insight into the impact of authors. Some previous works have studied if centrality measures of nodes are related to the impact of the corresponding authors. Results obtained from statistical analysis or from applying simple machine learning models have shown that in

some cases, the impact of an author is indeed related to their structural role in the co-authorship network [1, 21, 228].

The goal of this section is to uncover such connections between the two concepts, i.e. h-index and the co-authorship patterns of an author. Specifically, we leverage machine learning and study whether the collaboration patterns of an author are good indicators of the author's success i.e. the h-index. We treat the task as a regression problem and since the collaboration patterns of an author can be represented as a graph, we capitalize on recent advances in graph neural networks, presented in Chapter 2. We build a co-authorship network of the aforementioned processed MAG dataset, where nodes correspond to authors that have published papers in computer science journals and conferences. Their top-cited papers are utilized to create node features using text representations. We then train a GNN model to predict the h-index of each author. Our results demonstrate that the structural role of an individual in the co-authorship network is indeed empirically related to their h-index. Furthermore, we find that local features extracted from the neighborhood of an author are very useful for predicting the author's h-index. On the other hand, the textual features generated from the author's published papers do not provide much information. Overall, the proposed architectures quantify the potential gain of using GNNs overal traditional methods.

6.2.1 *Related Work*

In the past years, a substantial amount of research has focused on determining whether information extracted from the co-authorship networks could serve as a good predictor of the authors' scientific performance. For instance, McCarty *et al.* investigated in [145] which features extracted from the co-authorship network enhance most the h-index of an author. The authors found that the number of co-authors and some features associated with highly productive co-authors are most related to the increase in the h-index of an author. In another study [88], Heiberger and Wieczorek examined whether the structural role of researchers in co-authorship networks is related to scientific success, measured by the probability of getting a paper published in a high-impact journal. They found that the maintenance of a moderate number of persistent ties, i.e. ties that last at least two consecutive years, can lead to scientific success. Furthermore, they report that authors who connect otherwise unconnected parts of the network are very likely to publish papers in high-impact journals. Parish *et al.* employed in [167] the R index, an indicator that captures the co-authorship patterns of individual researchers, and studied the association between R and the h-index. They found that more collaborative researchers (i.e. lower values of R) tend to have higher values of h-index, and the effect is stronger in certain fields compared to others.

Several previous works have studied the relationship between network centrality and research impact. Yan and Ding investigated in [228] if four centrality measures (i.e. closeness centrality, betweenness centrality,

degree centrality, and PageRank) for authors in the co-authorship network are related to the author's scientific performance. They found that all four centrality measures are significantly correlated with citation counts. Abbasi *et al.* also studied the same problem in [1]. Results obtained from a Poisson regression model suggest that the g-index (an alternative of the h-index) of authors is positively correlated with four of the considered centrality measures (i.e. normalized degree centrality, normalized eigenvector centrality, average ties strength and efficiency). Sarigöl *et al.* examined in [180] if the success of a paper depends on the centralities of its authors in the co-authorship network. The authors showed that if a paper is represented as a vector where most of the features correspond to centralities of its authors, then a Random Forest classifier that takes these vectors as input can accurately predict the citation success of the paper. Bordons *et al.* explored in [21] the relationship between the scientific performance of authors and their structural role in co-authorship networks. The authors utilized a Poisson regression model to explore how authors' g-index is related to different features extracted from the co-authorship networks. The authors find that the degree centrality and the strength of links show a positive relationship with the g-index in all three considered fields. Centrality measures cannot only predict scientific success, but can also capture the publication history of researchers. More specifically, Servia-Rodríguez *et al.* studied in [183] if the research performance of an author is related to the number of their collaborations, and they found that if two authors have similar centralities in the co-authorship network, it is likely that they also share similar publication patterns.

Some other related studies have investigated the relationship between the number of authors and the success of a paper. For instance, Figg *et al.* analyzed in [67] the relationship between collaboration and the citation rate of a paper. The authors found that there exists a positive correlation between the number of authors and the number of citations received by a paper, for papers published in six leading journals. Hsu and Huang studied in [94] whether collaboration leads to higher scientific impact. They computed the correlation between the number of citations and the number of co-authors for papers published in eight different journals, and drew the same conclusions as Figg *et al.* above. Within each journal, there exists a positive correlation between citations and the number of co-authors, while single-authored articles receive the smallest number of citations. In a different study [3], Abramo *et al.* examined how the international collaborations of Italian university researchers is associated with their productivity and scientific performance. The authors found that the volume of international collaboration is positively correlated with productivity, while the correlation between the volume of international collaboration and the average quality of performed research is not very strong.

Our work is also related to mining and learning tasks in academic data such as collaboration prediction [85, 229] and prediction of scientific impact and success [4, 218]. When the scientific impact is measured by the author's h-index, the learning task boils down to predicting the author's h-index [51], while a similar problem seeks to answer the

question of whether a specific paper will increase the primary author's h-index [50]. In addition, some attempts have been made to predict the impact of an author based on its early indications and the use of graph mining techniques [166].

Finally, GNNs, as learning models, have also been applied to other types of graphs extracted from bibliographic data. More specifically, some of the most common benchmark datasets to evaluate GNN methods are the following paper citation networks: Cora, PubMed and CiteSeer [181]. These datasets are used to predict the field of the paper given the network and the paper's textual information (i.e. abstract), in small scale. In a larger scale, a recent graph transformer has been developed to perform link prediction on a heterogeneous network that includes papers, textual information extracted from papers, venues, authors, citations and fields [96]. The edges represent different types of relations and thus an attention mechanism is utilized to separate between them and aggregate the representations of the nodes. The final tasks evaluated are paper-field, paper-venue and author-paper matching which correspond to link prediction tasks.

6.2.2 *Methods*

Symbol	Meaning	Type
N	number of nodes	scalar
$N(u)$	neighbors of node u	set
P_u	papers of node u	set
$\mathbf{h}_u$	feature representation of node u	vector
$\mathbf{A}$	adjacency matrix	matrix
$\mathbf{W}^t$	parameters for the t layer	matrix
$\mathbf{H}^t$	activations for the t layer	matrix
$r_{u,f}$	relevance for a field f to an author u	scalar

Table 6.1: Table of symbols.

GRAPH METRICS One of the most prominent ways to estimate the scientific impact of an author is based on the author's position in the academic collaboration network [166]. The position can be estimated through multiple network science centralities and metrics developed to capture different dimensions of a node's impact on the graph. In this work, we utilize a number of them in order to serve as the input features for our model.

- **Degree**: Defined in Chapter 2.
- **Degree centrality**: The number of neighbors of the node divided by the maximum possible number of neighbors (i.e. $n-1$).

- **Neighbor's average degree**: The average degree of the neighborhood of a node:

$$ND(v) = \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} deg(u) \tag{6.1}$$

- **PageRank**: Pagerank is an algorithm that computes a ranking of the nodes in a graph based on the structure of the incoming egdes. The main idea behind the algorithm is that a node spreads its importance to all nodes it links to:

$$PR(v) = \sum_{u \in \mathcal{N}(v)} \frac{PR(u)}{deg(u)} \tag{6.2}$$

  In practice, the algorithm computes a weighted sum of two matrices, i.e. the column stochastic adjacency matrix and an all-ones matrix. Than, the pagerank scores are contained in the eigenvector associated with the first eigenvalue of that matrix [161].

- **Core number**: Defined in Chapter 2.

- **Diversity coefficient**: The diversity coefficient is a centrality measure based on the Shannon entropy. Given the probability of selecting a node's neighbor based on its edge weight $p_{u,v} = \frac{w_{v,u}}{\sum_{u \in N(v)} w_{v,u}}$, the diversity of a node is defined as the (scaled) Shannon entropy of the weights of its incident edges.

$$D(v) = \frac{-\sum_{u \in \mathcal{N}(v)} (p_{v,u} \log(p_{v,u}))}{\log(|\mathcal{N}(v)|)} \tag{6.3}$$

- **Community-based centrality**: This centrality measure calculates the importance of a node by considering its edges towards the different communities [203]. Let $d_{v,c}$ be the number of edges between node v and community c and n_c the size of community c retrieved by modularity optimization. The metric is then defined as follows:

$$CB_v = \sum_{c \in C} d_{v,c} \frac{n_c}{n} \tag{6.4}$$

- **Community-based mediator**: The mediator centrality takes into consideration the role of the node in connecting different communities [203], where the communities are again computed by maximizing modularity. The metric relies on the percentages of the node's weighted edges that lie in its community relative to its weighted degree and the corresponding percentage for edges on different communities. To compute it, we first calculate the internal density of node v as follows:

$$p_v^{c_v} = \frac{\sum_{u \in \mathcal{N}(v) \cup c_v} w_{v,u}}{deg(v)} \tag{6.5}$$

where c_v is the community to which v belongs and can be replaced with the other communities to obtain the respective external densities. Given all densities, we can calculate the entropy of a node as follows:

$$O_v = -p_v^{c_v} \log(p_v^{c_v}) - \sum_{c' \in C \setminus c_v} p_v^{c'} \log(p_v^{c'}) \tag{6.6}$$

where C is the set that contains all the communities. Finally, we compute the community mediator centrality:

$$CM_v = O_v \frac{deg(v)}{\sum_{u \in \mathcal{N}(v)} deg(u)} \tag{6.7}$$

TEXT REPRESENTATIONS In the past years, representation learning approaches have been applied heavily in the field of natural language processing. The Skip-Gram model [148] is one of the most well-established methods for generating distributed representations of words. Skip-Gram is a neural network comprising of one hidden layer and an output layer, and can be trained on large unlabeled datasets, similar to the node representation learning models described in Chapter 2 and INF2VEC described in Chapter 3. In our setting, let P_v denotes a set that contains the abstracts of some papers published by author v. Let also W denote the vocabulary of all the abstracts contained in $\bigcup_{v \in V} P_v$. Than, to learn an embedding for each word $w \in W$, our model is trained to minimize the following objective function:

$$\mathcal{L}(P, W) = \sum_{d \in P_v} \sum_{w_i \in d} \sum_{\substack{w_j \in \{w_{i-c}, \ldots, w_{i+c}\} \\ w_j \neq w_i}} \log\Big(p(w_j|w_i)\Big) \tag{6.8}$$

$$p(w_j|w_i) = \frac{\exp(\mathbf{v}_{w_i}^\top \mathbf{v}'_{w_j})}{\sum_{w \in W} \exp(\mathbf{v}_{w_i}^\top \mathbf{v}'_w)} \tag{6.9}$$

where c is the training context, $\mathbf{v}_{w_i}^\top$ is the row of matrix $\mathbf{H}$ that corresponds to word w_i, and $\mathbf{v}'_{w_j}$ is the column of matrix $\mathbf{O}$ that corresponds to w_j. Matrix $\mathbf{H} \in \mathbb{R}^{|W| \times d}$ is associated with the hidden layer, while matrix $\mathbf{O} \in \mathbb{R}^{d \times |W|}$ is associated with the output layer. The word embeddings are contained in the rows of matrix $\mathbf{H}$. The main intuition behind the representations learnt by this model is that if two words w_i and w_j have similar contexts (i.e. they both co-occur with the same words), they obtain similar representations. In real scenarios, the above formulation is impractical due to the large vocabulary size, and hence, a negative sampling scheme is usually employed [75]. The training set is built by generating word-context pairs, i.e. for each word w_i and a training context c, we create pairs of the form $(w_i, w_{i-c}), \ldots, (w_i, w_{i-1}), (w_i, w_{i+1}), (w_i, w_{i+c})$.

Note that since the abstract of a paper contains multiple words, we need to aggregate the embeddings of the words to derive an embedding for the entire abstract. To this end, we simply average the representations of the words. Furthermore, an author may have published multiple

papers, and to produce a vector representation for the author, we again compute the average of the representations of (the available subset of) their papers:

$$\mathbf{h}_v = \frac{1}{|P_v|} \sum_{d \in P_v} \frac{1}{|d|} \sum_{w \in d} \mathbf{v}_w \tag{6.10}$$

GRAPH LEARNING As defined in Chapter 2, a GNN model consists of a series of aggregation layers, where a different combination creates a different GNN variant. We use two different GNN variants which differ from each other only in terms of the employed message-passing procedure. The first is the **GCN** introduced in Chapter 2 Eq. 2.8.

