FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Temporal Link Prediction Using Graph Pattern Matching

PhD THESIS

submitted in partial fulfillment of the requirements for the degree of

Doctor of Technical Sciences

within the

Vienna PhD School of Informatics

by

Nataliia Rümmele
Registration Number 1028250

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Univ.Prof. Dr. Hannes Werthner

External reviewers:
Assoc.Prof. Dr. Alessandro Provetti. University of Messina, Italy.
Univ.Prof. i.R. Dr. Wilfried Grossmann. Vienna University, Austria.

Vienna, 30th April, 2015

Nataliia Rümmele	Hannes Werthner

Declaration of Authorship

Nataliia Rümmele
Novaragasse 33/2/6, 1020 Vienna

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Vienna, 30th April, 2015

Nataliia Rümmele

Acknowledgements

Science is a collaborative effort. Therefore, I would like to thank my co-authors for the productive and enjoyable joint work, the reviewers for their valuable comments, and all the scientists who I have met during my PhD studies, who have shared their ideas and provided feedback about my research.

In particular, I want to thank my advisor Hannes Werthner. I am deeply grateful for his guidance, support and inspiration. I especially want to thank him for giving me the freedom to pursue my own research interests. Thanks to his patience and efforts, I feel well prepared for the upcoming challenges of a research career.

Furthermore, I want to thank Ryutaro Ichise for hosting me in Tokyo as an intern in his group. Thanks to this opportunity, I have not only worked on interesting problems with him and his students, but I have also experienced the Japanese culture and great cuisine. I would like to particularly acknowledge Lankeshwara Munasinghe, a former PhD student of Ryutaro Ichise, for raising my awareness about the problem of link prediction in heterogeneous networks.

A special gratitude goes to my colleagues Julia Neidhardt and Mamen Calatrava Moreno for making my workplace such a great place to be. I am grateful to Julia for insightful discussions about statistics and network analysis.

I feel greatly indebted to my husband Stefan Rümmele for his valuable comments and remarks on my thesis and on my research work. He has provided a constant support during my studies and a shoulder to cry on when things did not go as expected.

Finally, I want to thank my family and friends for their patience and understanding whenever I had to prioritize my work over them. I owe my deepest gratitude to my mother Iryna Pobiedina. She has always supported my interest in formal sciences, and has given me a chance to pursue a PhD in technical sciences, an opportunity which her parents thought to be inappropriate for a young woman. I would also like to express my gratitude to Nikolai Pikhtar for sparking and nourishing my interest in mathematics, and to my English teacher Tamara Mironova who supervised my first academic research.

This thesis was supported by the Vienna PhD School of Informatics. Thanks to the administration and students of the Vienna PhD School of Informatics and the Electronic Commerce group, my scientific life and my stay in Austria were such a great experience.

I dedicate this thesis to Stefan and my family.

Abstract

Many real-world systems are composed of individual components which are inter-linked in some way. To understand and predict the behavior of such a system, we can model it as a network where components correspond to nodes and connections are links between the nodes. Real-world networks are highly dynamic: new nodes and links appear permanently while the existing ones can vanish, and various attributes for nodes and links may change. One of these changes, and the focus of this thesis, is the creation of new links.

In the field of network analysis the task of predicting the appearance of a link between two nodes in the network is known as *link prediction*. In the classical setting, we assume that there is no direct link between these nodes, and they already exist in the network. An established approach to build a network evolution model is to use an observed property of the global network structure. However, we follow another approach: by identifying the patterns of node connections, the so-called *graph patterns*, we investigate how these patterns that are observable at the current time can help us predict the future link formation in the network. So, temporal information is of great importance to us, hence, we refer to the problem at hand as *temporal link prediction*. The considered graph patterns can, first of all, vary in their size (either in terms of the number of considered nodes or links), and secondly, they can include various additional information, such as attributes of nodes and links, or different types of connections between nodes. Hence, we can capture on a very fine grained level the evolution of the network over time. Another advantage is that we can overcome the problem of predicting links for a node which does not yet exist in the network. Such problem is often referred to as a cold start problem.

The goal of this thesis is to advance the state-of-the-art of graph pattern based temporal link prediction. More specifically, we focus on the following three important aspects. The first goal is to improve the quality of prediction. Secondly, we focus on the efficiency of the involved computations. The third goal is to widen the field of application domains where link prediction techniques have not been used before. The three main contributions of this thesis are: (i) We introduce a new feature which allows us to apply link prediction techniques to solve the citation count prediction problem. This is a known problem in the area of bibliometrics, but it has never been perceived and studied before as a link prediction problem. (ii) We build time and heterogeneity scores which we use to predict links in heterogeneous networks over time, a new promising domain in the link prediction community. (iii) We design a benchmark of database systems for graph pattern matching which provides us guidelines to choose the best tool to implement the prediction methods above.

Contents

Abstract **vii**

Contents **ix**

1 Introduction **1**
1.1 Motivation . . . 2
1.2 Problem Statement . . . 3
1.3 Methodology . . . 4
1.4 Main Results . . . 5
1.5 Structure of the Work . . . 10

2 Preliminaries **13**
2.1 Networks . . . 13
2.2 Link prediction . . . 19
2.3 Graph pattern matching . . . 22

3 State of the Art **25**
3.1 Unsupervised learning approach . . . 27
3.2 Learning-based approach . . . 33
3.3 Evaluation of Link Prediction . . . 35
3.4 Link Prediction Problems . . . 36
3.5 Link Prediction Applications . . . 38

4 Citation Count Prediction **39**
4.1 Motivation . . . 39
4.2 Problem . . . 41
4.3 Related Work . . . 41
4.4 GERscore . . . 45
4.5 Experimental Results . . . 49
4.6 Summary and Discussion . . . 60

5 Heterogeneous Networks **63**
5.1 Motivation . . . 63
5.2 Problem . . . 65

5.3 Related Work . . . 65
5.4 Time and Heterogeneity based scores . . . 67
5.5 Experimental Results . . . 70
5.6 Summary and Discussion . . . 78

6 Tools for Graph Pattern Matching **83**
6.1 Motivation . . . 83
6.2 Problem . . . 85
6.3 Related Work . . . 85
6.4 Benchmark for Graph Pattern Matching . . . 87
6.5 Experimental Results . . . 97
6.6 Summary and Discussion . . . 103

7 Conclusion **105**
7.1 Summary . . . 105
7.2 Future Work and Open Issues . . . 107

List of Publications **111**

List of Figures **113**

List of Tables **115**

Bibliography **117**

Acronyms **127**

CHAPTER 1

Introduction

> The Future . . . is like a puzzle – with missing pieces – difficult to read, and never, NEVER, what you think.
>
> *Edward Kitsis and Adam Horowitz*, in lines for "Rumpelstiltskin in Manhattan", episode 2.14 of Once Upon a Time

Many real-world systems are composed of a group of individual components which are inter-linked in some way. For example, the Web is a set of web-pages with hyperlinks pointing from one web-page to another, the Internet is a collection of computers linked by data connections, and human societies can be seen as a group of people connected by acquaintance or social interaction. There are various aspects which can be studied within such systems in order to understand them and to predict their behavior [89]. Firstly, one can focus on the nature of individual components, for instance, studying the work of computers or different characteristics of people. Secondly, the nature of connections between these components can be investigated, for instance, the communication protocols of the Internet or the dynamics of human relationships. The third aspect is the pattern of connections between components, which can be regarded as the inter-play between the nature of the components and their connections, and which affects the behavior of the whole system.

To study the pattern of connections within the system, we can model a network with components as nodes in this network and connections as links between the corresponding nodes. It can be argued that a network simplifies the complexity of connection patterns [89]. Though a variety of tools and theories to understand, analyze and model networks have been already developed in different fields of science (such as computer

science, mathematics, social sciences, physics), there are many open issues. The focus of this thesis is to extend the knowledge base in the area of network analysis.

1.1 Motivation

Networks are one of the most generic data structures and therefore used to model data in various application areas. For example, friendship networks capture human relationships, collaboration networks model the team work of a group of people with a specific task, citation networks represent corpora of scientific publications, and the Web itself can be modeled as a network with nodes as pages and directed links as hyperlinks between the pages. The availability of large scale data on the Web provides an opportunity to examine such interactions with the goal to uncover many interesting aspects of the modern world. The approaches to accomplish this goal can be roughly grouped into those which study static properties and those which study dynamic properties of the corresponding networks. The results of such studies bring further insights into the areas of human relationships, composition of groups for collaboration [90], marketing [69], the flow of information on the Web [70] and many others.

As mentioned, on one hand, we can focus on static properties of real-world networks. Backstrom et al. provide evidence for the "small-world" structure of Facebook, a social network, by showing that people have on average only four degrees of separation from each other [11]. Barabasi et al. illustrate that the distribution of node degrees for the Web graph follows the power-law distribution [13]. Another property of social and biological networks, namely community structure, has been shown by Girvan and Newman [51]. This property means that nodes in such networks tend to be grouped in tightly knit clusters with looser connections between different clusters.

On the other hand, we can investigate dynamic properties of real-world networks. This is a big challenge: many real-world systems which are modeled as networks are highly dynamic with new components and connections appearing permanently. There have been numerous studies about the dynamic properties of networks at a global level, like the shrinking diameter over time and the super-linear growth of links with regard to the number of nodes [72]. Numerous models for link formation in networks have also been proposed. An established approach is to build a network evolution model based on an observed property of the global network structure, for example, small diameter or power law node degree distribution. However, in the recent years research is increasingly focusing on the dynamics of networks at a microscopic level.

Leskovec et al. studied several network formation strategies [71]. Unlike the top-down approach to modeling network evolution, the authors study the node arrival process, the edge initiation process and the edge destination selection process. By employing a likelihood-based approach they construct a network evolution model which takes into consideration the observed processes. The obtained model is a variation of a triad-closing model, meaning that connected three node patterns are considered in link formation process. The authors point out that their approach can be extended to the cases when nodes are more than two hops apart, however, it becomes computationally infeasible to

compute the required likelihoods at a distance greater than two hops. Nevertheless, new network evolution models, which consider more complex patterns of node connections, are being suggested and shown to capture link formation processes better [21, 31, 76]. These patterns are not only bigger in size (either in terms of the number of considered nodes or links), but also capture more additional information: various attributes of nodes and links, different types of connections between nodes, etc.

We study patterns of node connections (i.e., *graph patterns*) at a local level and intend to illustrate how these patterns can help us predict network evolution over time. In this thesis we consider only one aspect of network evolution, namely the formation of new links over time. Within the field of network analysis this problem is called *link prediction*. Thereby the task is to predict the appearance of a link between two nodes in the network. In the classical setting of the link prediction problem, we assume that there is no direct link between the considered nodes, and both nodes already exist in the network, meaning that we have sufficient knowledge about them. Unlike the classical setting, temporal information is of great importance to us since we want to predict links which will form in the future. Therefore, we put a stress on the temporal aspect by referring to the problem at hand as *temporal link prediction*.

1.2 Problem Statement

The goal of this thesis is to advance the state-of-the-art of graph pattern based temporal link prediction. Thereby we focus on the following three important aspects:

- Quality of prediction: The accuracy of a prediction method is arguably the most important factor. Hence, one goal is to increase this accuracy.
- Efficiency of the involved computations: High prediction accuracy is meaningless in practice if the involved methods can not be computed in a reasonable amount of time. Hence, the second goal is to lower run-time of the involved algorithms.
- Application areas: The third goal is to widen the field of possible applications. Often the gravity of the problem is illustrated by the amount of domains where it arises. Here, we want to show that we can apply link prediction in areas where it was not used before.

We intend to show that graph pattern based link prediction methods allow us to solve new interesting problems and to improve the prediction in the known scenarios.

Main Research Question: How can we use graph pattern matching to study temporal evolution of networks?

We study the temporal evolution of networks because real-world networks are highly dynamic. In general the structure of such networks permanently changes. One of these changes, and the focus of our work, is the creation of new links. In our setting, we do not consider how new nodes join the network and we also disregard the processes of link disappearance. Graph patterns correspond to the patterns of node connections at a

microscopic level. We are interested to identify how these patterns that are observable at the current time can help us predict the future link formation in the network.

Research question 1: How can we predict citation counts for academic publications using link prediction methods?

Thousands of new publications appear yearly and we need to find the relevant ones among them. One metric that is used in order to navigate this corpus of literature is the citation count. The drawback of this metric is that it is not available for very recent publications. Thus, predicting the future citation count reliably would be very useful. Since publications form a citation network based on who cites whom, we want to investigate how link prediction methods can solve this problem. Hence, citation count prediction is a novel application for link prediction.

Research question 2: Can we improve temporal link prediction in heterogeneous networks?

Networks with just one type of link between nodes are often not enough to faithfully model the relationships between entities. For example, in a network of authors we might want to have one type of link indicating co-authorship and another type of link to represent that one author cited the other one. Hence, it is important to generalize temporal link prediction methods in order to make them applicable to heterogeneous networks. We focus on heterogeneity of links and assume that nodes are of the same type.

Research question 3: Which tools can be used to efficiently solve graph pattern matching problems?

The main bottleneck of graph pattern based temporal link prediction is to identify graph patterns. Therefore, it is crucial to solve this sub-task of graph pattern matching as fast as possible. This NP-complete problem is well studied in theoretical computer science, but in practice it is not clear which tool is the best to manage the data efficiently in order to address the graph pattern matching problem. Moreover, we require data structures which can model complex networks with temporal information, different attributes for links and nodes. Hence, we need to evaluate and compare different tools which is an important first step in speeding up the whole graph pattern based link prediction method.

1.3 Methodology

As a research methodology, we apply the design science research framework to ensure a rigorous and relevant contribution to the Computer Science community [55]. Figure 1.1 shows the general framework for information system research which is a part of the design science methodology. Within this framework, the research is driven by business needs which can come from people, organizations or technology. These needs ensure the relevance of the conducted research. Based on the needs, researchers develop artifacts

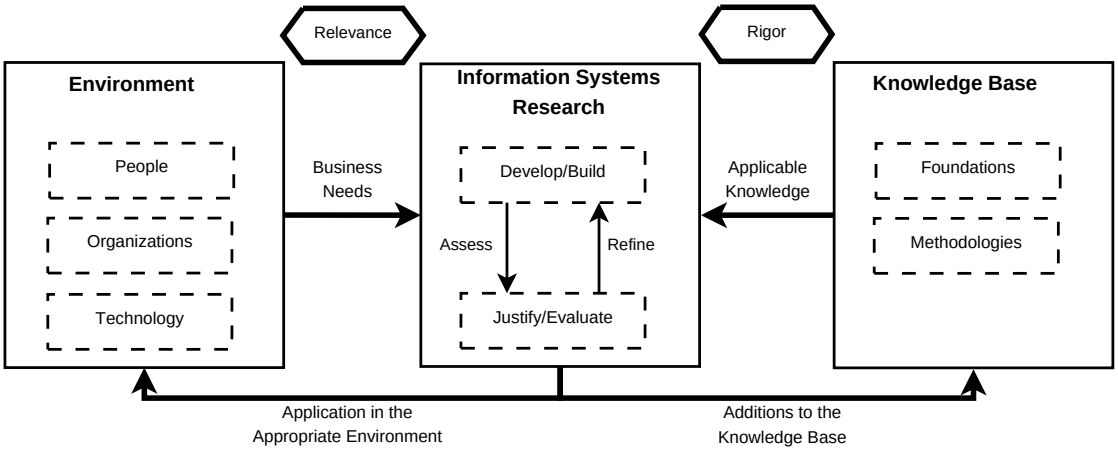


Figure 1.1: Information Systems Research Framework [55].

using existing foundations and methodologies which ensure the rigor of the research. The artifacts in the sense of the design science paradigm include instantiations (for example, systems, products), constructs (for example, concepts), models (for example, representations) and methods (for example, algorithms, techniques). These artifacts are built iteratively by assessing and refining their utility. As assessment methods, researchers employ case studies, experiments, simulations, etc. The obtained artifacts are applied to meet the business needs within appropriate environment. Additionally, they are added to the knowledge base for future research applications.

The research within this thesis is motivated by the need to understand and predict temporal evolution of networks. Figure 1.2 depicts the design science process applied during our research. To answer the main research question, we work on three research questions presented in 1.2. The work on the first two questions is conducted sequentially. However, the needs uncovered when solving the problems within these two questions motivated the work on the third question. The obtained results of the work on the third question can be applied to refine solutions for the first two questions.

We develop artifacts for each research question iteratively by conducting experiments within appropriate environment. When the artifact satisfies requirements, the development process ends and the obtained artifact is added to the knowledge base. In the following, we describe in detail the resulting artifacts which constitute the main contributions of this thesis.

1.4 Main Results

The three main contributions of this thesis are:

1. We introduce a new feature, called GERscore, which allows us to apply link prediction techniques to solve the citation count prediction problem.

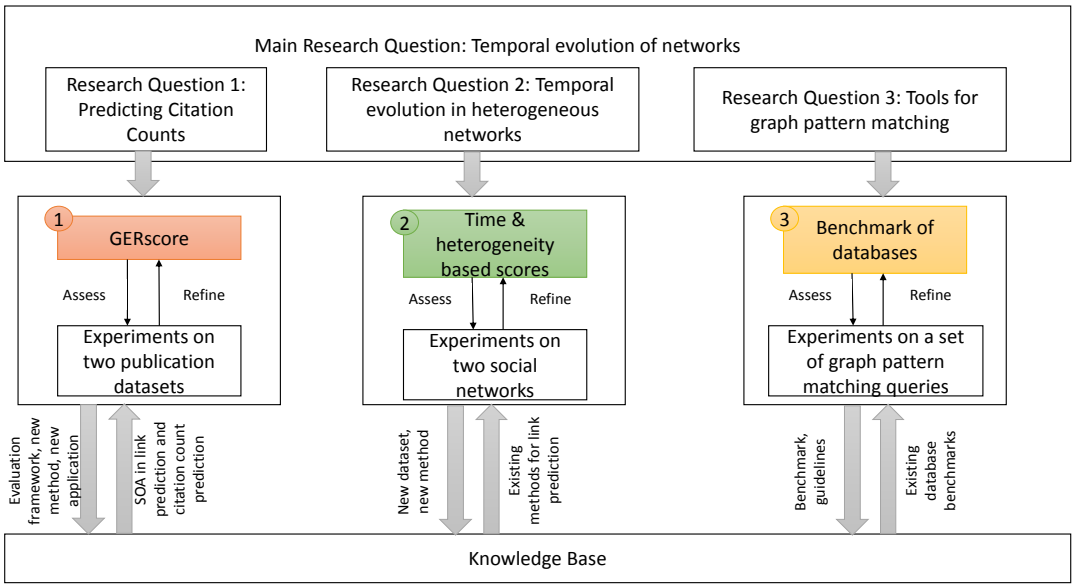


Figure 1.2: Design Science Framework of this thesis.

2. We build time and heterogeneity scores which we use to predict links in heterogeneous networks over time.

3. We design a benchmark of database systems for graph pattern matching which provides us guidelines to choose the best tool to implement the prediction methods above.

We define the maturity of these contributions based on the knowledge contribution framework outlined in the design science research methodology [55]. This framework divides artifacts into four quadrants according to two dimensions: the maturity of the application domain (whether the problem is known or new) and the maturity of solutions, or artifacts. The four quadrants are:

- *Routine design*: the contributions in this quadrant are obtained by solving known problems with existing solutions.
- *Inspiration*: the results here are obtained by solving known problems with new solutions.
- *Exaptation*: these contributions extend or adapt known solutions to new problems.
- *Invention*: the artifacts in this quadrant introduce new problems and suggest unknown solutions for them.

In Figure 1.3 we classify the contributions of this thesis according to the quadrants explained above.

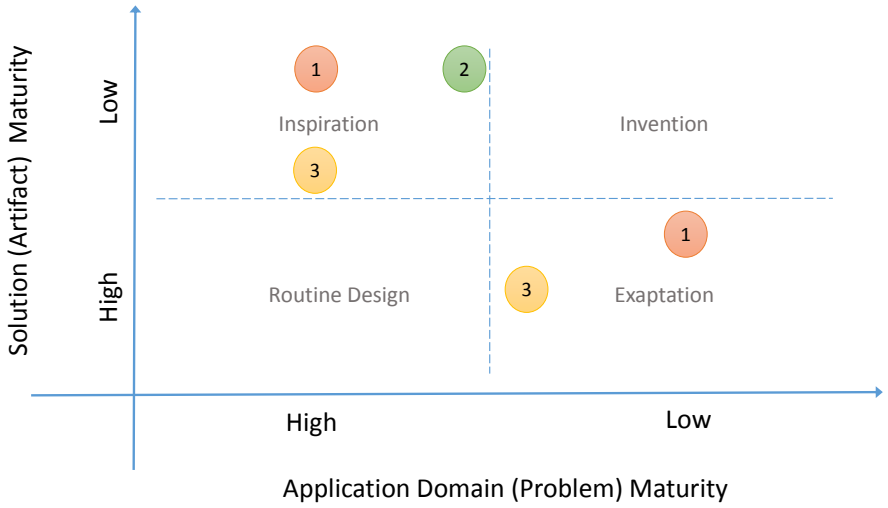


Figure 1.3: Classification of our artifacts according to Design Science framework [55]

1.4.1 Citation Count Prediction

The work in this part is directed to answer the first research question mentioned in Section 1.2.

Challenges. It is a major challenge in academia to identify important literature among recent publications due to the drastic growth of the amount of scientific publications each year. An accurate estimation of the future citation count can be used to facilitate the search for promising publications. Previous work points out that graph mining techniques lead to good results [80]. This observation motivated us to formulate the citation count prediction task as a variation of the link prediction problem in the citation network. A node in this network corresponds to a scientific publication, and a directed link between two nodes points from the paper, which references, to the paper which is cited. Here the citation count of a paper is equal to its in-degree in the network, in other words the number of links which point to it. Its out-degree, that is the number of outgoing links, corresponds to the number of references. Since out-degree remains the same over time, the appearance of a new link means that the citation count of the corresponding paper increases. In the link prediction problem we aim at predicting the appearance of links in the network. However, we cannot apply classical link prediction methods for several reasons. Firstly, we do not solve the standard link prediction problem since we need to estimate only the amount of new links for a specific node, but not with which other nodes in the network it gets connected. Secondly, the majority of link prediction methods neglect the temporal aspect. Since we intend to predict the citation counts

in the future, we want to capture the temporal evolution of the citation network with our link prediction method. Thirdly, classical link prediction methods can predict links only between nodes which already exist in the network. In our case, the new link forms between an existing node and a node which will appear in the network only in the future, hence, we do not possess any knowledge about such a node. This is also known as the *cold start problem*.

By examining the state-of-the-art regarding the citation count prediction problem, we have encountered another issue. Different approaches to evaluate the solution methods for this problem are used in the previous works. Hence, there exists no unified evaluation framework to compare the various solution methods and their performance.

Contributions. Firstly, we demonstrate how to formulate the citation count prediction problem as a link prediction problem. We are the first to suggest such a formulation, thus, extending the applicability of link prediction to a new domain. Secondly, we adopt score calculation based on the graph evolution rules of Bringmann et al. [21] to introduce a new feature GERscore for solving the citation count prediction problem. We also propose a new score calculation. The mentioned graph evolution rules are a special type of graph patterns which capture network evolution over time at a microscopic level. Thirdly, we design an extended evaluation framework which we apply not only to the new feature, but also to several state-of-the-art features. In such a way, we ensure a robust and extensive comparison of various approaches and their performance to solve the citation count prediction problem.

The contribution of this work to the link prediction community can be classified via the *exaptation* quadrant in Figure 1.3, since our new feature GERscore is an adaptation and extension of a known method to a novel problem. Link prediction methods based on graph patterns have been invented recently and present solutions with low maturity, hence, we put this contribution closer to the border with the *invention* quadrant. On the other hand, within the area of bibliometrics, a special field of information and library sciences which focuses on the statistical analysis of academic literature, our contribution can be regarded as applying a new method to a known problem. Therefore, the contribution to the bibliometrics field can be classified also via the *inspiration* quadrant in Figure 1.3.

1.4.2 Heterogeneous Networks

Challenges. Heterogeneity can be frequently observed in real-world networks. Classical methods for link prediction in homogenous networks cannot fully capture the complex structure in such scenarios. Hereby, the two easiest ways to adapt these methods to a heterogeneous setting are either to treat all link types equally or to consider only one link type and disregard the others. Either way we might lose valuable information which becomes even more critical in the light of sparse nature of many social networks.

On the other hand, temporal aspects and heterogeneity of node connections have often been omitted due to insufficient data [31]. In many cases temporal information as well as full information about nodes and links is missing. For example, different attributes

and types of nodes and links or presence of the links might not be available either due to privacy reasons or corresponding data is simply not collected by the system. Very often such shortcoming results in the oversimplification of the network evolution models.

Another challenge is graph pattern matching within heterogeneous networks which have node and link labels. Many tools to solve this problem in homogeneous networks are available, but very few can deal with complex networks.

Contributions. Firstly, we collect new data to study evolution of an online gaming community. By using a Web API provided by the gaming company Valve, we collect data about the gaming experience and friendship within an online community. We also take a dataset of scientific publications from our work on the citation count prediction problem (see Section 1.4.1) and create a collaboration network based on it. To this end, we create a node for each author and introduce two types of links between them. One indicates that two authors worked together, and the other link type shows that two authors are peers. Thus, we have two heterogeneous networks and their temporal evolution over a certain time period. The second contribution is the study of the performance of several supervised methods for temporal link prediction in heterogeneous social networks at several time points. The third contribution are new methods for temporal link prediction in heterogeneous networks. While graph pattern based link prediction methods for heterogeneous networks exist [31], they do not take into account the temporal aspect. Hence, we design three new methods inspired by the technique of Davis et al. [31]. Their technique is based on counting all possible 3-node graphlets, in other words graph patterns with three nodes, in the network and then deriving a feature for a pair of disconnected nodes which accumulates the counts of those 3-node graphlets that these nodes form with their neighbors. We introduce a weighting scheme for these 3-node graphlets which considers the temporality and heterogeneity of links. Our experiments illustrate that network evolution cannot be explained by one specific feature at all time points which emphasizes the importance of combining different features into efficient models. We observe that some network properties can point out which weighting scheme for 3-node graphlets is more effective for temporal link prediction.

The problem of temporal link prediction in heterogeneous networks is a fresh problem with low maturity and gains more attention recently. Currently there are very few works which study it [31, 76, 118]. We propose a new method to solve this problem. Hence, we classify the main contribution of this work via the *exaptation* quadrant in Figure 1.3 close to the border with the *invention* quadrant.

1.4.3 Tools for Graph Pattern Matching

Challenges. Graph pattern matching is a fundamental and central part of graph pattern based link prediction. On the other hand, it is NP-complete and, thus, very time consuming. This brings us to the point that we need the fastest tools or algorithms which can solve this problem. We can use either specialized algorithms or general purpose database systems. The difference between these approaches is how data is managed

and what data structure is used. The advantage of using a database system is that it is suitable for various tasks of data manipulation. Database systems allow us to solve the graph pattern matching problem in complex networks, unlike specialized algorithms which are tailored to specific network structures. From practical point of view, it is also easier to implement a prototype. Last, but not least, is the fact that this is a vivid research area, hence we can profit from the speed gains achieved in the database community. The challenge, however, is that there is a plethora of database systems. It is unclear which one is better for the task of graph pattern matching.