The second corresponds to the GIN-0 model [226]. Given the aforementioned co-authorship graph $G = (V, E)$ where nodes are annotated with feature vectors $\mathbf{h}_v^{(0)} \in \mathbb{R}^d$ stemming from the learnt representations of the author's papers and/or the graph metrics, each neighborhood aggregation layer of the first model updates the representations of the nodes as follows:

$$\mathbf{h}_v^{(t)} = \text{ReLU}\left(\mathbf{W}^{(t)}\, \mathbf{h}_v^{(t-1)} + \sum_{u \in \mathcal{N}(v)} \mathbf{W}^{(t)}\, \mathbf{h}_u^{(t-1)}\right) \tag{6.11}$$

where $\mathbf{W}^{(t)}$ is the matrix of trainable parameters of the t^{th} message-passing layer. In matrix form, the above is equivalent to:

$$\mathbf{H}^{(t)} = \text{ReLU}\left(\tilde{\mathbf{A}}\, \mathbf{H}^{(t-1)}\, \mathbf{W}^{(t)}\right) \tag{6.12}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. Note that besides the above two message-passing schemes, we also tried using GAT-like attention [208] in early experiments, without obtaining better results.

Similarly to the MPNN developed in Chapter 5 instead of using only the final node representations $\mathbf{h}_v^{(T)}$ (i.e. obtained after T message-passing steps), we also use the representations of the earlier steps $\mathbf{h}_v^{(1)}, \ldots, \mathbf{h}_v^{(T-1)}$ [227] to retain local and global information. Thus, we concatenate the representations produced at the different steps, finally obtaining $\mathbf{h}_v = [\mathbf{h}_v^{(1)} || \mathbf{h}_v^{(2)} || \ldots || \mathbf{h}_v^{(T)}]$. These node representations are then passed on to one or more fully-connected layers to produce the output, as we did in Chapter 5.

6.2.2.1 *Data*

The 'Field' allowed us to extract subgraphs of the network induced by authors that belong to a specific research field. In this experiment, due to the well-known difficulties of using GNNs on large graphs, we focused only on authors that have published papers in the field of Computer Science. It should be noted that the emerging dataset was missing textual information of some papers. If none of an author's papers contained text, we removed the corresponding node from the graph as we would not be able to leverage the heterogeneous information effectively. The final graph consists of 1,503,364 nodes and 37,010,860 edges. The weight of an edge is set equal to the number of papers two

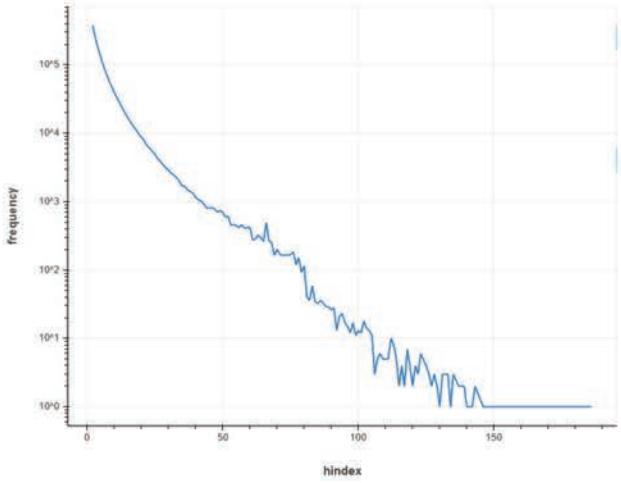


Figure 6.2: Distribution of h-index values of the authors contained in our dataset.

scholars have co-authored together. The h-index distribution of the authors is illustrated in Figure 6.2. Note that the values on the vertical axis are in logarithmic scale. As we see, it verifies the well-known power law distribution inherent in many real-world networks [63].

6.2.3 *Experimental Evaluation*

In this section, we first present the baselines against which we compared the proposed approach. We next give details about the employed experimental protocol. We last report on the performance of the different approaches and discuss the obtained results.

We compare the proposed models against the following learning models: (1) Lasso, a regression algorithm that performs both variable selection and regularization [200], (2) SGDRegressor, a linear model which is trained by minimizing a regularized empirical loss with SGD, (3) XGBoost, a scalable end-to-end tree boosting system [35], and a multi-layer perceptron (MLP). All the above models expect each input sample to be in the form of a vector. Given an author, the above vector is produced by the features extracted from the textual content of the author's papers and/or the features extracted from the co-authorship network.

In order to test the semi-supervised generalization capabilities of our model, we experimented with a 20/80 training/test split. For all algorithms, we standardize the input features by removing the mean and scaling to unit variance. With regards to the hyperparameters of the proposed models, we use 2 message-passing layers. The hidden-dimension size of the message-passing layers is set to 32 and 64, respectively. In the case of the first model (i.e. GNN), batch normalization is applied to the output of every message-passing layer. The representations produced by the second message-passing layer are passed on to a multi-layer perceptron with one hidden layer of dimensionality 64. All dense layers use ReLU activation. The dropout rate is set equal to 0.1. To train all models, we use the Adam optimizer with a learning rate of 0.01. We

Method	Text Features		Graph Features		All Features	
	MAE	MSE	MAE	MSE	MAE	MSE
Lasso	4.99	66.91	3.28	29.51	3.28	29.51
SGDRegressor	8.48	112.20	6.20	78.38	8.01	120.91
XGBoost	4.22	64.43	3.04	34.83	2.91	**21.04**
MLP	4.10	59.77	**2.62**	**22.46**	2.59	21.44
GCN	**4.05**	**59.45**	2.68	24.32	**2.57**	21.29
GNN	4.07	60.00	2.66	23.82	2.58	21.85

Table 6.2: Performance of the different methods in the h-index prediction task.

set the number of epochs to 300. The best model is chosen based on a validation experiment on a single 90% - 10% split of the training data and is stored. At the end of the training, the stored model is used to make predictions for the test instances. The MLP contains a hidden layer of dimensionality 64. All its remaining hyperparameters (e. g.dropout rate, activation function, etc.) take values identical to those of the two GNNs. For lasso, the weight of regularization is set to 1. For SGDRegressor, we use an l_2 regularizer with weight 0.0001. The initial learning rate is set to 0.01 and the number of epochs to 1000. For XGBoost, we use the default values of the hyperparameters.

The performance of the different models is illustrated in Table 6.2. We report the mean absolute error (MAE) and the mean squared error (MSE) of the different approaches. There are three different sets of features passed on to each algorithm (features extracted from graph, features extracted from text or from both). With regards to the performance of the different approaches, we first observe that neural network models outperform the other approaches in all settings and by considerable margins, except for one case. We hypothesize that this stems from the inherent capability of neural networks to detect meaningful and useful patterns in large amounts of data. In this case, though semi-supervised, the training set still consists of more than 300,000 samples, which allows for effective use of the representational power of neural network models. On the other hand, we see that XGBoost performs better in terms of MSE in one setting since it optimizes the MSE criterion, while the neural architectures are trained by minimizing a MAE objective function, as also shown in Figure 4.2. To train the proposed models, we chose to minimize MAE instead of MSE since MAE is more interpretable in the case of h-index, and provides a generic and even measure of how well our model is performing. Therefore, for very large differences between the h-index and the predicted h-index (e. g.$y = 120$ vs. $\hat{y} = 40$), the function does not magnify the error.

With regards to the different types of features, we can see that the graph metrics alone correspond to a much stronger predictor of the

Author	h-index	Predicted h-index	
		GCN	GNN
Jiawei Han	131	94.75	115.09
Jie Tang	45	30.71	33.74
Lada Adamic	48	30.67	22.71
Zoubin Ghahramani	77	40.64	43.33
Michalis Vazirgiannis	36	25.76	27.89
Jure Leskovec	68	37.77	38.26
Philip S Yu	120	100.42	119.93

Table 6.3: The actual h-index of a number of authors as defined from google scholar and their predicted h-index.

h-index compared to the features extracted from the papers' textual content. Due to the rich information encoded into these features, MLP achieves similar performance to that of the two GNN models. This is not surprising since GNN and GCN consist of two message-passing layers followed by a multi-layer perceptron which is identical in terms of architecture to MLP. The features extracted from the authors' papers seem not to capture the actual impact of the author. It is indeed hard to determine the impact of an author based solely on the textual content of a subset of the papers the author has published. Furthermore, these features have been produced from a limited number of an author's papers, and therefore, they might not be able to capture the author's relationship with similar authors because of the diversity of scientific abstracts and themes. GCN is in general the best-performing method and yields the lowest levels of MAE. Overall, given the MAE of 2.57, we can argue that GNNs can be useful for semi-supervised prediction of the authors' h-index in real-world scenarios.

To qualitatively assess the effectiveness of the proposed models, we selected a number of well-known scholars in the field of data mining and present their h-index along with the predictions of our two models in Table 6.3. Keeping in mind that the overwhelming majority of authors have a relatively small h-index (as has been observed in Figure 6.2), it is clear that these are some of the most extreme and hard cases which can pose a significant challenge to the proposed models. Even though the objective function of the proposed models is MAE which does not place more weight on large errors, still, as can be seen in Table 6.3, the two models' predictions are relatively close to each other. More importantly, ranking based on the predictions is similar to ranking using the actual h-index values of the authors in most cases.

6.3 SCIENTOMETRIC ANALYSIS WEB APPLICATION

Given the demand of the real-world for qualitative results and visualization, we demonstrate a set of analytics that provides novel insights on a scientist's impact. More specifically, we utilize information regarding papers, their authors and the fields they belong to, to extract field-based citations and the *Author Oriented Citation (AOCI)* network (to our knowledge the largest existing of its kind). We then derive two interactive visualizations that aspire to convey the author's success and influence over the academic network:

- We extract a field-based h-index (*f-h-index*) for each author using the citations she receives in her papers that belong to specific fields. Afterwards we form *f-h-index* distributions and position the author's *f-h-index* in them together with the top percentile she belongs to, for the distributions of the top three fields she most frequently publishes in.

- We compute the *D-core decomposition* of the AOCI graph, a measure of influence in directed networks [73] adopted in the Aminer scientific search tool. Subsequently, we use the sub-graphs induced by the decomposition to form the D-core matrix, a rectangular heat map that displays the outmost cores (i.e. the densest citation graphs) that an author belongs to, indicating her aggregate influence in terms of authority (incoming citations) and community integration (in terms of outgoing citations).

Both visualizations along with the corresponding author search engines are deployed online[3]. The first challenge towards computing the above was ensuring the quality of the dataset. Although MAG has been preprocessed extensively, we found out that more than 30% of the unique names have been assigned multiple IDs. To dive deeper, we performed an exploratory analysis on the number of papers for these IDs and show that more than 80% have been assigned to one paper only. This means that their h-index is 1 and they will have a minuscule contribution to the influence estimation, thus we proceed to remove them from the analysis. The second challenge adheres to the issue of scalability. Creating AOCI in a typical manner is not feasible since it includes a join operation between a table of 320Gb with a table of 30Gb. We overcome this by breaking the author-cites-paper table in multiple batches with non-overlapping authors and calculating the ego-networks of the authors in each batch in parallel. Subsequently, we filter the network based on edge weights to keep only the most consistent citing relationships and run the D-core decomposition. As a means of anecdotal evaluation, we examine the densest D-core subgraph and provide a list of the most influential authors as well as a wordcloud with the most frequent scientific fields in it. The vast majority of the retrieved fields are related to particle physics, a field well known for its massively dense collaboration and citation patterns [44]. The overall

3 http://graphdegeneracy.org/scientometrics/

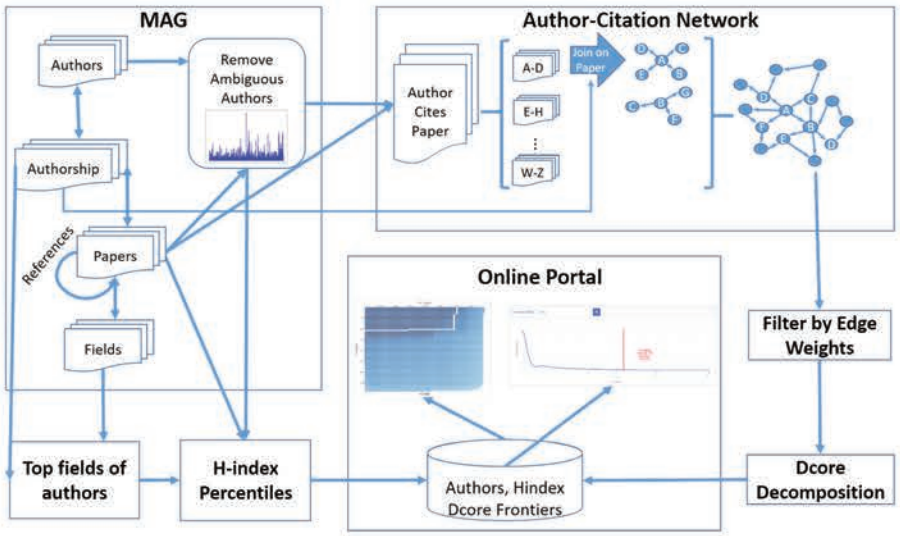


Figure 6.3: Schematic representation of the data pipeline.

data pipeline employed can be seen at Figure 6.3. The code to reproduce the analysis can be found online[4].