Contributions. Firstly, we build a benchmark set including both synthetic and real data. The synthetic data is created using two different graph models while the real-world datasets include a citation network of scientific publications and a global terrorist organization collaboration network. Secondly, we create sample graph patterns for the synthetic and real-world datasets. Again, part of these patterns are generated randomly. The second set of patterns is created using frequent graph pattern mining. This means we select specific patterns which are guaranteed to occur multiple times in our benchmark set. Thirdly, we express the graph pattern matching problem in the query languages of the four database systems we use. These are SQL for relational databases, Cypher for graph databases, ASP for deductive databases, and SPARQL for RDF-based systems. We conduct an experimental comparison within a uniform framework using PostgreSQL as an example of relational database system, Jena TDB representing RDF-based systems, Neo4j as a representative for graph databases systems, and Clingo as a deductive database.

The problem of graph pattern matching is a well established problem in the graph theoretic community, however, we suggest to solve this problem with ASP which has not been done before. Therefore, we classify this contribution of our work via the *inspiration* quadrant in Figure 1.3. The database systems present a mature tool for solving data manipulation problems, but no benchmark of database systems for the task of graph pattern matching has been constructed before. Hence, we classify this contribution also via the *exaptation* quadrant in Figure 1.3.

We do not provide explicit contributions to the *invention* quadrant, but contributions 1 and 2 can be regarded as such.

1.5 Structure of the Work

Chapter 2 introduces the basic notions and notations which we use throughout the thesis. This includes the definitions from the areas of network analysis and graph theory.

Our work is multi-disciplinary, but the main contributions are made to the area of network analysis, in particular, they are all related to the link prediction problem. Therefore, in Chapter 3 we discuss the state-of-the-art regarding the link prediction problem. State-of-the-art publications related to the specific contributions are discussed within corresponding chapters.

The main part of the thesis starts with Chapter 4 which presents a novel application area for link prediction methods. We first formulate the citation count prediction

problem in a corpora of scientific publications as a link prediction problem in a citation network. Then we adapt a link prediction method to calculate a new feature GERscore which is applied to predict future citation counts for publications. We demonstrate the performance of the new feature as well as of several state-of-the-art features on two publication databases.

Chapter 5 contains the study of temporal evolution in heterogeneous social networks. The heterogeneity is expressed by the presence of links of two types in the network. By incorporating the temporal aspect of links into the framework of a triad-closing model, we calculate time and heterogeneity based scores for a pair of nodes to determine whether they form a link of a certain type in the future.

In Chapter 6 we build a benchmark of database systems for graph pattern matching. As it is shown in two previous chapters, we need to employ graph pattern matching: to calculate our new feature GERscore as well as to calculate time and heterogeneity based scores. The graph pattern matching problem is known to be computationally hard, thus we need efficient tools to solve it. To this end, we design a benchmark of database systems to determine the best tool for such a problem.

Conclusions are given in Chapter 7. This contains a summary and the discussion of the results. Finally, we discuss open issues and give an outlook towards future work.

This thesis is based on the following peer-reviewed publications:

- [92] Nataliia Pobiedina and Ryutaro Ichise. Predicting citation counts for academic literature using graph pattern mining. In *Proceedings of the 27th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2014, Kaohsiung, Taiwan, June 3-6, 2014.*

- [96] Nataliia Pobiedina, Stefan Rümmele, Sebastian Skritek, and Hannes Werthner. Benchmarking database systems for graph pattern matching. In *Proceedings of the 25th International Conference on Database and Expert Systems Applications, DEXA 2014, Munich, Germany, September 1-4, 2014.*

- [93] Nataliia Pobiedina and Ryutaro Ichise. Citation count prediction as a link prediction problem. In *Journal of Applied Intelligence, 42(4):1-17, 2015.*

- [98] Nataliia Rümmele, Ryutaro Ichise, and Hannes Werthner. Exploring supervised methods for temporal link prediction in heterogeneous social networks. In *Proceedings of the 24th International World Wide Web Conference, WWW 2015, Florence, Italy, May 18-22, 2015 (Companion Volume).*

The following peer-reviewed publications appeared during the PhD studies, but are not included in this thesis:

- [91] Nataliia Pobiedina. Modeling the flow and change of information on the web. In *Proceedings of the 21st International World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, pages 173âĂŞ178, 2012.

- [40] Anna Fensel, Julia Neidhardt, Nataliia Pobiedina, Dieter Fensel, and Hannes Werthner. Towards an intelligent framework to understand and feed the web. In *Business Information Systems Workshops - BIS 2012 International Workshops and Future Internet Symposium, Vilnius, Lithuania, May 21-23, 2012 Revised Papers*, pages 255âĂŞ266, 2012.

- [95] Nataliia Pobiedina, Julia Neidhardt, Maria del Carmen Calatrava Moreno, and Hannes Werthner. Ranking factors of team success. In *Proceedings of the 22nd International World Wide Web Conference, WWW 2013, Rio de Janeiro, Brazil, May 13-17, 2013 (Companion Volume)*, pages 1185–1194, 2013.

- [94] Nataliia Pobiedina, Julia Neidhardt, Maria del Carmen Calatrava Moreno, Laszlo Grad-Gyenge, and Hannes Werthner. On successful team formation: Statistical analysis of a multiplayer online game. In *Proceedings of IEEE 15th Conference on Business Informatics, CBI 2013, Vienna, Austria, July 15-18, 2013*, pages 55âĂŞ62, 2013.

- [87] Julia Neidhardt, Nataliia Pobiedina, and Hannes Werthner. What Can We Learn From Review Data? In *Proceedings of ENTER Conference 2015, Lugano, Switzerland, February 3-6, 2015*, pages 1941-5842, 2015.

CHAPTER 2

Preliminaries

2.1 Networks

Simply put, a network is a collection of points, where each pair can be connected by a line. In the area of computer science these points are referred to as *nodes* and the lines are referred to as *links*. Networks are also studied in the area of graph theory, where they are called *graphs*. Here, the points are called *vertices* and the lines are called *edges*. There is another area which studies networks, namely social network analysis. In this field the points are often referred to as *actors* and the lines are referred to as *ties*. We will follow mainly the terminology used in the area of computer science throughout this thesis. Example of a small network is given in Figure 2.1. This network is *undirected* since the links do not have direction.

Formally, a network G is defined by a triplet (V, E, L), where V is a set of nodes, E is a set of links and L is a set of labels for nodes and links. The set of labels can be provided by two sets L_V, a set of node labels, and L_E, a set of link labels. Then $L = L_V \cup L_E$. Often the set of labels is provided with the so-called *label function* for links and nodes.

We do not consider networks with hyper-links in this thesis, therefore the set of links in all studied networks is a subset of the cross-product of the set of nodes, in other words, $E \subseteq V \times V$. In case $E = V \times V$ the corresponding network is called *complete*, in other words there is a link between any pair of nodes in the network. A link is called a *self-loop* if it connects a node to itself. If two nodes in a network are connected by more than one link, then such network is said to have *multi-links*. In such case we cannot put $E \subseteq V \times V$, however, depending on the context, we can decompose the set of links according to different types of links.

The network in Figure 2.1 is *simple* since it does not contain self-loops and multi-links. The set of nodes consists of eight elements $V = \{v_1, ..., v_8\}$. The set of links has nine elements. Each link can be represented as a pair of two nodes (u, w), where u is the *source* of the link and w is the *target* of the link. Then the whole network can be specified by the number of nodes and a list of all links. Such a specification is called an *edge list*.

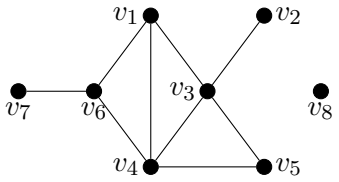


Figure 2.1: Example of a small network with eight nodes and nine links.

Another specification of a network is the *adjacency matrix*. If the network has n nodes, then the dimension of this matrix is $n \times n$, and its elements A_{ij} are calculated the following way:

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge between nodes } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases}$$

The *degree* of a node in the network is the number of links it has. For example, node v_3 in Figure 2.1 has degree four, which is often denoted as $deg(v_3) = 4$, and node v_5 has degree two. The nodes with which v_3 has links are called its *neighbors*.

A *path* is a sequence of nodes from the network such that every consecutive pair of nodes is connected by a link. For example, a path $p_1 = (v_7, v_6, v_1, v_3, v_2)$ is one of possible paths to reach node v_2 from node v_7. The *length* of this path is the number of traversed links which is four. Another possible path is $p_2 = (v_7, v_6, v_1, v_4, v_6, v_1, v_3, v_2)$ which is longer than the previous path. Furthermore, we traverse the link (v_6, v_1) twice.

We say that a path intersects itself if it visits a node or traverses a link more than once. The first considered path p_1 does not intersect itself, hence it is *self-avoiding*. The *shortest path* between two nodes is such a path between them so that no shorter path exists. The shortest path is self-avoiding. For example, the path p_1 is the shortest path between nodes v_2 and v_7 in our example network. Shortest paths are not unique, there can be more than one between a pair of nodes. For example, the path $(v_7, v_6, v_4, v_3, v_2)$ is also the shortest path between nodes v_2 and v_7.

However, there is no possible path for the node v_8 in the network. It has no links and is, thus, disconnected from the rest of the network. A network is called *disconnected* if there exist two nodes in the network with no path between them. We can divide the network into components so that each component represents a connected network. Such components are called *connected components* of the network. For the network in Figure 2.1 there are two connected components: the first one includes nodes $\{v_1, v_2, ..., v_7\}$ and the second component has only one node $\{v_8\}$. The connected component with the biggest amount of nodes is called *largest connected component*.

The *diameter* of a network is the length of the longest path among all possible shortest paths in the network. The diameter of the network in Figure 2.1 equals four and is calculated for the largest connected component.

Networks are one of the most generic data structures and therefore used to model data in various application areas. We can divide the networks into four big classes according

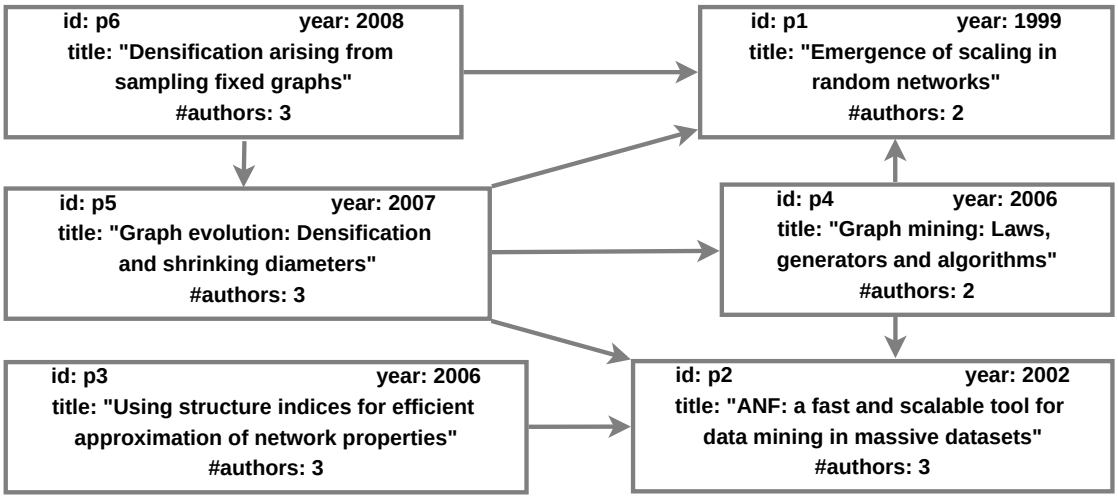


Figure 2.2: Example of a citation network.

to the natural phenomena which they model: technological, social, information and biological networks [89]. This classification is not strict, and in many cases a network can belong to more than one class. The advantage though is that problems and corresponding techniques look similar for networks in the same class. We will focus on social and information networks since the majority of experiments in the thesis are performed on the representatives of these two classes.

The *technological* networks encompass the Internet, the telephone network, various power grids and transportation networks. Here, the Internet is a worldwide network of physical data connections (e.g., optical fiber lines) between computers and related devices.

The World Wide Web (WWW) as a virtual network of web pages and hyperlinks belongs to the class of *information* networks. Another example of information networks is a citation network of scientific publications, where each node corresponds to a scientific publication or a patent, and links between them correspond to citations. In Figure 2.2 we provide a small example of a citation network which is constructed on the basis of six scientific publications. This is a *directed* network, meaning that each link has a direction. The link goes from the referencing paper to the paper which is cited. In case of directed networks we define two degrees for each node: *in-degree* as the number of incoming links and *out-degree* as the number of outgoing links. We can also see that nodes have attributes which are often called *node labels*. In this example we indicate such attributes of publications as the publication year, the title and the number of authors. We notice that this network is connected if we disregard the direction of links. However, once we consider the direction of links and when constructing paths between nodes follow this direction, we obtain that some pairs of nodes cannot be connected via directed paths. Such kind of networks are called *weakly connected*. In case a directed network remains connected when considering the direction of links, it is called *strongly connected*.

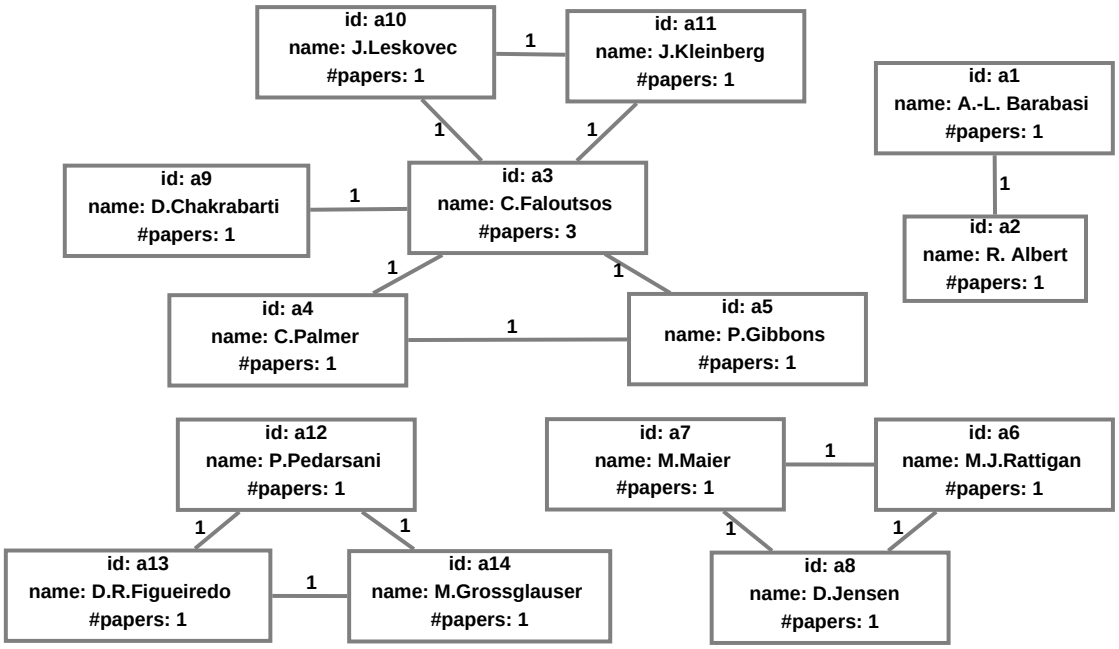


Figure 2.3: Example of a collaboration network between authors of scientific publications.

Biological networks include various networks which model interactions between different biological elements, e.g., metabolic networks, protein-protein interaction networks.

In *social* networks nodes correspond to people, or sometimes to groups of people, and links represent either interactions or relationships between them. One of the examples of social networks are collaboration networks. We can construct a collaboration network based on the citation network in Figure 2.2. If we put authors of papers as nodes in the network, then links between two nodes indicate that corresponding authors wrote a paper together. The obtained collaboration network is presented in Figure 2.3. Besides having node labels, this network has *link labels*. These labels indicate how many times the collaboration took place, in other words how many papers two connected authors wrote together. This special type of link labels are often called *weight*, and the networks are also referred to as *weighted networks*. In social network analysis the weights of links indicate how strong the ties between actors are.

In Figure 2.4 we show another example of a social network [120]. The nodes in this network correspond to 34 members of a karate club, and links between nodes indicate friendship ties between the members. The nodes are colored with regard to the community they belong to. There is no strict definition of a community in a social network. By a *community* one generally understands a subset of nodes of the network with many links between them and few connections to the rest of the nodes in the network. The communities in the karate network are discovered with the Louvain method [18]. We use Gephi to visualize the network [14].

The maximum possible number of links in a simple network is $\frac{1}{2}|V|(|V|-1)$, where

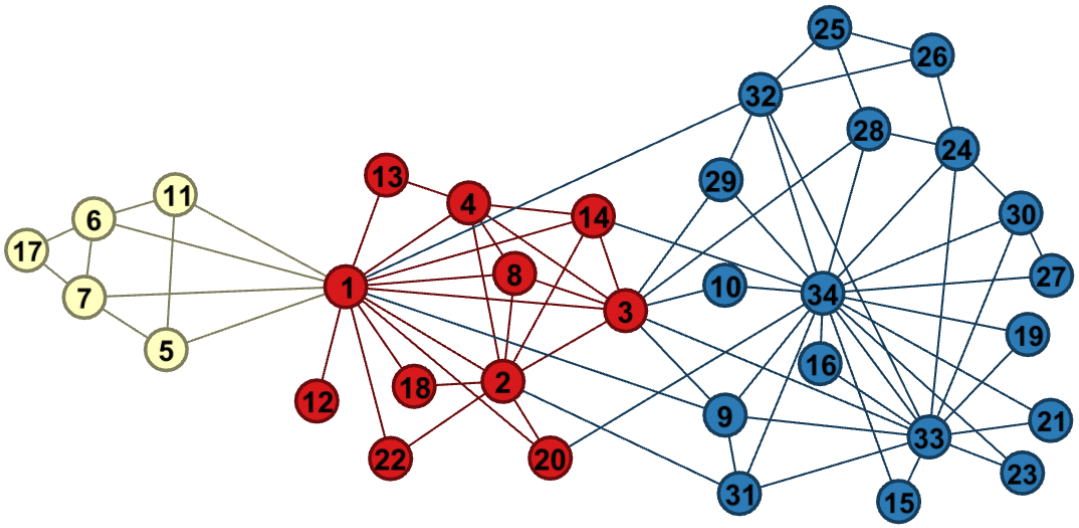
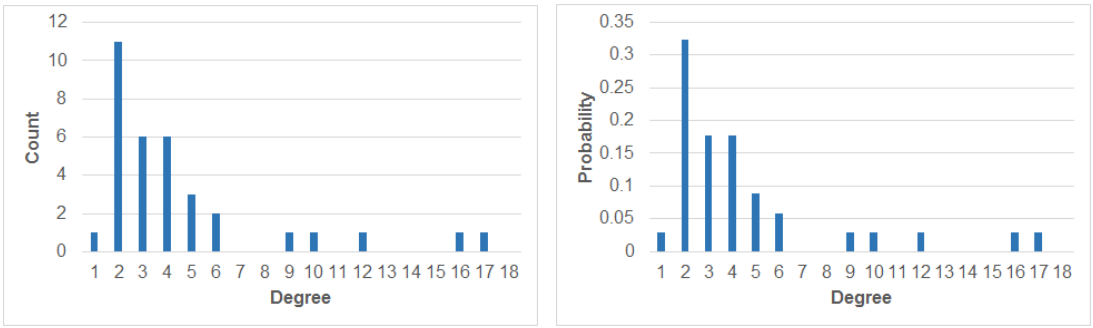


Figure 2.4: Friendship network between members of a karate club. Colors of the nodes correspond to the detected communities. Numbers in the nodes correspond to their ids.



(a) Showing counts on y-axis.

(b) Showing probabilities on y-axis.

Figure 2.5: Degree distribution for the karate network.

$|V|$ stands for the number of nodes in the network. Let $|E|$ stand for the number of present links in the network. Then the *density* of the network ρ is calculated as:

$$\rho = \frac{2|E|}{|V|(|V|-1)}.$$

The density is a value in the range between 0 and 1. A network is said to be *sparse* if the density tends to zero $\rho \to 0$ as $|V| \to \infty$. In practice a network is considered sparse if $|E| \sim O(|V|)$, that is the number of links is comparable to the number of nodes. In case the density tends to a constant when $|V| \to \infty$ the network is called *dense*.

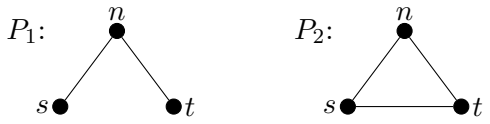


Figure 2.6: A connected triple (P_1) and a closed triad (P_2).

A very important property in network analysis is *degree distribution*. If we calculate for each node its degree and group nodes by their degrees normalized by the total number of nodes in the network, we obtain a degree distribution for the network. In Figure 2.5 we show the degree distribution for the karate network from Figure 2.4. We put the value of node degree on x-axis, and on y-axis we show either the normalized number of nodes which have this degree (Figure 2.5(b)) or the total number of nodes with this degree(Figure 2.5(a)). Clearly, there is no difference in these distributions. However, the second way is more correct since the normalized counts can be viewed as probabilities of having a node with a specific degree in the network. From the practical point of view, using counts is more convenient.

Assume that we are given the degree distribution p_k for the network. Here p_k denotes the probability of a node to have degree k. If the logarithm of the degree distribution is a linear function of degree k, then we call such distribution *power-law degree distribution*. The following formula is true for such distribution:

$$\ln p_k = -\alpha \ln k + c,$$

where α and c are constants. The constant α is referred to as the *exponent* of the power law. Networks with power-law degree distributions are often called *scale-free*.

There are many other interesting properties defined for the networks. We will introduce two more properties which are of relevance to the work performed in this thesis. A comprehensive study of various network properties is provided in the book by Newman [89].

The transitivity T of a graph is based on the relative number of triangles in the graph compared to the total number of connected triples of nodes. In Figure 2.6 we show two connected triples (P_1 and P_2). However, P_2 can be seen not only as a connected triple, but also as a closed triad, or alternatively a triangle. Then the transitivity is calculated as:

$$T = \frac{3 \times (\text{number of triangles})}{(\text{number of connected triples})}.$$

The transitivity is a global network property and provides an estimation how likely nodes s and t are linked if they have a common neighbor.

The local clustering coefficient of a node n is defined as:

$$C_n = \frac{(\text{number of pairs among neighbors of } n \text{ that are connected})}{(\text{number of pairs among neighbors of } n)}.$$

The clustering coefficient of the network is yet another measure of the number of triangles in the network like. The clustering coefficient of a network is based on the local clustering

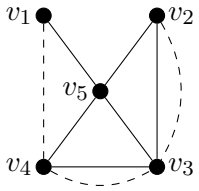


Figure 2.7: Example of a multi-relational network with two types of links: solid and dashed lines represent these types.

coefficient for each node:

$$\bar{C} = \frac{1}{|V|} \sum_{i=1}^{|V|} C_i.$$

Both transitivity and clustering coefficient measure the relative frequency of triangles in the network. However, average clustering coefficient places more weight on the low degree nodes, while the transitivity places more weight on the high degree nodes. Furthermore, local clustering coefficient can be used as an indicator of the presence of *structural holes* in the network. Informally, a node is said to span a structural hole if it has neighbors which are not well connected to each other. For example, node v_1 in the karate network in Figure 2.4 spans a structural hole and connects the "yellow" community to the rest of the network. If we remove this node from the network, it immediately becomes disconnected. We can see that node v_1 is very important in the network due to the fact that there are many structural holes around it in the network. Thus, the local clustering coefficient measures how influential a node is with regard to the flow in the network. It has lower values if there are more structural holes in the network around the considered node.

So far we have provided examples of homogeneous networks, i.e., networks with the same type of nodes and the same type of links. However, many natural phenomena are more complex. For example, on the social platform Facebook we can study the formation of friendship ties and communication interactions between users. Networks with different types of links are one of possible representatives of *heterogeneous networks*. They are also referred to as *multi-relational networks*. In Figure 2.7 we provide an example of a small multi-relational network with 6 links of the first type (solid line) and 3 links of the second type (dashed line). Another example of a heterogeneous network is a combination of the citation and collaboration networks. We have two types of nodes: authors and publications. We can define various links between the nodes: for example, authorship links between authors and publications, citing links between publications, friendship links between authors. But this list can be further extended depending on the available information.

2.2 Link prediction

Link prediction is an important problem in network analysis. Assume we have a friendship network of five people shown in Figure 2.8. We know about four friendship ties between

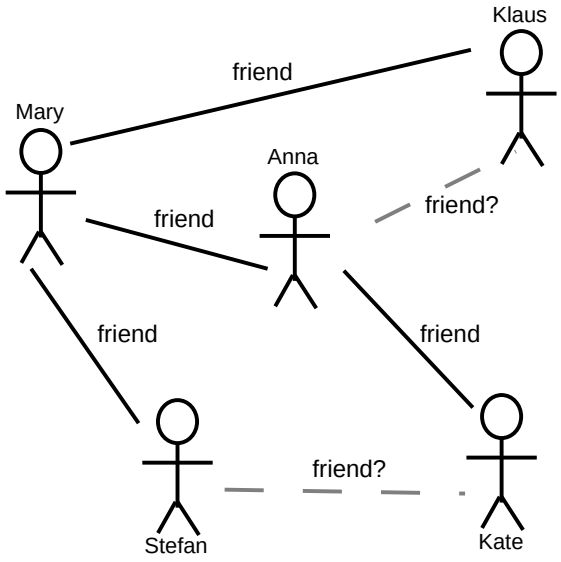


Figure 2.8: Illustration of the link prediction problem in a small friendship network.

them. But we want to learn if Klaus and Anna are friends as well as if Stefan and Kate are friends. Here, we might refer either to the future formation of friendship ties or to the friendship ties which are hidden. Real-world networks are highly dynamic, therefore we want to predict the evolution of networks and the appearance of new links. In practice, we often do not have complete knowledge about the observed phenomena, hence we need to infer missing or hidden information. However, there are also cases when links disappear over time. Though prediction of disappearing links is a valid formulation of the link prediction problem, it is beyond the scope of this thesis. Henceforth, we provide the formal definitions and overview of available methods to predict new or hidden/missing links in networks.

In general, we have a network $G = (V, E, \lambda, \tau)$, where the sets of nodes and links can be either homogeneous or heterogeneous, function λ assigns some node and link labels, and τ assigns time stamps to links. Note that λ in some cases can be also time dependent. For example, if we consider the friendship network from our small example (Figure 2.8), people's marital status or hobbies might change over time. Now let $G[t] = (V[t], E[t], \lambda, \tau)$ denote the sub-network of G which exists at time t. The link prediction problem consists either of finding hidden connections or predicting links which will appear in the future based on the previously observed network states [50]. Formally, these two problems can be defined the following way in G:

1. *Inferring missing links*: Given $G'[t] = (V[t], E'[t], \lambda, \tau)$ where $E'[t] \subset E[t]$, i.e., an incomplete set of links, infer $E[t]$.

2. *Predicting future links*: Given $G[t] = (V[t], E[t], \lambda, \tau)$ predict $E[t + h]$ for some $h > 0$.

The first problem, where researchers consider a network to be static and aim at inferring missing links without taking into account temporal evolution, is better studied in the link prediction community. The latter, predicting future links, has gained more interest recently. Note that in reality the statement $E[t] \subseteq E[t+h]$ does not always hold for the links can disappear over time. Since we do not consider networks with such property, we assume that $E[t] \subseteq E[t+h]$ for all possible $h > 0$, and the task of predicting future links is to predict new links which will form, in other words we need to predict $E[t+h] \setminus E[t]$. We provide an overview of some link prediction methods in Chapter 3.