6.3.1 *Field h-index Distribution*

As mentioned above, h-index is considered one of the most successful scientometrics [91], employed uniformly from academic institutions and bibliographical platforms. In spite of its broad usage though, it is known to suffer from certain limitations [24], one notably being that as is just a number, it can be easily misinterpreted if presented out of context. In other words, a scientist's success might be best depicted by comparing her h-index with the rest of academia. Another important disadvantage is the significant different rates of citations exhibited throughout different principles [211, 241], which renders interdisciplinary comparisons impossible. More specifically, in several occasions we aim to compare scientists from aberrant but collaborating fields, like biology and computer science, where that latter has a significantly lower publication rate than the former. In our dataset, biology has the higher average h-index with an average of 7.1 and compute science is seventh with 5.9 because h-index is biased towards scientific fields with increased publication volume. To this end, we propose a set of plots that present an author's position in her most relevant fields, by visualizing the h-index from citations she receives in papers that belong to each field (*f-h-index*), relative to the distribution of all authors' *f-h-index*.

To retrieve a field-specific h-index, we first assigned fields to each paper based on the subfield related to it, which was assigned with an associated *confidence* metric by Microsoft Academic Services [185]. The fields can be seen in Table (6.4). In case the sub-field contained no categories, we assigned to it the categories of its related sub-field. Thus, each paper p has a number of fields f assigned to it with a certain

4 https://github.com/geopanag/MAG-Author-Influence-Scientometrics

biology	medicine
mathematics	computer science
engineering	chemistry
physics	business
visual arts	media
language	geography
agriculture	food

Table 6.4: The 14 fields assigned to papers in Microsoft Academic Graph

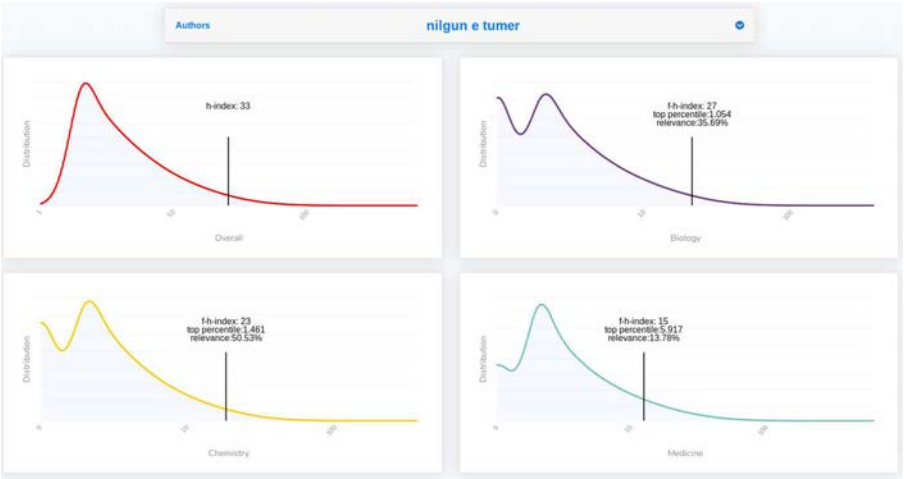


Figure 6.4: Field h-index percentiles for a researcher in chemistry, biology and medicine.

confidence $c_{p,f}$. To retrieve the most prevalent fields for an author u with P_u papers, we computed her *relevance* to a specific field as:

$$r_{u,f} = \sum_{p \in P_u} c_{p,f} \tag{6.13}$$

We keep the top three fields for each author based on this ranking. Subsequently, we separate the papers she wrote in each of these fields, and derive a list of their citations. Note here that a paper might belong to more than one field, hence its citations are included in the lists of all these fields.

The *f-h-index* is computed through the typical h-index formula analyzed above but adjusted for each field. Specifically, we sort the author's papers on a given field based on their number of citations and taking the maximum position h where the citations are at least h. We also derive the general h-index of all authors, as an indication of an author's overall performance. The relationship between an author's top three *f-h-index* and her h-index is not straight forward because of the presence of papers in multiple fields and our disregard for the fields she publishes more rare to i.e. they are not in the top three. Apart from the aforementioned preprocessing, we aggregate authors based on their name and keep the record with the biggest h-index, which results in a set of $23,202,638$ names. In order to make the distributions more legible

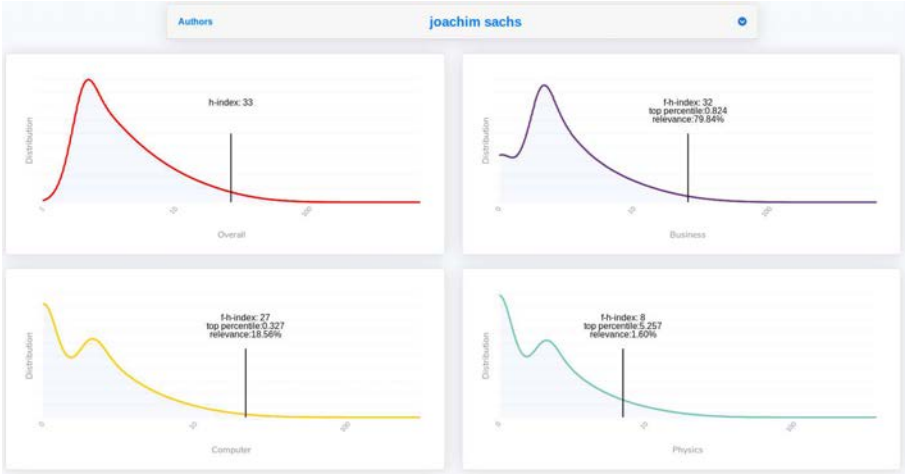


Figure 6.5: Field h-index percentiles for a researcher in business, computer science and physics.

we also removed all authors with an h-index of 1, which diminishes the number to $10,050,770$. Finally, the distribution is in logarithmic scale to alleviate the severe skewness that characterizes such distributions. The search is based on the last name and the results are sorted in descending fashion based on the overall h-index.
An example of use case can be seen in Figures 6.4 and 6.5. Both researchers have the same overall h-index, but the percentiles they belong to in their specific fields are different. More specifically, the one is close to the top 0.3 in computer science and 0.8 in business which is his actual field based on the normalized relevance score, while for the other, the highest percentile is close to 1 and over 1.45 for his actual field. Though indisputably both authors are successful, dr. Joachim Sachs can be considered more impactful in his fields, using this comparison.

6.3.2 *Author Influence*

The influence exerted by scientists to the academic world can be measured from multiple different perspectives. One could argue that co-authorship is a form of influence [180]. However, scientists can have an exceptional number of co-authors without performing remarkable work, thus citation could be considered an indication of influence. The semantic scholar defines an author's influence on another author based on how many times the latter cites the former [204]. Similar works have analyzed the impact of an institution given its citations to and from other institutions [143]. That said, simply relying on the number of citations does not suffice, as authors may get cited a lot but from a limited number of people, which constraints their actual visibility. Consequently, we deem more appropriate to measures influence based on the scientist's placement in the aforementioned AOCI, which can capture both, the number of in and out citations as well as the number of people that cite and get cited by an author. Especially the out citations is a metric that shows activity and integration in the community.

One of the most popular node influence metrics in other applications is the max core it resides in [140], based on the k-core decomposition analyzed in Chapter 2. However, the citation is a strictly directed relationship, meaning that there is a significant difference between citing and getting cited. In order to sustain this dual nature of in and out citations, we employ the D-core decomposition [73], a well-known adaptation of k-core that captures both the authority (via the in citations) and the integration (via the out citations) of an author's directed influence and has been adopted by the popular bibliographic exploration system Aminer.org [5].

The sole extraction of AOCI from the MAG was not straight forward due to the scale of the dataset and its dense nature. More specifically, given a certain paper with c citations, each cited paper with an average of a authors, the paper's authors will acquire $c * a$ edges, without taking author overlapping into account. To form the graph we used the paper-references table and the paper-author table from MAG[6]. We first created a table that consists of the authors and the papers they cite, by a simple join on the paper ID. Subsequently, we sort based on the author ID and break the table in 20 batches with no author shared. We can then join each batch with the paper-author table using the reference ID as key to the paper ID to compute author-cites-author recordings which can be turned in a weighted edge-list grouping by the authors and counting their co-occurrence. Essentially this is the out-egonetwork of each author in that batch, and we can create the whole network by their combination, as shown at Figure 6.3. The resulting edge weights are number of times an author cited another author. This creates a network of $67,614,736$ unique author ids with over 17 billion edges. By removing the aforementioned ambiguous IDs and the edges with weight less than 10, we end up with a directed network of $8,960,233$ nodes and $599,586,916$ edges. Removing edges with equal or less then 10 citations allows us to keep only the nodes that consistently cite each other. In other words, apart from reducing the size of the network in a manageable scale, it also allows us to perform a preliminary community detection, as the consistent citing patterns between authors represent the circulation and development of certain scientific ideas.

Given a directed network $D = (V, E)$ with a set V of nodes and a set E of directed edges between them, following the literature [73], we define a directed core $DC_{k,l}$ as the subgraph were each node $v \in V(DC_{k,l})$, has at least $deg_v^{in}(DC_{k,l}) \geq k$ and $deg_v^{out}(DC_{k,l}) \geq l$. It should be noted that like the original k-core decomposition analyzed in Chapter 2, this is not one-time filtering but rather a recursive process where the nodes that are not in accordance with the aforementioned criteria are removed iteratively. This means that even if a node initially complies with the requirements, it may eventually be removed after removing the rest of the nodes that do not comply if any of them are its neighbors. To compute the decomposition we first define the maximum K and L, which are the last bounds where the D-core is non-empty. We iterate over all

5 https://aminer.org/

6 All the queries were performed in PySpark in a cluster of 32 nodes, 16GB ram each, Intel(R) Xeon(R) CPU E5-2407 v2 @ 2.40GHz

$k' \in [0, K]$ and perform the decomposition for $k = k'$ and all $l \in [0, L_{k'}]$, where $DC_{k',L_{k'}+1}$ is empty. This provides us with a list of $L_{k'}$ subgraphs for each k'. Since this computation is independent throughout different k', we can parallelize this process by running a different range in $[0, K]$ to a different CPU. We used the retrieved D-cores to create a D-core matrix, a heatmap with the size of the D-cores in a gradient motif. We can then visualize a scientist's authority/integration aptitude based on the outmost D-cores she belongs to, as indicated in the example of Figure 6.6. We remove authors with the same name to facilitate search, based on who has the highest D-cores, and we are left with $6,252,393$ names. Since the maximum incore and outcore found are 7800 and 7900 respectively, creating an interactive heatmap of such scale is neither practical nor feasible. To this end, we break the plot in two versions. The 'macro' version shows the densest cores and uses a step of 100, meaning the D-cores shown are for $mod(k, 100) = 0$ and $mod(l, 100) = 0$. This includes only 175.952 authors, while rest lie in the D-cores for $k, l \in [0, 100]$, in which case we employ the "micro" version where the D-core matrix has a step of 1.