There are various measures used to estimate the performance of link prediction methods. If we know *positive instances* (**P**) and *negative instances* (**N**) as well as *true positives* **TP**, *true negatives* **TN**, *false positives* **FP**, *false negatives* **FN**, we can define the following evaluation measures:

- *Sensitivity, or True Positive Rate (TPR)*:

$$\text{TPR} = \frac{|TP|}{|TP| + |FN|}$$

- *Specificity, or True Negative Rate (TNR)*:

$$\text{TNR} = \frac{|TN|}{|FP| + |TN|}$$

- *Precision*:

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

- *Recall*:

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

- *F-measure*:

$$F\text{-measure} = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}$$

- *Accuracy*:

$$\text{Accuracy} = \frac{|TP| + |TN|}{|P| + |N|}$$

- *Top K predictive rate* is the percentage of correctly classified positive instances among the top K predicted scores by a link prediction method, where K is a threshold which needs to be specified.

- *Receiver Operating Characteristic curve (ROC curve)* is a curve which maps true positive rate against false positive rate at different decision boundary thresholds.

- *Area Under Receiver Operating Characteristic curve (AUROC)* stands for the area under the ROC curve.

- *Precision-Recall curve (PR-curve)* consists of points which correspond to different precision and recall values at different score thresholds.
- *Area Under Precision-Recall curve (AUPR)* stands for the area under the PR-curve.

2.3 Graph pattern matching

The problem of graph pattern matching originates from the area of graph theory. Therefore, when talking about this problem, we might often use the graph terminology from this field.

We will provide definition of the graph pattern matching problem for *undirected, simple graphs*, that means graphs without self-loops and with not more than one edge between two vertices. However, this definition can be easily extended to heterogeneous networks. We denote a (labeled) graph G by a triple $G = (V, E, \lambda)$, where V denotes the set of *vertices*, E is the set of *edges*, and λ is *labeling function* which maps vertices and/or edges to a set of labels, e.g. the natural numbers $\mathbb{N}$. An edge is a subset of V of cardinality two.

Let $G = (V_G, E_G, \lambda_G)$ and $P = (V_P, E_P, \lambda_P)$ be two graphs. An *embedding* of P into G is an injective function $f : V_P \to V_G$ such that for all $x, y \in V_P$:

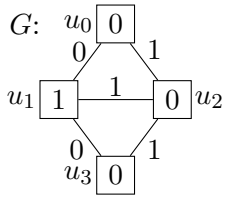
1. $\{x, y\} \in E_P$ implies that $\{f(x), f(y)\} \in E_G$;
2. $\lambda_P(x) = \lambda_G(f(x))$; and
3. $\lambda_P(\{x, y\}) = \lambda_G(\{f(x), f(y)\})$.

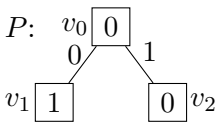
This means if two vertices are connected in P then their images are connected as well. Note that there is no requirement for the images to be disconnected if the original vertices are. This requirement would lead to the notion of *subgraph isomorphism*.

Instead, the problem of *graph pattern matching* is defined as follows: Given two graphs, G and P, where P is the smaller one, called *pattern* or pattern graph, the task is to compute all embeddings of P into G. Figure 2.9 shows an example of a graph G, a pattern graph P and all possible embeddings of P into G. In practice, we may stop the pattern matching after the first k embeddings are found.

The problem of deciding whether an embedding exists is NP-complete, since a special case of this problem is to decide if a graph contains a clique of certain size, which was shown to be NP-complete [60]. However, if the pattern graph is restricted to a class of graphs of bounded treewidth, then deciding if an embedding exists is fixed-parameter tractable with respect to the size of P, i.e., exponential with respect to the size of P but polynomial with respect to the size of G [5].

If we have a database of snapshot networks, then to uncover frequent structural patterns in this database we need to employ *graph pattern mining*. In this case graph pattern mining is performed within the transactional setting: given a multiset of structures, one has to find substructures common to a minimum number of the multiset members. However, some real-world networks are hard to represent in the transactional setting.





$M_1 = \{(v_0, u_0), (v_1, u_1), (v_2, u_2)\}$
$M_2 = \{(v_0, u_3), (v_1, u_1), (v_2, u_2)\}$

Figure 2.9: Examples data graph G, pattern graph P, and resulting embeddings M_1, M_2.

For example, the Web graph or any social network is rather a single large network, and in many cases it does not make sense to split it up into smaller parts. Here, the focus is to find structural regularities or anomalies within such networks, rather then within a set of them. We are dealing with frequent pattern mining in the *single-graph* setting [22].

When performing graph pattern mining we use a constraint which provides the lower limit on how frequent the discovered pattern should be. This constraint is called *minimum support*. The support measure needs to be non-negative and anti-monotonic which means that the support of a subgraph is no less than of the initial graph. Bringmann and Nijssen provide examples of support measures for the single-graph setting [22].

CHAPTER 3

State of the Art

Link prediction is an important problem in the field of network analysis which has attracted much attention from academic and industrial researchers in the recent years. Peng et al. pointed out the popularity growth of link prediction topic in social network analysis by collecting the number of published papers from three computer science libraries (Elsevier, ACM and IEEE) [111]. In Figure 3.1 we present the chart from their work which shows the number of publications with search keywords "link prediction social network" in the library Elsevier from 2000 till 2013. We can observe an exponential growth in the number of papers.

Interestingly, link prediction is a highly multi-disciplinary research field with contributions from such disciplines as psychology, sociology, physics, computer science and economics. Peng et al. studied closely 130 papers published in the prominent journals in the years 2000-2013, and they have found out that 98% authors are from four disciplines, such as computer science, physics, economics and management [111]. However, they also

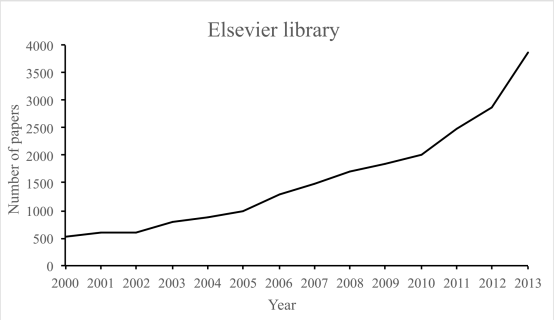


Figure 3.1: Number of published papers in Elsevier library which are related to the link prediction problem (taken from [111]).

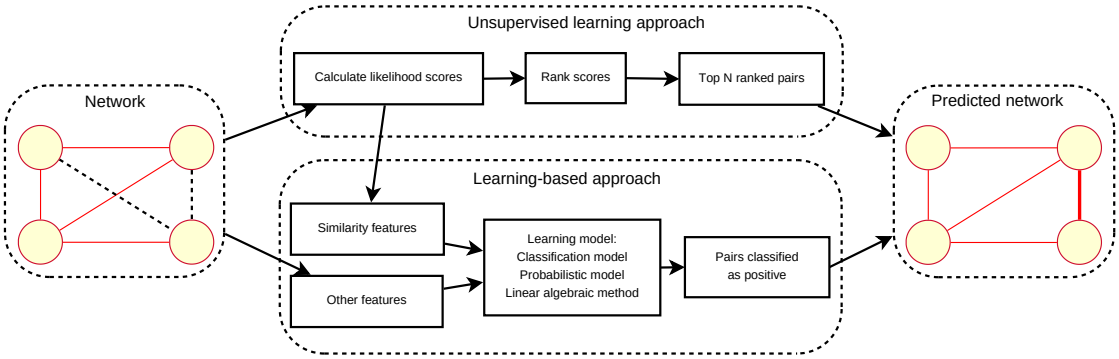


Figure 3.2: The general framework for link prediction [111].

observe that authors from different disciplines lack co-operation, and the majority of papers are published by the scientists from the same field.

With such abundance of scientific papers on the link prediction problem, it is no wonder that many algorithms to solve this problem have been proposed. There are also several excellent survey works [74, 114, 54, 82, 111]. Liben-Nowell and Kleinberg provide an overview of some classical topology-based measures to solve the link prediction problem [74]. Lü and Zhou summarize popular link prediction methods from the area of physics for complex networks [82]. Hasan and Zaki provide an overview of link prediction methods by categorizing them according to the considered prediction model: binary classification model, probabilistic model and linear algebraic model [54].

In our opinion, the recent survey work on link prediction in social networks proposes a comprehensive systematization and overview of link prediction techniques [111]. Though this work focuses on social networks, the presented link prediction framework as well as considered methods are applicable to a wide range of real-world networks. Therefore, the current chapter will follow their systematization with minor revisions. We present the general framework to solve the link prediction problem in Figure 3.2.

We are given a network with four nodes and four links which are marked with solid lines in the figure. We have two pairs of disconnected nodes in this network which are marked with dashed lines in the figure. We want to find out whether the corresponding links are missing or will appear in the network in the future. Remember that we have provided two formulations of the link prediction problem in Chapter 2. The general process of solving the link prediction problem in both cases is similar. According to the available solutions at the current stage, we can choose between an unsupervised learning and a learning-based approaches. Peng et al. call the first approach the similarity-based approach [111]. However, we will show in the follow section that not all methods within this approach are based on the similarity between two nodes. In the unsupervised learning approach we calculate for each candidate pair of nodes a score which estimates the likelihood of a link between them. Then we rank the pairs according to the scores and take top N (for example, 10) pairs as the ones which will form the link. In the learning-based approach we regard the link prediction problem as a binary classification

Table 3.1: Social theories behind link prediction techniques.

Social theory	Rationale	Examples
Community structure	Users within the same community exhibit similar behavior patterns.	W-form measures [108]
Structural balance	A friend of my friend is my friend.	TranFG [105], TRFG [37]
Structural hole	Nodes with access to local bridges are successful.	TranFG [105]
Weak-tie theory	Weak ties maintain the network's connectivity.	NC-CN [81]
Homophily	We connect with people who are similar to us.	Katz [62], CN [88], AA [2], JC [99]
Centrality	Rich gets richer.	PA [88], Katz [62]
Microscopic mechanism	Network evolution on a small scale.	TRFG [37], VCP [76], GERM [21]

problem where each pair of nodes is assigned a class label with regard to the presence of a link between them [53]. So, a pair of nodes is assigned to the positive class if there is a link between them, otherwise they are assigned to the negative class. Each pair of nodes is represented by a feature vector: we can take either the scores calculated in the unsupervised approach or we can design some new features. Then we can choose a learning model which takes the feature vector and the class label as input and outputs those pairs which are most likely to form the links.

In the rest of this chapter we will present various link prediction techniques following either the unsupervised learning or the learning-based approaches.

3.1 Unsupervised learning approach

In the unsupervised learning approach we want to compute a score for a potential pair of disconnected nodes which reflects the probability that these nodes get connected by a link. The higher the score is, the higher the likelihood that these two nodes are linked. The opposite is also true: we expect the score to be high for any connected pair of nodes. The majority of calculation techniques for such a score are based on the similarity between the nodes. This assumption has roots in the social theory of *homophily* [66]. This theory suggests that a node in the social network tends to be similar to its neighbors, or "friends". The neighbors of a node are often similar to this node with regard to different attributes, for example, age, occupations, interests and beliefs. McPherson et al. study this phenomenon in social networks, and refer to it as "birds of a feather" [84]. The concept of similarity is extended to other types of networks, and it is also argued that node similarity is one of the most crucial factors of link formation.

However, there are other social theories which motivate link prediction techniques. We enumerate some of these theories in Table 3.1 and provide examples of link prediction methods which support those theories. Peng et al. outline three categories of link prediction techniques in the similarity-based approach: node-based metrics, topology-based metrics and social theory based metrics [111]. But we think that such classification is not accurate since almost every node-based or topology-based metrics has some underlying social theory behind it. We rather provide a rough classification of some node-based and topology-based metrics according to the social theories. In the following we will summarize the social theories mentioned in Table 3.1. Remember that in the area

of social network analysis links are called ties and nodes are often referred to as actors. So, in the context of social theories we will often use this terminology.

In the network evolution the spread of influence and the process of link formation are closely interrelated [29, 118]. One can argue that the link between two nodes forms due to the reason that the mutual influence between them is strong enough. On the other hand, the appearance of a link between these two nodes enforces the influence between them. Thus, many methods to study influence in the networks can be adapted to solve the link prediction problem. The opposite is also true: many link prediction techniques are used to construct influence models.

Node centrality is an important factor in the field of social influence analysis in networks [43]. A node with a high centrality has more influence in the network. If we talk about social networks, we can say that a high centrality node is perceived to have a higher social status. Since the other nodes tend to connect to already influential nodes in the network, there are a few link prediction methods which are constructed based on the centrality measures of the nodes.

Another interesting social theory is the theory of *community structure* present in many real-world networks. In particular, social networks can be divided into sub-communities, which correspond to tightly-knit groups of actors within the larger looser network. For example, friendship networks often contain circles of friends within which connections are strong and frequent, but between which they are weaker and rarer. This observation provides the rationale for using community structure in the link prediction task [108]. Moreover, it has been also discovered that nodes within the same community exhibit similar behavioral patterns in the network [41].

According to Granovetter, the *tie strength* between two nodes can be determined by the amount of common neighbors they have [52]. If they share many neighbors, then the tie between them is considered strong, otherwise weak. In practice, the tie strength is often determined by the weights of the links. Based on this theory, we expect two nodes to be connected in the network if they share many neighbors. The concept of tie strength is related to triadic closure or *structural balance* in the network. If strong ties connect nodes v_1 to v_2 and v_2 to v_3, then v_1 and v_3 are very likely to be connected by a strong tie. Conversely, if the ties are weak, v_1 and v_3 are unlikely to be connected by a strong tie. An analogy to the real-world phenomena would be the following notions: "a friend of my friend is my friend" and "an enemy of my enemy is my friend".

Not only the presence of strong ties is used to predict links in the network. Lü and Zhou observe that considering the weights of links often leads to the worse performance in the link prediction task [81]. This observation supports the *Weak-Ties theory* which emphasizes that weak ties play an important role in preserving the network's connectivity

When two nodes v_1 to v_2 are connected by a weak tie so that they have no neighbors in common, such tie is called a *local bridge*. If the removal of such a link in the network results in the disconnection of the corresponding components, such tie is referred to as a *global bridge*. The effect of local and global bridges is such that they give more influence to the adjacent nodes since they regulate the flow of information between the components which they bridge. The presence of local bridges points also to *structural holes* in the

Table 3.2: Link prediction techniques.

Category	Subcategory	Examples
Node	Profile similarity	[17]
	Text similarity	[17]
	Actions similarity	[7]
Topology	Neighbor	CN, JC, AA, PA
	Path	Katz, Shortest Path
	Graph pattern	VCP, GERM
	Randow walk	SimRank, PropFlow

network We gave an informal definition of a structural node in the previous chapter. An alternative definition would be if a node is connected to multiple local bridges. It has been shown that an actor's success within a social network (respectively, node's influence in the network) often depends on its access to local bridges [23]. Thus, we expect that structural holes are more likely to accrue new links.

The *microscopic mechanism* refers to the process of network evolution on the local level. Here, we are interested to investigate the local embedding of nodes and extrapolate the obtained observations to the process of link formation in the network. In the field of sociology, microsociology studies the human social interactions on a small scale [102].

Following the presented social theories, many link prediction methods have been developed. In this section we will provide an overview of techniques which calculate a score for a potential pair of nodes. The obtained score estimates the likelihood of a link formation for this pair: the higher the score is, the higher chances are that the link is formed. The summary of these techniques is provided in Table 3.2. We have two main categories: node-based and topology-based metrics. Node-based metrics are based mainly on the theory of homophily: the more similar nodes are, the more likely they are to get connected. Topology-based metrics can be further classified into neighborhood based, path based, random walk based and graph pattern based metrics. The difference is in how these methods account for the network structure.

3.1.1 Node-based metrics

When using this approach, a similarity measure for a pair of nodes is calculated based on the content and semantics of these nodes, and the link presence between them is determined by the obtained measure [114].

In case of social networks people can indicate their profiles with such attributes as age, gender, occupation, interests and so on. If we have an information network, then some text can be associated with each node. Thus, we can develop measures which estimate the similarity of profiles or texts. We mention profile and text similarity measures as possible methods in this category.

Bhattacharyya et al. develop a node-based metrics for link prediction by using the keywords of user profiles [17]. The similarity between two people in the network is defined by the distance between the keywords of their profiles. Anderson et al. track actions of users within the network and use this information to determine interests of users [7]. The similarity of users is calculated as the cosine between the action vectors of those users.

The disadvantage of node-based metrics is two-fold: firstly, in many real-world networks it is impossible to obtain full and credible information about nodes, and secondly, the nature of interactions between nodes is disregarded.

3.1.2 Topology-based metrics

The techniques within this category aim at calculating a likelihood score between a pair of nodes based on the topological metrics of the network. Such scores are then used to build models for the link prediction. This approach is the most widespread currently.

Liben-Nowell and Kleinberg discuss several unsupervised methods [74]. According to the structural information they use, topology-based metrics can be classified into neighborhood-based, path-based, graph pattern based and random walk based metrics. Unlike previous survey works, we differentiate graph pattern based metrics from the other categories. Our rationale is that, firstly, the graph pattern matching problem is NP-complete, secondly, graph pattern based metrics consider not only connected paths, but also disconnected graphs found within the network. In the following sections we will provide an overview of some popular methods within each category.

Neighborhood based metrics

Among the topological metrics which are used, the neighborhood based scores form the biggest category. Let $\mathbf{A}$ be the adjacency matrix of the network of interest, and let N_x denote the set of immediate neighbors of the node x.

The *Preferential Attachment score (PA)* for a node pair (s, t) is the product of their degrees [88]:

$$|N_s| * |N_t|.$$

The preferential attachment score is based on the theory of centrality and supports the concept "rich gets richer", which is observed in many social networks. It means that nodes with a high centrality, which is measured in this case by the node's degree, are more likely to get new links.

The *Common Neighbors score (CN)* counts the number of common neighbors [88]:

$$|N_s \cap N_t|.$$

This measure supports several social theories. In particular, the theory of homophily: two nodes with many shared neighbors are considered to be very similar; and the theories of tie strength and structural balance: if two nodes are linked to the same node, then they are likely to be connected as well.

The *Jaccard's coefficient (JC)* is a normalized number of common neighbors [99]:

$$\frac{|N_s \cap N_t|}{|N_s \cup N_t|}.$$

The *Adamic-Adar score (AA)* weights the impact of neighbors inversely according to their degrees [2]:

$$\sum_{n \in N_s \cap N_t} \frac{1}{\log |N_n|}.$$

AA weighs the common neighbors with smaller degree more heavily. Out of JC, CN and AA, several works have pointed out that AA performs better.

There are many other neighborhood-based metrics which can be looked up in the survey work by Peng et al. [111]. The advantage of neighborhood-based metrics is that they are easy to compute.

Path based metrics

Shortest path distance is one of possible path-based metrics. The rationale behind this measure is that nodes which are closer to each other are more likely to get connected. However, due to the small world phenomenon [113], many pairs of nodes in real-world networks are separated by a small number of nodes. This leads to the poor performance of this measure which was reported in several works [53, 74].

Katz metric is a variant of the shortest path distance [62]. However, the performance of Katz metric for link prediction is usually better than for the shortest path distance. This metric sums over all paths that exist between a node pair (s, t):

$$\sum_{l=1}^{\infty} \beta^l \cdot |\text{paths}^{\langle l \rangle}(s, t)|.$$

By $\text{paths}^{\langle l \rangle}(s, t)$ we mean paths of length l between nodes s and t. This metric penalizes the longer paths with the factor β^l, where l is the length of the path. On one hand, Katz metric estimates the similarity of two nodes by the number of paths which connect them. On the other hand, it also measures the centrality of the nodes since nodes which have many paths going through them are perceived to be more central and, thus, have more influence on the network evolution. The disadvantage of Katz metric is that it is computationally expensive. This task has roughly cubic complexity, however, to improve the computational cost, one often limits the class of considered paths to a certain length.

Random walk based metrics

The *Hitting Time (HT)* for two nodes s and t is the expected number of steps required for a random walk starting at s to reach the node t [42]. Shorter hitting time indicares that the nodes are similar, hence there is a higher chance that they get connected. The advantage of this metric is that it is easy to compute by performing some trial random walks. However, the performance of HT for the link prediction task can be poor since its value can have high variance [74].

Chung and Zhao noticed that PageRank, which is used to rank web pages, is related to the hitting time [27] This observation led to the point that PageRank can be adapted for the link prediction task. The original PageRank of a node is proportional to the

probability that this node will be reached through a random walk in the network [20]. The factor ϵ specifies how likely the random walk is to visit node's neighbors with $1 - \epsilon$ standing for the probability to return to the node. Let $\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j}$ be a diagonal degree matrix of the adjacency matrix $\mathbf{A}$. Then *Rooted PageRank (RPR)* is calculated the following way [74]:

$$(1 - \epsilon)(\mathbf{I} - \epsilon \mathbf{D}^{-1}\mathbf{A})^{-1}.$$

PropFlow (PF) is another metric based on random walks which is more localized than Rooted PageRank [77]. It is proportional to the probability that a restricted random walk starting at the node s ends at the node t in no more than l steps. The restricted walk selects links based on weights and terminates when it reaches the node t or revisits any node. If nodes s and t are linked, the PropFlow metric is calculated the following way:

$$PF(s,t) = PF(a,s) \times \frac{w_{st}}{\sum_{k \in N(s)} w_{sk}}.$$

In this formula w_{st} denotes the weight of the link between nodes s and t, a is the previous node on the path of the random walk. If s is the starting node, then $PF(a,s) = 1$. Unlike Rooted PageRank, the computation of PropFlow does not require walk restarts or convergence. It simply uses a modified breadth-first search which is restricted to the height l. Therefore, the calculation of PropFlow is faster.

Graph pattern based metrics

In this category of topology-based metrics we have methods which calculate the likelihood score of link formation for a pair of nodes based on some defined graph patterns. These patterns can be either provided by default or one can use a graph pattern mining procedure to discover them. The score is calculated based on the local topological embedding of the considered two nodes with regard to the specified graph patterns. From the computational point of view, graph pattern based metrics have higher costs since they employ a graph pattern matching procedure. However, due to the advances in graph theory and development of more efficient graph pattern matching algorithms, this type of metrics are gaining more attention and have been shown to provide good results. The advantage of the graph pattern based metrics is that there is a natural way to account for the global characteristics of the network by calculating the frequencies of the defined graph patterns in the network. Another benefit is that there is an opportunity to predict links between nodes which were not yet observed in the network.

By introducing graph evolution rules Bringmann et al. illustrate a way to predict links between an existing node and a new node in the network [21]. Their approach, *Graph Evolution Rule Miner (GERM)*, is based upon mining Graph Evolution Rules (GER) in a network where links are stamped with the creation time and nodes may have up to one integer label. The obtained rules provide an opportunity to capture the temporal evolution of the network and are used to calculate the score for a candidate pair of nodes to estimate the connection likelihood in the future. This is one of very few

link prediction methods which can be used to calculate a score between an old and a new nodes in the network.

Davis et al. construct a weighting scheme for different edge type combinations based on the frequencies of 3-node graphlets [31]. Davis et al. define patterns, or connected 3-node graphlets, following the theory of social balance and microscopic mechanism, by constructing graph patterns with all possible combinations of link types. They also suggest a new feature-based model to solve the link prediction task in networks with more than one type of links. They call this model *Multi-Relational Link Prediction model (MRLP)*. In the following work Lichtenwalter and Chawla extend the class of graphlets and introduce a new method for link prediction called Vertex Collocation Profile (VCP) [76]. Here the authors consider graphlets with more than 2 nodes. In particular they conduct experiments on 11 networks using graphlets with 3 and 4 nodes. Theirs results indicate that VCP outperforms the classical features in many cases either in terms of AUROC (area under ROC curve) or AUPR (area under precision recall curve). Overall, the approach of VCP can be used either as a feature vector [76] or one can compute a score which estimates the connection likelihood [31].

3.2 Learning-based approach

Learning-based link prediction methods allow the combination of the metrics from the unsupervised learning approach together with external information. Lichtenwalter et al. promote supervised methods for the link prediction problem [77]. The authors show that supervised learning provides better results due to the ability to reduce variance and to cope with high imbalance in class distribution. Unlike a classical approach in machine learning to overcome imbalance by oversampling the minority class, the authors suggest to use skew-insensitive trees based on Hellinger distance or to undersample the majority class in a particular neighborhood.

Depending on the type of the learning model, we differentiate such approaches as feature-based classification, probabilistic graphical models and linear algebraic methods.

3.2.1 Feature-based classification

In the feature-based classification approach the problem of link prediction is treated as a binary classification problem where a pair of nodes is assigned the positive class if there is a link between them, otherwise it is the negative class. A pair of nodes is represented by a vector of features which can correspond to various metrics from the unsupervised learning approach. Additionally, new features are developed to improve the performance of link prediction.

Lichtenwalter et al. develop a *High Performance Link Prediction model (HPLP)* model to solve the link prediction problem [77]. This model combines several topology-based metrics as well as degrees and volumes of two considered nodes. The included metrics are common neighbors, number of shortest paths of length up to 5, PropFlow with $l = 5$, Adamic-Adar score, Jaccard's coefficient, Katz metric with parameters $\beta = 0.005, l = 5$

and preferential attachment score. This model often serves as the baseline for new methods in the learning-based approach.

Valverde-Rebaza and Lopes combined topology with community information on the Twitter social platform [108]. They consider interests and behaviors of users when performing community detection. The assumption of their new method is that common neighbors of two nodes within the same community contribute to the connection likelihood of these two nodes much more than inter-community neighbors.

Lü and Zhou point out that weak ties play a significant role in network evolution, and by emphasizing the contribution of weak ties the prediction accuracy of link prediction methods can be improved for some networks [81]. Based on this observation, Liu et al. proposed a link prediction model which combines the information about weak ties and three node centralities: degree, closeness and betweeness centrality [78]. They argue that each common neighbor for a candidate pair of nodes plays a different role which is determined by the centrality measure. The link prediction accuracy is also improved by considering weak ties.

Supervised methods are appropriate if we know the ground truth. This poses a challenge since the real links in many networks are often not observed. That is why a set of methods have been developed to predict links across heterogeneous networks. Here, there are two networks: a source network with observed links and a target network for which we need to predict links. These two networks can be completely different: with different nodes and different types of interactions (links). The approach is to learn a model on the source network and then to apply it to the target network. Several works have considered this problem constructing a transfer learning framework by incorporating social theories [105, 37]. These works focus on generalizing the link prediction methods.

3.2.2 Probabilistic graphical models

In the probabilistic approach, the goal is to find a model which best describes the network [114]. The model is defined by a set of parameters which are estimated on the observed network. The link presence between a pair of nodes is determined then by the conditional probability.

When using probabilistic graphical models which rely on Bayesian concepts, we want to obtain a posterior probability which denotes the connection likelihoods for the considered pairs of nodes. The advantage of this approach is that it is possible to use the output of these models as an additional feature in any feature-based method. Wang et al. suggest a Markov Random Field (MRF) based local probabilistic model, an undirected graphical model [110]. Their model also incorporates a node-based and topology-based metrics, namely Katz and a similarity score for node attributes.

Kashima et al. developed a parameterized probabilistic model which tunes parameters in the proposed generic network evolution model [61] The performance of this model, though, depends on the extent ti which the network agrees with the proposed evolution model.