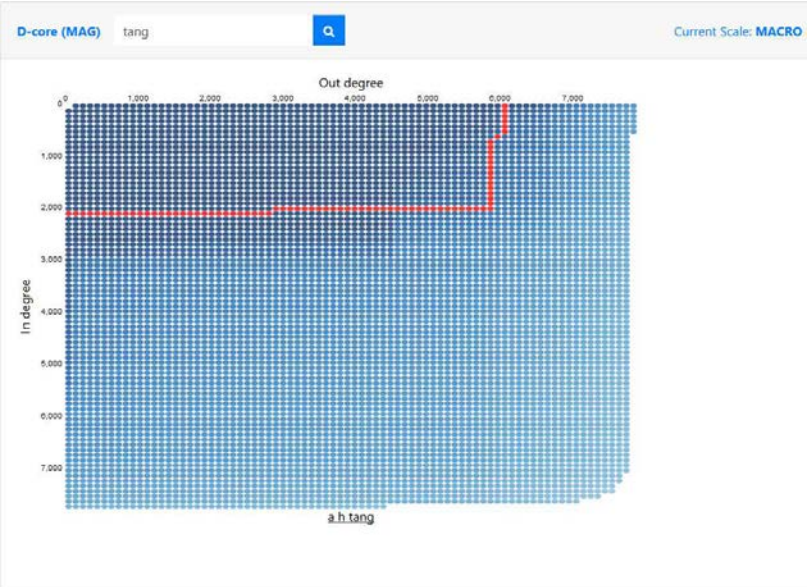


Figure 6.6: D-core matrix visualization.

The aforementioned plot can be used to derive the most influential users in the network, which reside in the optimal collaboration core (OCD-core) based on $OCI = (k + l)/2$ [73], in our case $k = 7600$ and $l = 7480$. The top authors in OCD-core in terms of degree, which resembles the number of people they cite and getting cited by, along with the h-index and their fields of study as found by searching in the web, are shown in table (6.5). We notice that degree and h-index are not totally correlated - implying that the density of the citation network captures different influence aspects, as indicated in the recent rooted citation influence metric [72].

We can observe that the majority of the top authors are physicists. In general, OCD-core includes authors that cite and getting cited by each other in a massive rate, which could be characteristic of some fields. To

Name	Degree	h-index	Field of Study
Hsi Shu Chen	80300	88	Human Space Systems
Andrea Bocci	73476	98	Particle Physics
Swagato Banerjee	72728	86	Particle Physics
Barbara Abbott	72243	84	Developmental Biology
Pauk Kyberd	70650	83	Particle Physics
Martin Weber	69633	77	Particle Physics
George Azuelos	69372	80	Materials Science
Wolfram D. Zeuner	69266	79	Particle Physics
Thomas Hebbeker	69247	88	Particle Physics
Daria Bisello	69211	86	Particle Physics

Table 6.5: Top 10 authors in terms of degree in OCD-core.

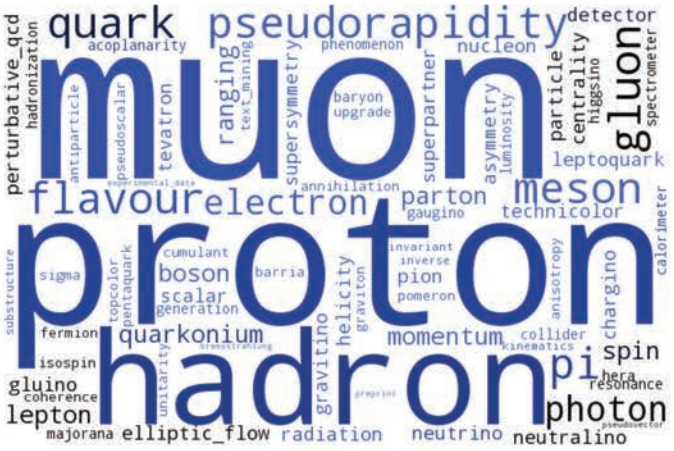


Figure 6.7: The most frequent fields appearing in the OCD-core.

evaluate further this hypothesis, we employ the one-word fields assigned to papers in MAG, keeping only assignments with confidence$>$ 0.5. We retrieve the fields of the papers of the authors in OCD-core and aggregated the number of authors in each field. Figure 6.7 shows the word-cloud indicating the number of authors of the top 100 fields. As one can see, most of the words belong to particle physics and related fields.This happens because physics laboratories throughout the world collaborate with each other to validate their experimental findings. Thus the publications are comprised of tens or even hundreds of authors who cite similar older publications, resulting in a very dense network with an increased rate of citations.

6.4 CONCLUSION

In this chapter we described a heterogeneous GNN that produces quantitiative estimates of a scientist's success. Furthermore, we presented

an online system with interactive visualizations that capitalize on the author oriented citation network and field-based citations from MAG to capture different dimensions of a scientist's impact. We claim that this effort contributes to talent management and ranking within but also out of the academic domain. In terms of future work, we plan to extend our online portal portal further with the addition of several novel analytics that address other well-known scientometric problems such as self-citation ratios. Moreover, as the author to author citing patterns are inherently weighted (reflecting the quantified impact of an author to another) - we plan to compute the D-core taking the edge weights into account, aiming at more informative and valid influence rankings. Finally, we aspire to advance the GNN to address the rising star prediction problem [166] and incorporate it into the online portal.

7 CONCLUSION

The prominence of social networks and their projected effect on our future lives has motivated the analysis of influence relationships using algorithmic and database methods. With the increasing data heterogeneity and availability in the web, machine learning methods emerge as a promising workhorse to improve the solutions on problems that were addressed using algorithms or simulation models in the past. In this thesis, we have proposed methods that complement or substitute solutions based on non-learning algorithms or models on graph mining problems that pertain to social influence.

7.1 SUMMARY OF CONTRIBUTIONS

We have introduced several methods that tackle different problems in graph mining. The main contributions of the thesis are:

DIFFUSION-BASED INFLUENCE MAXIMIZATION Previous approaches to influence maximization relied heavily on randomly assigned influence probabilities. We proposed a machine learning method to include complementary information such as diffusion cascades to circumvent this problem. Specifically, we leverage a multi-task neural model that learns from historical traces of information flow through the network to derive representations that capture the influence relationships between nodes. Based on these, we propose a new formulation of the influence spread, which is proved to be submodular and monotone. IMINFECTOR is proposed to optimize efficiently the new marginal gain using the representations and the prediction of a node's influence aptitude. Experimental results showcase the clear advantage of the proposed method over benchmarks, either in accuracy or computational efficiency.

LEARNING TO ESTIMATE INFLUENCE For networks with no complementary data, we propose to utilize graph neural networks to improve computational efficiency. We developed GLIE, a Graph Neural Network developed based on an upper bound of influence estimation and trained with supervision to tighten this bound. We rely on GLIE to propose three methods for influence maximization. 1) An adaptation of a classical algorithm that surpasses SOTA 2) a Q-learning model that learns to retrieve seeds sequentially 3) a submodular function that acts as a proxy for the marginal gain and can be optimized adaptively. Our experiments indicate that GLIE is faster compared to alternatives and its error is comparable for small seed sets and graphs up to 10 times larger than the train set. Moreover, the proposed influence maximization methods surpass the benchmarks in efficiency and effectiveness.

NEURAL-NETWORK BASED EPIDEMIC FORECASTING The recent pandemic of COVID-19 served as motivation for the development of novel machine learning solutions. Here, we proposed a graph learning solution to epidemic forecasting, a problem traditionally solved through mathematical models. Initially, a temporal graph is created using data from mobile phone apps indicating mass movements between small geographical regions in different European countries. Subsequently, we develop a graph neural network that creates representations based on the mobility patterns on the graph and the recent history of the epidemic in the area. The model is trained to produce forecasts of daily COVID-19 positive cases in each region. To account for the cold start problem inherent during the first weeks of the pandemic outbreak in a country, we apply Model Agnostic Meta-Learning. The method utilizes data from countries hit earlier by the pandemic to come up with better initialization points to start training on the current country's data. Our results on 4 different countries indicate that the proposed method can retrieve thousands of unseen cases compared to standard methods for epidemic forecasting, and justify the use of transfer learning.

PREDICTION AND EVALUATION OF ACADEMIC INFLUENCE Academic evaluation has been a prevalent problem for practical reasons of hiring or funding, which has intensified with the democratization of science in social media due to the public's need to understand and follow scientific authorities. In our case, we developed computational methods to predict and assess the influence of a scientist, using the biggest heterogeneous bibliometric dataset available, the Microsoft Academic Graph. We developed a graph neural network that takes as input the coauthorship graph and the text representations of the author's papers and predicts the author's h-index. Moreover, we created an online web application to visualize the overall impact of a scientist qualitatively. Initially, we developed a distribution plot of the field-based h-indices, to visualize the position of each author in her most relevant fields. Finally, we compute and visualize the directed core metric of the authors in an author-citation network, which quantifies an author's authority/integration over the rest of the academic world.

7.2 FUTURE STEPS

In this section, we propose future research directions and open questions on the problems addressed in the current thesis as well as problems pertaining to the application of submodular maximization in machine learning.

HETEROGENEOUS NEURAL NETWORK FOR INFLUENCE ALGORITHMS In practice, the problem of estimating influence between two nodes includes multiple sources of information. Apart from the social graphs, a social network contains diffusion cascades, the topic of the information spread, or the information in the user profiles. In order to create a method for influence prediction and maximization that takes

multiple information sources into account, the most promising way is to rely on neural networks that retrieve heterogeneous embeddings from sequential events such as cascades, text or images that represent the post topic, and node characteristics that include demographics. This information along with the main graph forms a whole image of the user's state inside the system. It would allow the aforementioned problems to be solved conditioned on topic, time, trend, and other contextual data that require arduous work to include in current methodologies. Moreover, it can serve as means to sparsify the graph or cluster the users for recommendations or integrity purposes, such as identifying toxic cliques. One of the biggest challenges towards this goal, apart from the methodological novelty, is to create and open large-scale datasets with such heterogeneous information.

One important question that arises from our observations in Chapter 4 Section 4.4.3 where IMINFECTOR is compared with PUN in simulations and unseen cascades, is whether we can build a universal way to evaluate influence maximization algorithms. Our results indicate that simulations are not enough, but in many cases cascades are not available. Even in the presence of ground truth cascades, our proposed metric overlooks the contribution of participating in a cascade and the possibility of nodes starting copying information after a certain point, which is not realistic. Finally, another important goal is to gradually move towards model-independent methods, that would capture the diffusion property without relying on the standard diffusion models.

EFFICIENT TRAINING THROUGH SUBMODULAR ACTIVE LEARNING Training a neural network can be too computationally demanding, which is a substantial hinder for real-world systems that require continuous retraining because of the stream of input data. Particularly, when applied to web-scale graphs, graph neural networks suffer from long training times and obligatory information loss due to neighborhood sampling [34]. There is a clear need to achieve faster training times while minimizing the loss of supervision. One promising approach is active learning on the training batches, where sumodular optimization has proven effective [8, 42, 60, 76]. Although active learning on graphs has not been extensively examined, there are algorithms that can perform optimally on trees and with a provable error bound on regular graphs, [32]. It is thus a promising application of submodular optimization and online learning.

ENHANCING PREDICTIONS AND QUANTIFYING CONFIDENCE In our proposed applications, we utilized graph neural networks for real-world tasks such as pandemic forecasting and academic ranking. While the predictions are useful, real-world systems have to be able to explain and provide confidence intervals, especially in something as crucial and sensitive as pandemic forecasting [102]. Thus, one direct extension of our GNNs is to utilize explainability methods [56, 173] to improve the outcome of the models. In the application of academic ranking, we plan to expand it towards predicting future success while scientist is in the early stages of their career. This problem, commonly called rising star

identification [166], can be cast as a long-term forecasting method that aims to uncover which patterns are important for a scientist e.g. strong collaborations, heavy productivity etc. to succeed. Another interesting path is to enhance the visualizations in our web application based on other elaborate scientometrics, such as self-citation counts or weighted D-core decomposition.

A

APPENDIX

L'influence en ligne est le socle de l'effet des réseaux sociaux sur nos vies et son impact n'a cessé de croître. Du marketing viral aux campagnes politiques à la transmission de maladies, la façon dont nous sommes influencés par les autres est plus répandue que jamais. Dans cette thèse, nous abordons le problème de l'apprentissage et de l'analyse efficaces des représentations d'influence pour de nombreux problèmes d'exploration de graphes qui sont à propos.