Clauset et al. propose a hierarchical probabilistic model [28]. This model considers the hierarchical organization of the network, according to which nodes divide into groups

that can be further subdivided into groups of groups, and so on. The model infers this hierarchical structure from the observed network data by using a combination of maximum likelihood approach and Monte Carlo sampling algorithm.

Unlike previously discussed graphical models, Probabilistic Relational Model (PRM) provides a way to incorporate node and link attributes into the probabilistic model. The goal is to learn the joint probability distribution of a set of nodes and the links that connect them. The advantage of such a model is that it considers the object-relational nature of network data by capturing probabilistic interactions between entities and the links [54]. There are two possible models: relational Bayesian networks [49], where links are directed, and relational Markov networks [106], where links are undirected. The undirected model is usually preferred due to its flexibility.

3.2.3 Linear algebraic methods

Kunegis et al. proposed a method that uses graph kernels and dimensionality reduction methods to solve the link prediction problem [65]. In this method we need to learn a function F which works on the network adjacency matrix $\mathbf{A}$.

Let $\mathbf{A}$ be the adjacency matrix of the network which is built upon the training set, and $\mathbf{B}$ be the corresponding adjacency matrix for the testing dataset. We assume that both matrices have the same nodes. To solve the link prediction problem, we need to find such a spectral transformation function F which maps $\mathbf{A}$ to $\mathbf{B}$ with the minimal error By using the eigen-value decomposition, or *matrix factorization*, that is $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, we can obtain an equivalent optimization problem for some function f on real numbers:

$$\min_f \sum_i (f(\mathbf{\Lambda}_{ii}) - \mathbf{U}_{.i}^T \mathbf{B} \mathbf{U}_{.i})^2$$

This way we reduce the link prediction problem to a one-dimensional least square curve fitting problem. We can use different graph kernels for the function F, for example an exponential kernel where function F is represented as: $F(\mathbf{A}) = \sum_{i=0}^{d} \alpha_i \mathbf{A}^i$ [54].

The advantage of applying this approach is that it is quite simple and generic. However, in practice for large networks this method is computationally costly.

Menon et al. propose a model where the adjacency matrix of the network is factorized by introducing latent features [85]. Ermis et al. suggest to solve the link prediction problem in the heterogeneous network with the simultaneous factorization of several tensors where latent features are shared among each observation [38]. They treat the observed network as a relational dataset which is represented by several matrices and multiway arrays, called tensors.

3.3 Evaluation of Link Prediction

The evaluation measures for the link prediction problem are usually divided into two types: fixed threshold metrics and threshold curves. The precision and recall on top-N predictions are typical fixed threshold metrics. These measures often suffer from the

limitation that they rely upon a reasonable threshold. Another kind of metrics are the threshold curves, such as the receiver operating characteristic (ROC) curves and precision-recall curves, are widely used in link prediction evaluation. Additionally the AUROC (Areas Under ROC) is often used. AUROC can be interpreted as the probability that a randomly chosen missing link has higher score than a randomly chosen nonexistent link. It is more difficult to specify and explain link prediction evaluation strategies than with standard classification wherein it is sufficient to fully specify a dataset, therefore, new evaluation methods or performance metrics are also to be proposed.

Yang, Lichtenwalter and Chawla provide an extensive comparison of different evaluation measures for the link prediction task [119] They conclude that due to a highly imbalanced class distribution Precision-Recall curve and Area under Precision-Recall curve (AUPR) are preferred over Other works also indicate that in the case of a highly imbalanced class distribution Area under Precision-Recall curve (AUPR) is a better performance measure [32, 31, 76].

3.4 Link Prediction Problems

In Chapter 2 we have provided two definitions for the link prediction problem, namely inferring missing links and predicting future links. The second problem is often called *temporal link prediction*. We also mentioned that there are other variants of the link prediction problem which we did not cover since they are not relevant to this thesis. Nevertheless, Peng et al. differentiate such variations as: temporal link prediction, active/unactive link prediction, link prediction in bipartite networks, link prediction in heterogeneous networks, unfollow or disappearing link prediction, and link prediction scalability [111]. In this section we will provide overview of literature studying temporal link prediction and link prediction in heterogeneous networks. Both problems are addressed later on in this thesis.

3.4.1 Temporal link prediction

There are already works which investigate the temporal evolution of networks and which incorporate time information into the calculation of similarity scores. For example, Munasinghe and Ichise suggest a Time Score (TS) which combines the time and weight of links with common neighbors [86]. Their approach is based on two concepts: the strength of a link decreases with time, and the common neighbors are more effective if the interaction with them happens in a closer proximity of time. They show that the constructed score achieves better f-measures for temporal link prediction in the supervised setting.

Soares and Prudencio suggest another approach to perform temporal link prediction [30]. They do not construct one single score for a node of pairs, but rather explore the evolution of topological metrics by constructing a time series of scores. Based on the obtained time series for a specific neighborhood based score, they predict the next value of the series and use the predicted value in the link prediction model. They use both

supervised and unsupervised neighborhood based methods (PA, CN, AA, JC) on two co-authorship networks and show that by applying one of time series techniques better AUROC results can be acheived.

3.4.2 Heterogeneous networks

Many social networks contain different types of nodes and links which may follow different mechanisms of link formation and influence each other. From the practical perspective, in many cases the task of link prediction is such networks is complicated not only due to the lack of available methods, but also due to the fact that obtaining full information about attribute values in real world is difficult, and often such information is not available in the digital form at all. Thus, link prediction in heterogeneous networks is a non-trivial task. It also attracts more research efforts in the recent years: both in the link prediction community and in the general community of network analysis.

The correlation between different user behaviors and activities in heterogeneous social networks has been studied on several platforms and shown to be an important factor to understand the network properties [3, 100]. In the area of recommender systems several frameworks have been constructed to provide recommendation in heterogeneous networks [26, 63]. The mentioned studies stress the significance of investigating the mechanisms of network evolution by considering all relationship types present in the network.

Davis et al. raise the problem of link prediction in multi-relational networks [31]. In their approach they construct a weighting scheme for different edge type combinations based on the frequencies of 3-node graphlets. Davis et al. define patterns, or connected 3-node graphlets, following the theory of social balance and microscopic mechanism similar to [37], but unlike the latter work where the patterns differ in node attributes (opinion leader or not), this work constructs patterns with all possible combinations of edge types. In the following work Lichtenwalter and Chawla extend the class of graphlets and introduce a new method for link prediction called VCP [76]. Here the authors consider graphlets with more than 2 nodes which can be either undirected (U) or directed (D). In particular they conduct experiments on 11 networks using graphlets with three (VCP3U or VCP3D) and four (VCP4U or VCP4D) nodes. Their results indicate that VCP outperforms the classical features in many cases either in terms of AUROC (area under ROC curve) or AUPR (area under precision recall curve).

Yang et al. consider the temporal link prediction problem in a heterogeneous network constructed on a DBLP dataset by performing a multi-relational influence propagation [118]. To account for temporal dimension, the authors introduce several features. Firstly, they calculate recency as the time passed since a node made its last new link and activeness as the number of new links made in the last time step. These two features are somewhat similar to the time score [86]: instead of introducing time awareness for the links this work makes nodes time aware. Secondly, the authors calculate a degree preferential vector which can be perceived as a time series approach [30]. However, they use a different probabilistic technique to calculate the final feature. Finally, they design two new models, Temporal and Multi-Relational Influence Propagation model (MRIP)

models, and compare their performance against three state-of-the-art models such as HPLP, VCP3U and VCP4U. They achieve the best AUROC for the Temporal model in the supervised setting.

3.5 Link Prediction Applications

Link prediction can be used to solve a variety of problems and not just to study network evolution. In the area of Internet and web science link prediction methods are successfully applied to automatically create web hyper-links [1] and to predict web site hyper-links [121]. In e-commerce link prediction is widely applied to design recommendation systems [59, 73, 79]. It also has various applications in fields outside of Computer Science. In bioinformatics link prediction has been used to predict Protein-Protein Interaction (PPI) [4] as well as to annotate the PPI graph [44]. Another interesting application of link prediction is in the security related area. One can use link prediction to identify hidden groups of terrorists and criminals [39, 115].

CHAPTER 4

Citation Count Prediction

Publish or perish.

Logan Wilson, The Academic Man: A Study in the Sociology of a Profession

Link prediction can be applied to solve problems in different areas. In this chapter we illustrate a novel application area for link prediction methods. On one hand, we extend the applicability of link prediction methods, hence emphasizing the importance of this task. On the other hand, we contribute to the corresponding application area by providing efficient solutions to its problems. The work in this chapter was done under the supervision of Ryutaro Ichise, and appeared in the proceedings of the 27th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems 2014 [92] and in the journal on Applied Intelligence [93].

4.1 Motivation

Due to the drastic growth of the amount of scientific publications each year, it is a major challenge in academia to identify important literature among recent publications. The problem is not only how to navigate through a huge corpus of data, but also what search criteria to use. While the Impact Factor [46] and the h-index [56] measure the significance of a particular venue or a particular author, the citation count aims at estimating the impact of a particular paper. Furthermore, Beel and Gipp find empirical evidence that the citation count is the highest weighted factor in Google Scholar's ranking of scientific publications [15]. In other bibliography search systems the citation count is also considered as one of the major search criteria [16]. The drawback about using the citation count as a search criteria is that it works only for the papers which are old enough. We will not be able to judge new papers this way. To solve this problem, we

need to estimate the future citation count. An accurate estimation of the future citation count can be used to facilitate the search for promising publications.

A variety of research articles have already studied the problem of citation count prediction with differences in evaluation approaches, predictive models and features used in the models. In earlier work the researchers experimented on relatively small datasets and simple predictive models [24, 34, 64]. Nowadays due to the opportunity to retrieve data from the online digital libraries the research on citation behavior is conducted on much larger datasets. The predictive models have also become more sophisticated due to the advances in machine learning. The major challenge is the selection of features. Therefore, our goal is to discover features which are useful in the prediction of citation counts.

Previous work points out that graph mining techniques lead to good results [80]. This observation motivated us to formulate the citation count prediction task as a variation of the link prediction problem in the citation network. Here the citation count of a paper is equal to its in-degree in the network. Its out-degree corresponds to the number of references. Since out-degree remains the same over years, the appearance of a new link means that the citation count of the corresponding paper increases. In the link prediction problem we aim at predicting the appearance of links in the network. However, we do not solve the classical link prediction problem since we need to estimate only the amount of new links for a specific node, but not with which other nodes in the network it gets connected.

Our idea is to utilize frequent graph pattern mining in the citation network and to calculate a new feature based on the mined patterns – *GERscore* (Graph Evolution Rule score). Since we intend to predict the citation counts in the future, we want to capture the temporal evolution of the citation network with the graph patterns. That is why we mine frequent graph patterns of a special type - the so-called graph evolution rules [21].

The main contributions within this chapter are the following:

- We study the citation count prediction problem as a link prediction problem.
- We adopt score calculation based on the graph evolution rules to introduce a new feature GERscore for solving the citation count prediction problem, we also propose a new score calculation.
- We design an extended evaluation framework which we apply not only to the new feature, but also to several state-of-the-art features.

The rest of the chapter is structured as follows. In the next section we formulate the problem of citation count prediction. Section 4.3 covers the work related to the stated problem. In Section 4.4 we present our methodology to calculate the new feature. Section 4.5 describes our approach to evaluate the new feature. This section also includes the experimental results on two datasets followed by a discussion. Finally, we provide a summary for this chapter and point out future directions for work.

4.2 Problem

We want to predict citation counts for scientific papers. For this purpose we take the definition of the citation count problem introduced by Yan et al.[117]. Formally, we are given a set of scientific publications $\mathcal{D}$, the *citation count* of a publication $d \in \mathcal{D}$ at time t is defined as:

$$Cit(d,t) = |\{d' \in \mathcal{D} : d \textit{ is cited by } d' \textit{ at time } t\}|.$$

To achieve our goal, we need to estimate $Cit(d, t + \Delta t)$ for some $\Delta t > 0$. We can solve this task by using either classification or regression.

Classification Task: Given a vector of features $\bar{X_{d,t}} = (x_1, x_2, \ldots, x_n)$ for each scientific publication $d \in \mathcal{D}$ at time t, the task is to learn a function for predicting $CitClass(d, t + \Delta t)$ whose value corresponds to a particular range of the citation count for the publication d at the time $t + \Delta t$.

Regression Task: Given a vector of features $\bar{X_{d,t}} = (x_1, x_2, \ldots, x_n)$ for a publication $d \in \mathcal{D}$ at time t, the task is to learn a function for predicting $Cit(d, t + \Delta t)$ whose value corresponds to the citation count of the publication d at the time $t + \Delta t$.

We propose a new perspective on the citation count prediction problem. We construct a paper citation network from the set of scientific publications $\mathcal{D}$. An example of a citation network is given in Figure 4.1. Nodes represent scientific papers. A link from one node to another means that the first paper cites the latter. As we see, nodes and links have attributes which we will discuss later on. In this setting, the citation count of a paper is equal to the in-degree of the corresponding node. Its out-degree corresponds to the number of references present in the network and does not change over time. Since a node's in-degree increases if a new link appears, we can regard the citation count problem as a variation of the link prediction problem in citation networks. Generally, the link prediction problem answers the question whether there will be a link between two disconnected nodes in the network. In our case there are two major differences from the general link prediction problem. Firstly, new links are formed with the nodes which do not yet exist in the network (since the corresponding papers are not yet published). Therefore, we cannot use classical link prediction methods. Secondly, for a specific node we are not interested to identify the nodes with which it will form links in the future, rather we want to estimate the amount of such nodes. Thus, we need to construct a suitable link prediction method to estimate future citation counts for scientific publications.

4.3 Related Work

To solve the problem at hand, we build upon the works studying the citation count prediction and link prediction problems. The former works provide the baseline to compare our new feature, while the latter are used to construct it.