La première moitié de la thèse est consacrée au problème de la maximisation de l'influence, un problème NP d'optimisation combinatoire. L'objectif est de trouver les nœuds d'un réseau qui peuvent maximiser la propagation de l'information, où la propagation est typiquement définie par des probabilités d'influence aléatoires et des modèles de diffusion simples. Pour répondre à ce problème, nous concevons un modèle d'apprentissage de la représentation des nœuds basé sur des cascades de diffusion ainsi qu'une adaptation d'un algorithme traditionnel de maximisation de l'influence qui utilise la sortie du modèle. Ce cadre surpasse les méthodes concurrentes, évaluées en termes de temps de calcul et d'influence des graines prédites dans les cascades du futur immédiat. La prochaine partie est consacrée à l'apprentissage de la maximisation de l'influence. Nous développons un réseau neuronal de graphe qui paramètre de manière inhérente une limite supérieure d'estimation de l'influence, et nous l'entraînons sur de petits graphes simulés. Nous montrons expérimentalement qu'il peut fournir des estimations précises plus rapidement que les autres solutions pour des graphes /dix/ fois plus grands que l'ensemble d'entraînement. En outre, nous utilisons les prédictions et les représentations des modèles pour proposer nouvelles méthodes de maximisation de l'influence. Une adaptation de un ancien algorithme, un modèle de Q-learning, et une fonction submodulaire qui agit comme un proxy pour le gain marginal et peut être optimisée de manière adaptative et avide avec certaines garanties théoriques. Cette dernière fonction offre le meilleur équilibre entre efficacité et précision.

Dans la deuxième moitié de la thèse, nous sommes concentrés sur des applications spécifiques de l'influence. Nous abordons la prévision des épidémies en utilisant l'apprentissage par influence. Nous utilisons le passage de messages inhérent aux réseaux neuronaux graphiques pour apprendre des représentations de nœuds basées sur les réseaux de mobilité des régions d'un pays et l'histoire de la contagion. Ces représentations sont utilisées pour prédire le nombre de nouveaux cas de COVID-19 avec une fenêtre de prévision allant jusqu'à 14 jours. En outre, pour tirer parti du décalage de la propagation entre les pays, un algorithme de méta-learning est proposé pour transférer les connaissances entre les modèles formés dans le cercle épidémique complet de certains pays, à un modèle prédisant les cas pour un autre pays au début de l'épidémie, où les données d'apprentissage disponibles sont

limitées. Notre approche surpasse les modèles de référence, les séries temporelles et d'autres modèles d'apprentissage profond.

Dans la dernière partie, nous analysons différentes versions de l'influence académique et des méthodes de dispositifs pour la quantifier et la prédire. Dans un premier temps, nous utilisons le MAG pour construire un réseau de citations d'auteurs avec des milliards d'arêtes. Nous le sous-échantillonnons et effectuons une décomposition en noyaux dirigés pour le quantifier et le visualiser au moyen d'une application web interactive. Ensuite, nous expérimentons la classification du h-index d'un auteur sur la base d'un GNN sur son graphe de coauteurs et le texte de ses articles. Nous concluons la thèse en abordant de futures directions concernant l'entraînement efficace des réseaux neuronaux par l'apprentissage actif submodulaire.

BIBLIOGRAPHY

[1] Alireza Abbasi, Jörn Altmann, and Liaquat Hossain. "Identifying the effects of co-authorship networks on the performance of scholars: A correlation and regression analysis of performance measures and social network analysis measures." In: *Journal of Informetrics* 5.4 (2011), pp. 594–607.

[2] Alireza Abbasi, Liaquat Hossain, and Loet Leydesdorff. "Betweenness centrality as a driver of preferential attachment in the evolution of research collaboration networks." In: *Journal of Informetrics* 6.3 (2012), pp. 403–412.

[3] Giovanni Abramo, Ciriaco Andrea D'Angelo, and Marco Solazzi. "The relationship between scientists' research performance and the degree of internationalization of their research." In: *Scientometrics* 86.3 (2011), pp. 629–643.

[4] Daniel E Acuna, Stefano Allesina, and Konrad P Kording. "Predicting scientific success." In: *Nature* 489.7415 (2012), pp. 201–202.

[5] Lada A Adamic and Natalie Glance. "The political blogosphere and the 2004 US election: divided they blog." In: *Proceedings of the 3rd international workshop on Link discovery*. 2005, pp. 36–43.

[6] James D Adams, Grant C Black, J Roger Clemmons, and Paula E Stephan. "Scientific teams and institutional collaborations: Evidence from US universities, 1981–1999." In: *Research Policy* 34.3 (2005), pp. 259–285.

[7] Réka Albert, Hawoong Jeong, and Albert-László Barabási. "Error and attack tolerance of complex networks." In: *nature* 406.6794 (2000), pp. 378–382.

[8] Ayya Alieva, Aiden Aceves, Jialin Song, Stephen Mayo, Yisong Yue, and Yuxin Chen. "Learning to Make Decisions via Submodular Regularization." In: *International Conference on Learning Representations*. 2020.

[9] Sinan Aral and Paramveer S Dhillon. "Social influence maximization under empirical influence models." In: *Nature Human Behaviour* 2.6 (2018), p. 375.

[10] Sinan Aral and Dean Eckles. "Protecting elections from social media manipulation." In: *Science* 365.6456 (2019), pp. 858–861.

[11] Sinan Aral and Dylan Walker. "Creating social contagion through viral product design: A randomized trial of peer influence in networks." In: *Management science* 57.9 (2011), pp. 1623–1639.

[12] Sinan Aral and Dylan Walker. "Identifying influential and susceptible members of social networks." In: *Science* 337.6092 (2012), pp. 337–341.

[13] Eytan Bakshy, Jake M Hofman, Winter A Mason, and Duncan J Watts. "Everyone's an influencer: quantifying influence on twitter." In: *Proceedings of the fourth ACM international conference on Web search and data mining*. 2011, pp. 65–74.

[14] Eytan Bakshy, Brian Karrer, and Lada A Adamic. "Social influence and the diffusion of user-created content." In: *Proceedings of the 10th ACM conference on Electronic commerce*. 2009, pp. 325–334.

[15] Eytan Bakshy, Solomon Messing, and Lada A Adamic. "Exposure to ideologically diverse news and opinion on Facebook." In: *Science* 348.6239 (2015), pp. 1130–1132.

[16] Albert-László Barabási and Réka Albert. "Emergence of scaling in random networks." In: *science* 286.5439 (1999), pp. 509–512.

[17] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. "Neural combinatorial optimization with reinforcement learning." In: *arXiv preprint arXiv:1611.09940* (2016).

[18] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. "Machine learning for combinatorial optimization: a methodological tour d'horizon." In: *European Journal of Operational Research* (2020).

[19] Shishir Bharathi, David Kempe, and Mahyar Salek. "Competitive influence maximization in social networks." In: *International workshop on web and internet economics*. Springer. 2007, pp. 306–311.

[20] Robert M Bond, Christopher J Fariss, Jason J Jones, Adam DI Kramer, Cameron Marlow, Jaime E Settle, and James H Fowler. "A 61-million-person experiment in social influence and political mobilization." In: *Nature* 489.7415 (2012), pp. 295–298.

[21] María Bordons, Javier Aparicio, Borja González-Albo, and Adrián A Díaz-Faes. "The relationship between the research performance of scientists and their position in co-authorship networks in three fields." In: *Journal of Informetrics* 9.1 (2015), pp. 135–144.

[22] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. "Maximizing social influence in nearly optimal time." In: *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2014, pp. 946–957.

[23] Lutz Bornmann and Hans-Dieter Daniel. "Does the h-index for ranking of scientists really work?" In: *Scientometrics* 65.3 (2005), pp. 391–392.

[24] Lutz Bornmann and Hans-Dieter Daniel. "What do we know about the h index?" In: *Journal of the American Society for Information Science and technology* 58.9 (2007), pp. 1381–1385.

[25] Lutz Bornmann, Rüdiger Mutz, and Hans-Dieter Daniel. "Are there better indices for evaluation purposes than the h index? A comparison of nine different variants of the h index using data from biomedicine." In: *Journal of the American Society for Information Science and Technology* 59.5 (2008), pp. 830–837.

[26] Léon Bottou. "Stochastic gradient descent tricks." In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[27] Simon Bourigault, Sylvain Lamprier, and Patrick Gallinari. "Representation learning for information diffusion through social networks: an embedded cascade model." In: *Proceedings of the Ninth ACM international conference on Web Search and Data Mining*. 2016, pp. 573–582.

[28] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[29] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners." In: *arXiv preprint arXiv:2005.14165* (2020).

[30] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. "Limiting the spread of misinformation in social networks." In: *Proceedings of the 20th international conference on World wide web*. 2011, pp. 665–674.

[31] Rich Caruana. "Multitask learning." In: *Machine Learning* 28.1 (1997), pp. 41–75.

[32] Nicolo Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. "Active learning on trees and graphs." In: *arXiv preprint arXiv:1301.5112* (2013).

[33] Deepayan Chakrabarti and Christos Faloutsos. "Graph mining: laws, tools, and case studies." In: *Synthesis Lectures on Data Mining and Knowledge Discovery* 7.1 (2012), pp. 1–207.

[34] Jie Chen, Tengfei Ma, and Cao Xiao. "Fastgcn: fast learning with graph convolutional networks via importance sampling." In: *arXiv preprint arXiv:1801.10247* (2018).

[35] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794.

[36] Wei Chen, Tian Lin, and Cheng Yang. "Real-time topic-aware influence maximization using preprocessing." In: *Computational social networks* 3.1 (2016), pp. 1–19.

[37] Wei Chen, Chi Wang, and Yajun Wang. "Scalable influence maximization for prevalent viral marketing in large-scale social networks." In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 1029–1038.

[38] Wei Chen, Yajun Wang, and Siyu Yang. "Efficient influence maximization in social networks." In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 199–208.

[39] Wei Chen, Yifei Yuan, and Li Zhang. "Scalable influence maximization in social networks under the linear threshold model." In: *2010 IEEE international conference on data mining*. IEEE. 2010, pp. 88–97.

[40] Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Wang. "Variational knowledge graph reasoning." In: *arXiv preprint arXiv:1803.06581* (2018).

[41] Xueqin Chen, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Fengli Zhang. "Information diffusion prediction via recurrent cascades convolution." In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE. 2019, pp. 770–781.

[42] Yuxin Chen and Andreas Krause. "Near-optimal batch mode active learning and adaptive submodular optimization." In: *International Conference on Machine Learning*. PMLR. 2013, pp. 160–168.

[43] Vinay Kumar Reddy Chimmula and Lei Zhang. "Time Series Forecasting of COVID-19 transmission in Canada Using LSTM Networks." In: *Chaos, Solitons & Fractals* (2020), p. 109864.

[44] Ivan Chompalov, Joel Genuth, and Wesley Shrum. "The organization of scientific collaborations." In: *Research Policy* 31.5 (2002), pp. 749–767.

[45] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F Werneck. "Sketch-based Influence Maximization and Computation: Scaling up with Guarantees." In: *International Conference on Conference on Information and Knowledge Management (CIKM)*. 2014, pp. 629–638.

[46] Vittoria Colizza, Alain Barrat, Marc Barthelemy, and Alessandro Vespignani. "Prediction and predictability of global epidemics: the role of the airline transportation network." In: *Bulletin of the American Physical Society* (2006).

[47] Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. "Learning combinatorial optimization algorithms over graphs." In: *arXiv preprint arXiv:1704.01665* (2017).

[48] Songgaojun Deng, Shusen Wang, Huzefa Rangwala, Lijing Wang, and Yue Ning. "Graph message passing with cross-location attentions for long-term ILI prediction." In: *arXiv preprint arXiv:1912.10202* (2019).

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

[50] Yuxiao Dong, Reid A Johnson, and Nitesh V Chawla. "Will this paper increase your h-index? Scientific impact prediction." In: *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*. 2015, pp. 149–158.

[51] Yuxiao Dong, Reid A Johnson, and Nitesh V Chawla. "Can scientific impact be predicted?" In: *IEEE Transactions on Big Data* 2.1 (2016), pp. 18–30.

[52] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. "Recurrent marked temporal point processes: Embedding event history to vector." In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. 2016, pp. 1555–1564.

[53] Nan Du, Yingyu Liang, Maria Balcan, and Le Song. "Influence function learning in information diffusion networks." In: *ICML*. 2014, pp. 2016–2024.

[54] Nan Du, Le Song, Manuel Gomez Rodriguez, and Hongyuan Zha. "Scalable influence estimation in continuous-time diffusion networks." In: *Advances in neural information processing systems*. 2013, pp. 3147–3155.

[55] Lorenzo Ductor. "Does co-authorship lead to higher academic productivity?" In: *Oxford Bulletin of Economics and Statistics* 77.3 (2015), pp. 385–407.

[56] Alexandre Duval and Fragkiskos D Malliaros. "GraphSVX: Shapley Value Explanations for Graph Neural Networks." In: *arXiv preprint arXiv:2104.10482* (2021).

[57] David Easley, Jon Kleinberg, et al. "Networks, crowds, and markets: Reasoning about a highly connected world." In: *Significance* 9.1 (2012), pp. 43–44.

[58] John P Eaton, James C Ward, Ajith Kumar, and Peter H Reingen. "Structural Analysis of Co-Author Relationships and Author Productivity in Selected Outlets for Consumer Behavior Research." In: *Journal of Consumer Psychology* 8.1 (1999), pp. 39–59.

[59] Leo Egghe. "Theory and practise of the g-index." In: *Scientometrics* 69.1 (2006), pp. 131–152.

[60] Hossein Esfandiari, Amin Karbasi, and Vahab Mirrokni. "Adaptivity in adaptive submodularity." In: *Conference on Learning Theory*. PMLR. 2021, pp. 1823–1846.

[61] Theodoros Evgeniou and Massimiliano Pontil. "Regularized multi–task learning." In: *KDD*. 2004.

[62] *Facebook Data For Good 2020 Annual Report*. https://dataforgood.fb.com/docs/facebook-data-for-good-2020-annual-report/. Accessed: 2021-08-20.

[63] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. "On power-law relationships of the internet topology." In: *ACM SIGCOMM computer communication review* 29.4 (1999), pp. 251–262.

[64] Changjun Fan, Li Zeng, Yizhou Sun, and Yang-Yu Liu. "Finding key players in complex networks through deep reinforcement learning." In: *Nature Machine Intelligence* (2020), pp. 1–8.

[65] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. "Graph neural networks for social recommendation." In: *The World Wide Web Conference*. 2019, pp. 417–426.

[66] Shanshan Feng, Gao Cong, Arijit Khan, Xiucheng Li, Yong Liu, and Yeow Meng Chee. "Inf2vec: Latent Representation Model for Social Influence Embedding." In: *ICDE*. 2018.

[67] William D Figg, Lara Dunn, David J Liewehr, Seth M Steinberg, Paul W Thurman, J Carl Barrett, and Julian Birkinshaw. "Scientific collaboration results in higher citation rates of published articles." In: *Pharmacotherapy: The Journal of Human Pharmacology and Drug Therapy* 26.6 (2006), pp. 759–767.

[68] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." In: *34th International Conference on Machine Learning*. 2017, pp. 1126–1135.

[69] Seth Flaxman, Swapnil Mishra, Axel Gandy, H Juliette T Unwin, Thomas A Mellan, Helen Coupland, Charles Whittaker, Harrison Zhu, Tresnia Berah, Jeffrey W Eaton, et al. "Estimating the effects of non-pharmaceutical interventions on COVID-19 in Europe." In: *Nature* 584.7820 (2020), pp. 257–261.

[70] Lazaros K Gallos, Chaoming Song, and Hernán A Makse. "Scaling of degree correlations and its influence on diffusion in scale-free networks." In: *Physical review letters* 100.24 (2008), p. 248701.

[71] Junyi Gao, Rakshith Sharma, Cheng Qian, Lucas M Glass, Jeffrey Spaeder, Justin Romberg, Jimeng Sun, and Cao Xiao. "STAN: Spatio-Temporal Attention Network for Pandemic Prediction Using Real World Evidence." In: *arXiv preprint arXiv:2008.04215* (2020).

[72] Christos Giatsidis, Giannis Nikolentzos, Chenhui Zhang, Jie Tang, and Michalis Vazirgiannis. "Rooted citation graphs density metrics for research papers influence evaluation." In: *Journal of Informetrics* 13.2 (2019), pp. 757 –768.

[73] Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis. "D-cores: measuring collaboration of directed graphs based on degeneracy." In: *Knowledge and information systems* 35.2 (2013), pp. 311–343.

[74] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. "Neural message passing for Quantum chemistry." In: *34th International Conference on Machine Learning*. 2017, pp. 1263–1272.

[75] Yoav Goldberg and Omer Levy. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method." In: *arXiv preprint arXiv:1402.3722* (2014).

[76] Daniel Golovin and Andreas Krause. "Adaptive submodularity: Theory and applications in active learning and stochastic optimization." In: *Journal of Artificial Intelligence Research* 42 (2011), pp. 427–486.

[77] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Scholkopf. "Uncovering the Temporal Dynamics of Diffusion Networks." In: *International Conference on Machine Learning (ICML)*. 2011, pp. 561–568.

[78] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[79] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. "Learning influence probabilities in social networks." In: *WSDM*. 2010.

[80] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. "A data-based approach to social influence maximization." In: *VLDB* (2011).

[81] Amit Goyal, Wei Lu, and Laks VS Lakshmanan. "Simpath: An efficient Algorithm for Influence Maximization under the Linear Threshold model." In: *ICDM*. 2011.

[82] Alex Graves and Navdeep Jaitly. "Towards End-to-End Speech Recognition with Recurrent Neural Networks." In: *31st International Conference on Machine Learning*. 2014, pp. 1764–1772.

[83] Jonathan L Gross and Thomas W Tucker. *Topological graph theory*. Courier Corporation, 2001.

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

[85] Raf Guns and Ronald Rousseau. "Recommending research collaborations using link prediction and random forest classifiers." In: *Scientometrics* 101.2 (2014), pp. 1461–1473.

[86] William L Hamilton. "Graph representation learning." In: *Synthesis Lectures on Artifical Intelligence and Machine Learning* 14.3 (2020), pp. 1–159.

[87] Simon Haykin. *Neural networks and learning machines, 3/E*. Pearson Education India, 2010.

[88] Raphael H Heiberger and Oliver J Wieczorek. "Choosing collaboration partners. How scientific success in physics depends on network positions." In: *arXiv preprint arXiv:1608.03251* (2016).

[89] Karin Hellfeldt, Laura López-Romero, and Henrik Andershed. "Cyberbullying and psychological well-being in young adolescence: the potential protective mediation effects of social support from family, friends, and teachers." In: *International journal of environmental research and public health* 17.1 (2020), p. 45.

[90] Alfred Hermida, Fred Fletcher, Darryl Korell, and Donna Logan. "Share, like, recommend: Decoding the social media news consumer." In: *Journalism studies* 13.5-6 (2012), pp. 815–824.

[91] Jorge E Hirsch. "An index to quantify an individual's scientific research output." In: *Proceedings of the National academy of Sciences* 102.46 (2005), pp. 16569–16572.

[92] Petter Holme and Beom Jun Kim. "Growing scale-free networks with tunable clustering." In: *Physical review E* 65.2 (2002), p. 026107.

[93] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." In: *Neural networks* 2.5 (1989), pp. 359–366.

[94] Jiann-Wien Hsu and Ding-Wei Huang. "Correlation between impact and collaboration." In: *Scientometrics* 86.2 (2011), pp. 317–324.

[95] Zhigang Hu, Chaomei Chen, and Zeyuan Liu. "How are collaboration and productivity correlated at various career stages of scientists?" In: *Scientometrics* 101.2 (2014), pp. 1553–1564.

[96] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. "Heterogeneous graph transformer." In: *Proceedings of The Web Conference 2020*. 2020, pp. 2704–2710.

[97] Mohammad Raihanul Islam, Sathappan Muthiah, Bijaya Adhikari, B Aditya Prakash, and Naren Ramakrishnan. "Deepdiffuse: Predicting the'who'and'when'in cascades." In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2018, pp. 1055–1060.

[98] Shahla Jafari and Hamidreza Navidi. "A game-theoretic approach for modeling competitive diffusion over social networks." In: *Games* 9.1 (2018), p. 8.

[99] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[100] Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. "On learning paradigms for the travelling salesman problem." In: *arXiv preprint arXiv:1910.07210* (2019).

[101] Kyomin Jung, Wooram Heo, and Wei Chen. "Irie: Scalable and robust influence maximization in social networks." In: *2012 IEEE 12th International Conference on Data Mining*. IEEE. 2012, pp. 918–923.

[102] Harshavardhan Kamarthi, Lingkai Kong, Alexander Rodríguez, Chao Zhang, and B Aditya Prakash. "When in Doubt: Neural Non-Parametric Uncertainty Quantification for Epidemic Forecasting." In: *arXiv preprint arXiv:2106.03904* (2021).

[103] Amol Kapoor, Xue Ben, Luyang Liu, Bryan Perozzi, Matt Barnes, Martin Blais, and Shawn O'Banion. "Examining COVID-19 Forecasting using Spatio-Temporal Graph Neural Networks." In: *16th International Workshop on Mining and Learning with Graphs*. 2020.

[104] Márton Karsai, Mikko Kivelä, Raj Kumar Pan, Kimmo Kaski, János Kertész, A-L Barabási, and Jari Saramäki. "Small but slow world: How network topology and burstiness slow down spreading." In: *Physical Review E* 83.2 (2011), p. 025102.

[105] J Katz and Diana Hicks. "How much is a collaboration worth? A calibrated bibliometric model." In: *Scientometrics* 40.3 (1997), pp. 541–554.

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

[107] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks." In: *5th International Conference on Learning Representations*. 2017.

[108] Wouter Kool, Herke Van Hoof, and Max Welling. "Attention, learn to solve routing problems!" In: *arXiv preprint arXiv:1803.08475* (2018).

[109] Andreas Krause and Daniel Golovin. "Submodular function maximization." In: *Tractability* 3 (2014), pp. 71–104.

[110] Hildrun Kretschmer. "Author productivity and geodesic distance in bibliographic co-authorship networks, and visibility on the Web." In: *Scientometrics* 60.3 (2004), pp. 409–420.

[111] Tadeusz Kufel. "ARIMA-based forecasting of the dynamics of confirmed Covid-19 cases for selected European countries." In: *Equilibrium. Quarterly Journal of Economics and Economic Policy* 15.2 (2020), pp. 181–204.

[112] Jérôme Kunegis. "Konect: the koblenz network collection." In: *Proceedings of the 22nd international conference on World Wide Web*. 2013, pp. 1343–1350.

[113] Paul Lagrée, Olivier Cappé, Bogdan Cautis, and Silviu Maniu. "Algorithms for Online Influencer Marketing." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13.1 (2018), p. 3.

[114] Vasileios Lampos, Simon Moura, Elad Yom-Tov, Ingemar J Cox, Rachel McKendry, and Michael Edelstein. "Tracking COVID-19 using online search." In: *arXiv preprint arXiv:2003.08086* (2020).

[115] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." In: *nature* 521.7553 (2015), pp. 436–444.

[116] Jaekoo Lee, Hyunjae Kim, Jongsun Lee, and Sungroh Yoon. "Transfer learning for deep learning on graph-structured data." In: *31st AAAI Conference on Artificial Intelligence*. 2017.

[117] Sooho Lee and Barry Bozeman. "The impact of research collaboration on scientific productivity." In: *Social Studies of Science* 35.5 (2005), pp. 673–702.

[118] Kristina Lerman, Rumi Ghosh, and Tawan Surachawala. "Social contagion: An empirical study of information spread on Digg and Twitter follower graphs." In: *arXiv preprint arXiv:1202.3162* (2012).

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

[120] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. "Graph evolution: Densification and shrinking diameters." In: *ACM transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), 2–es.

[121] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. "Cost-effective outbreak detection in networks." In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, pp. 420–429.

[122] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters." In: *Internet Mathematics* 6.1 (2009), pp. 29–123.

[123] Cheng Li, Jiaqi Ma, Xiaoxiao Guo, and Qiaozhu Mei. "DeepCas: An end-to-end predictor of information cascades." In: *International Conference on World Wide Web (The WebConf)*. 2017, pp. 577–586.

[124] Cheng Li, Jiaqi Ma, Xiaoxiao Guo, and Qiaozhu Mei. "Deepcas: An end-to-end predictor of information cascades." In: *Proceedings of the 26th international conference on World Wide Web*. 2017, pp. 577–586.

[125] Hui Li, Mengting Xu, Sourav S Bhowmick, Changsheng Sun, Zhongyuan Jiang, and Jiangtao Cui. "DISCO influence maximization meets network embedding and deep learning." In: *arXiv preprint arXiv:1906.07378* (2019).

[126] Na Li and Denis Gillet. "Identifying influential scholars in academic social media platforms." In: *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 2013, pp. 608–614.

[127] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. "Influence maximization on social graphs: A survey." In: *IEEE Transactions on Knowledge and Data Engineering* 30.10 (2018), pp. 1852–1872.

[128] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. "Combinatorial optimization with graph convolutional networks and guided tree search." In: *arXiv preprint arXiv:1810.10659* (2018).

[129] Bo Liu, Gao Cong, Yifeng Zeng, Dong Xu, and Yeow Meng Chee. "Influence spreading path and its application to the time constrained social influence maximization problem and beyond." In: *IEEE Transactions on Knowledge and Data Engineering* 26.8 (2013), pp. 1904–1917.

[130] Shenghua Liu, Huawei Shen, Houdong Zheng, Xueqi Cheng, and Xiangwen Liao. "CT LIS: Learning Influences and Susceptibilities through Temporal Behaviors." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13.6 (2019), pp. 1–21.

[131] Andrey Y Lokhov and David Saad. "Scalable Influence Estimation Without Sampling." In: *arXiv preprint arXiv:1912.12749* (2019).

[132] Edward Loper and Steven Bird. "NLTK: the natural language toolkit." In: *arXiv preprint cs/0205028* (2002).

[133] Lars Lorch, William Trouleau, Stratis Tsirtsis, Aron Szanto, Bernhard Schölkopf, and Manuel Gomez-Rodriguez. "A spatiotemporal epidemic model to quantify the effects of contact tracing, testing, and containment." In: *arXiv preprint arXiv:2004.07641* (2020).

[134] Paige Maas, Shankar Iyer, Andreas Gros, Wonhee Park, Laura McGorman, Chaya Nayak, and P Alex Dow. "Facebook Disaster Maps: Aggregate Insights for Crisis Response & Recovery." In: *ISCRAM*. 2019.

[135] Sakib Mahmud. "Bangladesh COVID-19 Daily Cases Time Series Analysis using Facebook Prophet Model." In: *Available at SSRN 3660368* (2020).

[136] Fragkiskos D Malliaros, Christos Giatsidis, Apostolos N Papadopoulos, and Michalis Vazirgiannis. "The core decomposition of networks: Theory, algorithms and applications." In: *The VLDB Journal* 29.1 (2020), pp. 61–92.

[137] Fragkiskos D Malliaros, Christos Giatsidis, Apostolos N Papadopoulos, and Michalis Vazirgiannis. "The core decomposition of networks: Theory, algorithms and applications." In: *The VLDB Journal* 29.1 (2020), pp. 61–92.

[138] Fragkiskos D. Malliaros, Christos Giatsidis, Apostolos N. Papadopoulos, and Michalis Vazirgiannis. "The core decomposition of networks: theory, algorithms and applications." In: *VLDB J.* 29.1 (2020), pp. 61–92.

[139] Fragkiskos D Malliaros, Maria-Evgenia G Rossi, and Michalis Vazirgiannis. "Locating influential nodes in complex networks." In: *Scientific reports* 6.1 (2016), pp. 1–10.

[140] Fragkiskos Malliaros, Christos Giatsidis, Apostolos Papadopoulos, and Michalis Vazirgiannis. "The Core Decomposition of Networks: Theory, Algorithms and Applications." In: (2019).

[141] Tanwi Mallick, Prasanna Balaprakash, Eric Rask, and Jane Macfarlane. "Transfer Learning with Graph Neural Networks for Short-Term Highway Traffic Forecasting." In: *arXiv preprint arXiv:2004.08038* (2020).

[142] Sahil Manchanda, Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj Singh. "GCOMB: Learning Budget-constrained Combinatorial Algorithms over Billion-sized Graphs." In: *Advances in Neural Information Processing Systems* 33 (2020).

[143] Francesco Alessandro Massucci and Domingo Docampo. "Measuring the academic reputation through citation networks via PageRank." In: *Journal of Informetrics* 13.1 (2019), pp. 185–201.

[144] Neil Mathew, Stephen L Smith, and Steven L Waslander. "Planning paths for package delivery in heterogeneous multirobot teams." In: *IEEE Transactions on Automation Science and Engineering* 12.4 (2015), pp. 1298–1308.

[145] Christopher McCarty, James W Jawitz, Allison Hopkins, and Alex Goldman. "Predicting author h-index using characteristics of the co-author network." In: *Scientometrics* 96.2 (2013), pp. 467–483.

[146] Brendan D McKay and Adolfo Piperno. "Practical graph isomorphism, II." In: *Journal of symbolic computation* 60 (2014), pp. 94–112.

[147] Panagiotis T Metaxas and Eni Mustafaraj. "Social media and the elections." In: *Science* 338.6106 (2012), pp. 472–473.

[148] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In: *Advances in neural information processing systems* 26 (2013), pp. 3111–3119.

[149] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. "A graph placement methodology for fast chip design." In: *Nature* 594.7862 (2021), pp. 207–212.

[150] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning." In: *nature* 518.7540 (2015), pp. 529–533.

[151] Flaviano Morone and Hernán A Makse. "Influence maximization in complex networks through optimal percolation." In: *Nature* 524.7563 (2015), pp. 65–68.

[152] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. "Weisfeiler and leman go neural: Higher-order graph neural networks." In: *33rd AAAI Conference on Artificial Intelligence*. 2019, pp. 4602–4609.

[153] James Munkres. "Algorithms for the assignment and transportation problems." In: *Journal of the Society for Industrial and Applied Mathematics* 5.1 (1957), pp. 32–38.

[154] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. "An analysis of approximations for maximizing submodular set functions—I." In: *Mathematical programming* 14.1 (1978), pp. 265–294.

[155] Mark EJ Newman. "Assortative mixing in networks." In: *Physical review letters* 89.20 (2002), p. 208701.

[156] Mark EJ Newman. "The structure and function of complex networks." In: *SIAM review* 45.2 (2003), pp. 167–256.

[157] Hung T Nguyen, My T Thai, and Thang N Dinh. "Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks." In: *Proceedings of the 2016 international conference on management of data*. 2016, pp. 695–710.

[158] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. "Matching node embeddings for graph similarity." In: *Thirty-first AAAI conference on artificial intelligence*. 2017.

[159] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. "Graph kernels: A survey." In: *arXiv preprint arXiv:1904.12218* (2019).

[160] Giannis Nikolentzos, Antoine Jean-Pierre Tixier, and Michalis Vazirgiannis. "Message Passing Attention Networks for Document Understanding." In: *34th AAAI Conference on Artificial Intelligence*. 2020, pp. 8544–8551.

[161] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab, 1999.

[162] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning." In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[163] George Panagopoulos. "Multi-task learning for commercial brain computer interfaces." In: *BIBE*. 2017.

[164] George Panagopoulos, Fragkiskos D Malliaros, and Michalis Vazirgiannis. "Diffugreedy: An influence maximization algorithm based on diffusion cascades." In: *International Conference on Complex Networks and their Applications*. Springer. 2018, pp. 392–404.

[165] George Panagopoulos, Fragkiskos Malliaros, and Michalis Vazirgiannis. "Multi-task learning for influence estimation and maximization." In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[166] George Panagopoulos, George Tsatsaronis, and Iraklis Varlamis. "Detecting rising stars in dynamic collaborative networks." In: *Journal of Informetrics* 11.1 (2017), pp. 198–222.

[167] Austin J Parish, Kevin W Boyack, and John PA Ioannidis. "Dynamics of co-authorship and productivity across different fields of scientific research." In: *PloS one* 13.1 (2018), e0189742.

[168] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library." In: *Advances in Neural Information Processing Systems*. 2019, pp. 8026–8037.

[169] Sen Pei, Flaviano Morone, and Hernán A Makse. "Theories for influencer identification in complex networks." In: *Complex Spreading Phenomena in Social Systems*. Springer, 2018, pp. 125–148.

[170] Sen Pei, Jiannan Wang, Flaviano Morone, and Hernán A Makse. "Influencer identification in dynamical complex systems." In: *Journal of Complex Networks* 8.2 (2020), cnz029.

[171] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.

[172] *Political Advertising on Social Media Platforms*. https://www.americanbar.org/groups/crsj/publications/human_rights_magazine_home/voting-in-2020/political-advertising-on-social-media-platforms/. Accessed: 2021-08-20.

[173] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. "Explainability methods for graph convolutional neural networks." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10772–10781.

[174] Marcelo Prates, Pedro HC Avelar, Henrique Lemos, Luis C Lamb, and Moshe Y Vardi. "Learning to solve np-complete problems: A graph neural network for decision tsp." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 4731–4738.

[175] JZ Qiu, Jian Tang, Hao Ma, YX Dong, KS Wang, and J Tang. "DeepInf: Modeling influence locality in large social networks." In: *KDD*. 2018.

[176] Henry L Roediger III. "The h index in science: A new measure of scholarly contribution." In: *APS Observer* 19.4 (2006).

[177] Maria Evgenia G Rossi and Michalis Vazirgiannis. "Exploring Network Centralities in Spreading Processes." In: *International Symposium on Web AlGorithms (iSWAG)*. 2016.

[178] David E Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. "Backpropagation: The basic theory." In: *Backpropagation: Theory, architectures and applications* (1995), pp. 1–34.

[179] Kazumi Saito, Masahiro Kimura, Kouzou Ohara, and Hiroshi Motoda. "Learning continuous-time information diffusion model for social behavioral data analysis." In: *ACML*. 2009.

[180] Emre Sarigöl, René Pfitzner, Ingo Scholtes, Antonios Garas, and Frank Schweitzer. "Predicting scientific success based on coauthorship networks." In: *EPJ Data Science* 3.1 (2014), p. 9.

[181] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. "Collective classification in network data." In: *AI magazine* 29.3 (2008), pp. 93–93.

[182] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. "Structured sequence modeling with graph convolutional recurrent networks." In: *International Conference on Neural Information Processing*. Springer. 2018, pp. 362–373.

[183] Sandra Servia-Rodríguez, Anastasios Noulas, Cecilia Mascolo, Ana Fernández-Vilas, and Rebeca P Díaz-Redondo. "The evolution of your success lies at the centre of your co-authorship network." In: *PloS one* 10.3 (2015), e0114302.

[184] Cosma Rohilla Shalizi and Andrew C Thomas. "Homophily and contagion are generically confounded in observational social network studies." In: *Sociological methods & research* 40.2 (2011), pp. 211–239.

[185] Zhihong Shen, Hao Ma, and Kuansan Wang. "A Web-scale system for scientific knowledge exploration." In: *arXiv preprint arXiv:1805.12216* (2018).

[186] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. "Deterministic policy gradient algorithms." In: *International conference on machine learning*. PMLR. 2014, pp. 387–395.

[187] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. "An overview of microsoft academic service (mas) and applications." In: *Proceedings of the 24th International Conference on World Wide Web*. 2015, pp. 243–246.

[188] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang. "An overview of microsoft academic service (mas) and applications." In: *International conference on World Wide Web (The WebConf)*. 2015.

[189] *Social Network Usage Growth Statistics: How Many People Use Social Media in 2021?* https://backlinko.com/social-media-users. Accessed: 2021-08-20.

[190] David Soriano-Panos, Gourab Ghoshal, Alex Arenas, and Jesús Gómez-Gardenes. "Impact of temporal scales and recurrent mobility patterns on the unfolding of epidemics." In: *Journal of Statistical Mechanics: Theory and Experiment* 2020.2 (2020), p. 024006.

[191] Lichao Sun, Weiran Huang, Philip S Yu, and Wei Chen. "Multi-round influence maximization." In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 2249–2258.

[192] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[193] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. "Social influence analysis in large-scale networks." In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 807–816.

[194] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. "Arnetminer: extraction and mining of academic social networks." In: *Knowledge Discovery and Data mining (KDD)*. 2008, pp. 990–998.

[195] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. "Transferring Robustness for Graph Neural Network Against Poisoning Attacks." In: *13th International Conference on Web Search and Data Mining*. 2020, pp. 600–608.

[196] Youze Tang, Yanchen Shi, and Xiaokui Xiao. "Influence maximization in near-linear time: A martingale approach." In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015, pp. 1539–1554.

[197] Youze Tang, Xiaokui Xiao, and Yanchen Shi. "Influence maximization: Near-optimal time complexity meets practical efficiency." In: *International Conference on Management of Data (SIGMOD)*. 2014, pp. 75–86.

[198] Youze Tang, Xiaokui Xiao, and Yanchen Shi. "Influence maximization: Near-optimal time complexity meets practical efficiency." In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 2014, pp. 75–86.

[199] Shan Tian, Songsong Mo, Liwei Wang, and Zhiyong Peng. "Deep reinforcement learning-based approach to tackle topic-aware influence maximization." In: *Data Science and Engineering* 5.1 (2020), pp. 1–11.

[200] Robert Tibshirani. "Regression shrinkage and selection via the lasso: a retrospective." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.3 (2011), pp. 273–282.

[201] Nora Touati-Moungla and Vincent Jost. "Combinatorial optimization for electric vehicles management." In: *Journal of Energy and Power Engineering* 6.5 (2012), pp. 738–743.

[202] Sijing Tu, Cigdem Aslay, and Aristides Gionis. "Co-exposure maximization in online social networks." In: *Advances in Neural Information Processing Systems* 33 (2020).

[203] Muluneh Mekonnen Tulu, Ronghui Hou, and Talha Younas. "Identifying influential nodes based on community structure to speed up the dissemination of information in complex network." In: *IEEE Access* 6 (2018), pp. 7390–7401.

[204] Marco Valenzuela, Vu Ha, and Oren Etzioni. "Identifying meaningful citations." In: *Workshops at the twenty-ninth AAAI conference on artificial intelligence*. 2015.

[205] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[206] Joaquin Vanschoren. "Meta-learning: A survey." In: *arXiv preprint arXiv:1810.03548* (2018).

[207] Sharan Vaswani, Branislav Kveton, Zheng Wen, Mohammad Ghavamzadeh, Laks VS Lakshmanan, and Mark Schmidt. "Model-independent online learning for influence maximization." In: *ICML*. 2017.

[208] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. "Graph attention networks." In: *International Conference on Learning Representations*. 2018.

[209] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. "Pointer networks." In: *arXiv preprint arXiv:1506.03134* (2015).

[210] Soroush Vosoughi, Deb Roy, and Sinan Aral. "The spread of true and false news online." In: *Science* 359.6380 (2018), pp. 1146–1151.

[211] Ludo Waltman and Nees Jan Van Eck. "The inconsistency of the h-index." In: *Journal of the American Society for Information Science and Technology* 63.2 (2012), pp. 406–415.

[212] Chi Wang, Wei Chen, and Yajun Wang. "Scalable influence maximization for independent cascade model in large-scale social networks." In: *Data Mining and Knowledge Discovery* 25.3 (2012), pp. 545–576.

[213] Jia Wang, Vincent W Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. "Topological recurrent neural network for diffusion prediction." In: *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2017, pp. 475–484.

[214] Jia Wang, Vincent W Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. "Topological recurrent neural network for diffusion prediction." In: *2017 IEEE International Conference on Data Mining (ICDM)*. 2017, pp. 475–484.

[215] Yongqing Wang, Huawei Shen, Shenghua Liu, Jinhua Gao, and Xueqi Cheng. "Cascade dynamics modeling with attention-based recurrent neural network." In: *IJCAI*. 2017.

[216] Zhitao Wang and Wenjie Li. "Hierarchical diffusion attention network." In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press. 2019, pp. 3828–3834.

[217] Christopher J. C. H. Watkins and Peter Dayan. "Q-learning." In: *Machine Learning*. 1992, pp. 279–292.

[218] Luca Weihs and Oren Etzioni. "Learning to predict citation-based impact measures." In: *2017 ACM/IEEE Joint Conference on Digital Libraries*. 2017, pp. 1–10.

[219] Zheng Wen, Branislav Kveton, Michal Valko, and Sharan Vaswani. "Online influence maximization under independent cascade model with semi-bandit feedback." In: *NeurIPS*. 2017.

[220] *Why fake news on social media travels faster than the truth.* https://www.theguardian.com/commentisfree/2018/mar/19/fake-news-social-media-twitter-mit-journalism. Accessed: 2021-08-20.

[221] Bryan Wilder, Han-Ching Ou, Kayla de la Haye, and Milind Tambe. "Optimizing Network Structure for Preventative Health." In: *AAMAS*. 2018, pp. 841–849.

[222] Bryan Wilder, Amulya Yadav, Nicole Immorlica, Eric Rice, and Milind Tambe. "Uncharted but not Uninfluenced: Influence Maximization with an Uncertain Network." In: *AAMAS*. Vol. 17. 2017, pp. 1305–1313.

[223] Lorna Wildgaard, Jesper W Schneider, and Birger Larsen. "A review of the characteristics of 108 author-level bibliometric indicators." In: *Scientometrics* 101.1 (2014), pp. 125–158.

[224] Qingyun Wu, Zhige Li, Huazheng Wang, Wei Chen, and Hongning Wang. "Factorization bandits for online influence maximization." In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 636–646.

[225] Miao Xie, Qiusong Yang, Qing Wang, Gao Cong, and Gerard De Melo. "DynaDiffuse: A Dynamic Diffusion Model for Continuous Time Constrained Influence Maximization." In: *AAAI*. 2015, pp. 346–352.

[226] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How powerful are graph neural networks?" In: *International Conference on Learning Representations*. 2019.

[227] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation Learning on Graphs with Jumping Knowledge Networks." In: *Proceedings of the 37th International Conference on Machine Learning*. 2018, pp. 5453–5462.

[228] Erjia Yan and Ying Ding. "Applying centrality measures to impact analysis: A coauthorship network analysis." In: *Journal of the American Society for Information Science and Technology* 60.10 (2009), pp. 2107–2118.

[229] Erjia Yan and Raf Guns. "Predicting and recommending collaborations: An author-, institution-, and country-level analysis." In: *Journal of Informetrics* 8.2 (2014), pp. 295–309.

[230] Cheng Yang, Jian Tang, Maosong Sun, Ganqu Cui, and Zhiyuan Liu. "Multi-scale Information Diffusion Prediction with Reinforced Recurrent Networks." In: *IJCAI*. 2019, pp. 4033–4039.

[231] Cheng Yang, Jian Tang, Maosong Sun, Ganqu Cui, and Zhiyuan Liu. "Multi-scale information diffusion prediction with reinforced recurrent networks." In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press. 2019, pp. 4033–4039.

[232] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Zhenhui Li, Jieping Ye, and Didi Chuxing. "Deep multi-view spatial-temporal network for taxi demand prediction." In: *32nd AAAI Conference on Artificial Intelligence*. 2018, pp. 2588–2595.

[233] Huaxiu Yao, Chuxu Zhang, Ying Wei, Meng Jiang, Suhang Wang, Junzhou Huang, Nitesh V Chawla, and Zhenhui Li. "Graph few-shot learning via knowledge transfer." In: *arXiv preprint arXiv:1910.03053* (2019).

[234] Abdelhafid Zeroual, Fouzi Harrou, Abdelkader Dairi, and Ying Sun. "Deep Learning Methods for Forecasting COVID-19 Time-Series Data: A Comparative Study." In: *Chaos, Solitons & Fractals* (2020), p. 110121.

[235] Chun-Ting Zhang. "The e-index, complementing the h-index for excess citations." In: *PLoS One* 4.5 (2009), e5429.

[236] Jing Zhang, Biao Liu, Jie Tang, Ting Chen, and Juanzi Li. "Social Influence Locality for Modeling Retweeting Behaviors." In: *IJCAI*. Vol. 13. 2013, pp. 2761–2767.

[237] Kaichen Zhang, Jingbo Zhou, Donglai Tao, Panagiotis Karras, Qing Li, and Hui Xiong. "Geodemographic influence maximization." In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 2764–2774.

[238] Yuan Zhang, Tianshu Lyu, and Yan Zhang. "COSINE: Community-Preserving Social Network Embedding From Information Diffusion Cascades." In: *AAAI Conference on Artificial Intelligence*. 2018, pp. 2620–2627.

[239] Yutao Zhang, Fanjin Zhang, Peiran Yao, and Jie Tang. "Name Disambiguation in AMiner: Clustering, Maintenance, and Human in the Loop." In: *Knowledge Discovery & Data Mining (KDD)*. 2018, pp. 1002–1011.

[240] Chuan Zhou, Peng Zhang, Wenyu Zang, and Li Guo. "On the upper bounds of spread for greedy algorithms in social network influence maximization." In: *IEEE Transactions on Knowledge and Data Engineering* 27.10 (2015), pp. 2770–2783.

[241] Michel Zitt and Henry Small. "Modifying the journal impact factor by fractional citation weighting: The audience factor." In: *Journal of the American Society for Information Science and technology* 59.11 (2008), pp. 1856–1860.

[242] Bin Zou, Vasileios Lampos, and Ingemar Cox. "Transfer Learning for Unsupervised Influenza-like Illness Models from Online Search Data." In: *2019 World Wide Web Conference*. 2019, pp. 2505–2516.

[243] Difan Zou, Lingxiao Wang, Pan Xu, Jinghui Chen, Weitong Zhang, and Quanquan Gu. "Epidemic model guided machine learning for COVID-19 forecasts in the United States." In: *medRxiv* (2020).

[244] Harriet Zuckerman. "Nobel laureates in science: Patterns of productivity, collaboration, and authorship." In: *American Sociological Review* (1967), pp. 391–403.

ECOLE DOCTORALE

Title : Learning Influence Representations : Methods and Applications

Keywords : Graph Neural Networks, Combinatorial Optimization, Social Networks

Abstract : As social networks become an increasing part of our lives, the impact of online influence in the real world is increasing dramatically. In this thesis, we address different problems pertaining to various versions of social influence using neural networks. The first half of the thesis is devoted to influence maximization, an NP-hard combinatorial optimization problem. We propose a method to merge traditional methods with complementary data using neural representations. Moreover, we develop algorithms based on deep learning and simulation models, that exhibit competitive performance with state of the art. In the second half of the thesis, we focused on applications of influence. We approached epidemic forecasting with graph learning, using the history of the disease spread and mass mobility temporal networks, resulting in clear improvement over competing benchmarks. Finally, we provide a set of data mining methods to quantify, predict and visualize the scientific success of an author.

Title : Apprentissage des représentations d'influence : Méthodes et Applications

Keywords : Réseaux de neurones graphiques, Optimisation combinatoire, Réseaux sociaux

Abstract : Alors que les réseaux sociaux font de plus en plus partie de nos vies, l'impact de l'influence en ligne dans le monde réel augmente considérablement. Dans cette thèse, nous abordons différents problèmes relatifs à diverses versions de l'influence sociale en utilisant des réseaux de neurones. La première moitié de la thèse est consacrée à la maximisation de l'influence, un problème d'optimisation combinatoire NP-hard. Nous proposons une méthode pour fusionner les méthodes traditionnelles avec des données complémentaires en utilisant des représentations neuronales. De plus, nous développons des algorithmes basés sur l'apprentissage profond et des modèles de simulation, qui présentent des performances compétitives par rapport à l'état de l'art. Dans la seconde moitié de la thèse, nous nous sommes concentrés sur les applications d'influence. Nous avons abordé la prévision des épidémies avec l'apprentissage par graphe, en utilisant l'historique de la propagation de la maladie et les réseaux temporels de mobilité de masse, ce qui a entraîné une nette amélioration par rapport aux références concurrentes. Enfin, nous fournissons un ensemble de méthodes d'exploration de données pour quantifier, prédire et visualiser le succès scientifique d'un auteur.

Institut Polytechnique de Paris
91120 Palaiseau, France