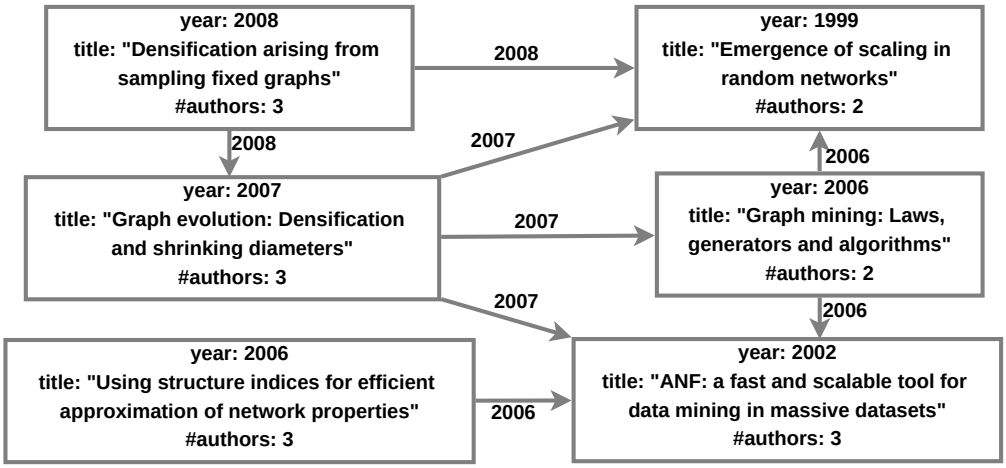


Figure 4.1: Example of a citation network.

4.3.1 Citation count prediction

The task of predicting the citation counts for scientific publications as well as the general study of citing behavior have long attracted attention in the academic world. In particular, bibliometrics is a field of information and library sciences which focuses on the statistical analysis of academic literature. Bibliometric assessment of performance for scholars, scientific publications or journals is based on the following assumptions:

1. Scholars, who have achieved significant results, do publish their findings either in conference proceedings or journals.

2. Scholars refer in their own work to the earlier work of other scholars to acknowledge intellectual debt and the use of information.

Thus, citations can be used as an indicator of impact of a publication.

For example, Callaham et al. predicted citation counts for 204 publications from the 1991 emergency medicine specialty meeting [24]. They used decision trees and showed that the journal's impact factor is the most significant factor. Kulkarni et al. studied 328 medical articles published in 1999 and 2000 [64]. By using linear regression they achieved R^2 of 0.2 for predicting citation counts five years ahead.

Nowadays with the majority of digital libraries, such as ACM, IEEE, arXiv, etc., providing access online, it is possible to retrieve data about scientific publications automatically and to conduct studies of citation behavior on a large scale. Recent studies on citation count problem are performed on much larger datasets using more sophisticated predictive models and features of papers.

Using a dataset of 30,199 papers from the arXiv, McGovern et al. suggested to predict non-self citations for a set of papers by performing a classification task of papers into quartiles $\{0-1, 2-5, 6-14, >14\}$ according to their citations [83]. When constructing a training dataset, they considered characteristics of papers, the referenced papers, authors,

number of pages and previous papers written by the authors. On several data samples the authors achieved an average classification accuracy of 44% using relational probability trees. As an outcome of their study, several patterns are outlined according to which a paper has an 85% probability of obtaining more than 14 non-self citations. For example, one of the patterns is that the paper has more than 8 references. However, the authors do not provide detailed description of the features in their prediction model as well as their performance. The main focus of the paper is to uncover interesting patterns of citing and publishing behavior in the corresponding physics community.

Yan et al. introduce the citation counts prediction task [117]. They propose several factors which correlate with citation counts. These factors are based on content, author, venue and publication year of scientific publications, e.g., they use such features as author and venue ranks. To obtain author rank, the average citation counts in the previous years for every author is determined and a rank is assigned based on this number among the other authors. Venue rank is calculated the same way using the venue of the paper instead of the authors. In the succeeding work Yan et al. extend the list of factors, but they still show that the author rank is the most influential factor among those considered [116].

In these works the authors have also compared the performance of different predictive models with Classification and Regression Tree (CART) and Gaussian Process Regression (GPR) providing higher R^2 values compared to k-nearest neighbor (kNN), support vector regression (SVR) and linear regression (LR) models. The dataset which is used in their experiments is publicly available. Yan et al. do not use any features constructed from the citation network [116, 117].

Livne et al. extract a large and diverse dataset from Microsoft Academic Search [80]. This dataset contains 38 million papers which they group into seven major academic domains: computer science, biology, chemistry, medicine, engineering, mathematics and physics. For the citation count problem they construct features based on the authors, author institutions, venue, references and content of the papers. By using SVR they show that the most significant group of features is the one based on the citation network. However, the venue factor is more significant in two out of seven domains. The authors suggest that graph mining techniques might be better suited to capture the interest of research community.

Similar results are obtained by Didegah and Thelwall when analyzing a set of papers published in nanoscience and nanotechnology journals from 2007 to 2009 [34]. They observe that the impact factor of the publication venue and of the references are the most significant determinants of the citation count.

Summarizing the results of the recent work [80, 34, 116], we come to the conclusion that properties of papers which are related either to the paper co-authorship or citation networks are among the most significant factors for the paper citation prediction task. This observation indicates that formulating this problem as a link prediction problem in the citation network might be a promising approach. None of the above mentioned works considered a link prediction method to predict future citation counts. We will show that it is possible to solve the citation count prediction problem with a link prediction method and that by doing so we improve the performance.

4.3.2 Link prediction

A known model for the link prediction task in social networks is the preferential attachment model [12]. This model assumes that new nodes are more likely to form relationships with those nodes in the network which have already high degree. This behavior creates the so called "rich-get-richer" effect. Among the other methods for link prediction in social networks, there are, for example, methods suggested by Adamic and Adar [2], and by Liben-Nowell and Kleinberg [74]. Munasinghe and Ichise introduce a time-aware feature which considerably improves the performance of classical models for link prediction [86]. However, these methods can predict links only between nodes which already exist in the network. The citation count for a given paper increases if the corresponding node in the network gets an incoming relationship from a new node.

By introducing graph evolution rules Bringmann et al. illustrate a way to predict links between an existing node and a new node in the network [21]. Their approach is based upon mining graph evolution rules in a network where links are stamped with the creation time and nodes may have up to one integer label. In our example network (see Figure 4.1) we introduce link attributes which correspond to the years when the corresponding references appear. As a possible node label, Bringmann et al. take the degree of the node. In our work we consider number of authors and number of references as possible node labels. Our choices are motivated by the results from the work of McGovern et al. [83]. The obtained rules provide an opportunity to capture the temporal evolution of the network.

Another evidence that graph mining in the citation network might lead to good results for the citation count prediction can be found in [101]. Shi et al. investigate the patterns of citations by constructing citation projection graphs. The citation projection graph of a specific publication is a subgraph of the citation network which includes the references and citations among the papers which are referenced by this publication and also cite it. The authors observe that certain properties of the projection graphs are more common for papers with high impact. The impact of a publication is measured by its citation count normalized by the average citation count for all other papers published in the same year. The publications are classified into three classes according to their impact – high, medium and low.

Though the authors apply a graph mining technique to study the citing behavior in three domains (namely, natural sciences, social sciences and computer sciences), they do not use any link prediction method. The structure of the patterns, which they uncover, is not fixed and is more or less unique for each node. Therefore, the patterns in their work are not graph patterns in the classical understanding, but rather these patterns refer to the structural properties of the citation projection graphs which differ for papers with high, medium and low impact. We, on the other hand, mine the local graph patterns which have specific properties in the whole network. These patterns have fixed structure, capture the temporal aspect of the citation counts and can be used for link prediction unlike the work of Shi et al. [101].

Thus, we suggest a new feature GERscore which is based upon frequent patterns of a specific form, i.e., graph evolution rules, mined from the citation network.

4.3.3 Evaluation

The estimation of future citations can be done with *classification* [83] or *regression* [80, 116, 117]. The classification task, where we predict intervals of citation counts, is in general easier [33, Ch. 6.7], and in many applications it is enough. For example, papers with more than 100 citations are referred to as influential in [116]. Shi et al. also study the properties of papers with regard to three classes of normalized citation counts [101]. Though both classification and regression tasks provide estimations for future citation counts in our case, there is a fundamental difference: the former estimates the probability of a paper to belong to a specific interval of citation counts, whereas the latter estimates the real citation count for this paper.

Yan et al. apply the regression task to predict future citation counts and then use the results to construct a recommender system for scientific literature [116]. We also perform the regression task to predict the exact future citation counts. Furthermore, a dataset of publications from physics is used by Mcgovern et al. [83], and from computer science by Yan et al. [116, 117]. There are also two different evaluation approaches.

The first one is to test the performance for the freshly published papers [80, 83]. The second approach is to predict the citation counts for all available papers [116, 117]. To ensure a comprehensive study of performance of our new feature and several state-of-the-art features, our evaluation framework includes both classification and regression, two evaluation approaches and two datasets of scientific publications. Furthermore, we include two performance measures for each of the learning tasks: average accuracy and precision for classification, R^2 and $RMSE$ for regression. The previous works report their results in terms of one performance measure.

To sum it up, we extend the evaluation frameworks from the previous works, and we use the works of Mcgovern et al. and Yan et al. as our baseline [83, 117].

4.4 GERscore

Our methodology to tackle the stated problem consists of several steps which are depicted in Figure 4.2. First, we construct a citation network from a publication database (block (1)). By using additional constraints, which are called maximum size and minimum support (block (2)), we mine the so-called graph evolution rules in the constructed network (block (3)). Then we derive the GERscore for each paper using several calculation techniques (block (4)). We also calculate several state-of-the-art features (block (5)). All features are obtained using data from previous years.

To estimate the performance of these features, we prepare training and testing datasets (blocks (7) and (8)) following two different scenarios (block (6)) and construct several predictive models for the classification and regression tasks (block (9)). Depending on the scenario (block (6)), we choose which publications from the database get sampled into the training and testing datasets.

In the rest of this section, we explain the process of obtaining graph evolution rules and GERscores, i.e., blocks (2)–(4). Blocks (1) and (5)–(9) are described in Section 4.5.

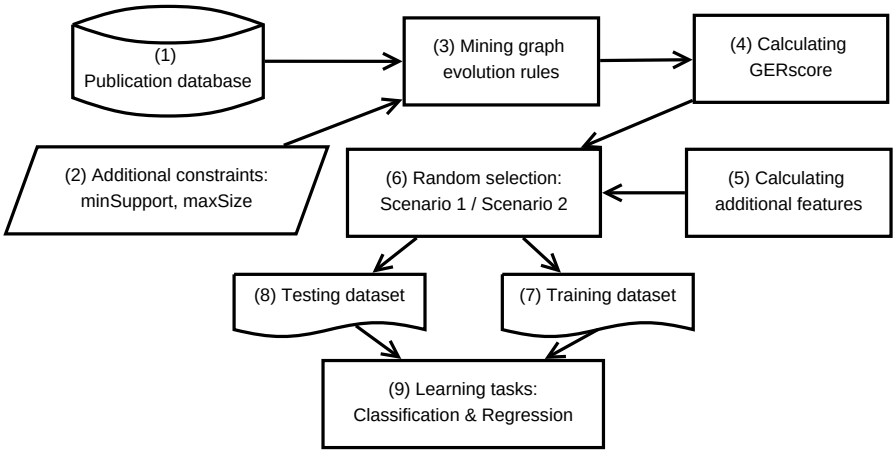


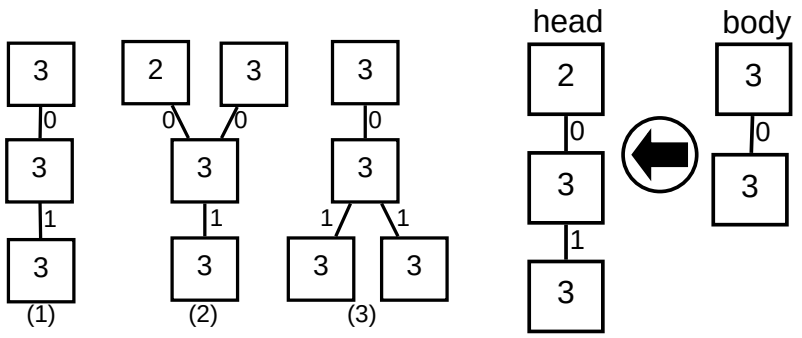
Figure 4.2: Process flow for predicting citation counts using a modified link prediction method.

4.4.1 Mining Graph Evolution Rules

To calculate the GERscore, we start with the discovery of rules which govern the temporal evolution of links and nodes. These rules are based on the frequent patterns of a special form, the so-called *relative time patterns*, and are introduced in [21]. Informally, a relative time pattern is a connected graph with one type of labels over nodes (exactly one integer label or no label at all) and one integer label over links which represents relative time. Examples of relative time patterns are given in Figure 4.3(a). We can embed this pattern into a network if we can match each node of the pattern to some node in the network by preserving node labels and the structure of links between these nodes. Additionally, link labels in the pattern should correspond with a fixed gap to the labels of the matched links in the network.

In Figure 4.1 the network is directed, but to apply the notion of relative time patterns we ignore the direction of links. Besides, we may infer the direction of links: they point from a new node towards the older one. As link attributes, we have the year of link appearance in the network which corresponds to the year of publication of the citing paper. Nodes can have various attributes in the citation network, but we focus on three possible options: no label, the number of authors and the number of references of the corresponding paper. If we consider only the number of authors as a node label in the example citation network, then the pattern in Figure 4.3(a)(1) can be embedded with the time gap 2007 or 2006 into the citation network in Figure 4.1 while the pattern in Figure 4.3(a)(3) cannot be embedded at all.

A *graph evolution rule* is a pair of relative time patterns called *body* and *head* which is denoted as $head \Leftarrow body$ [21]. Informally, the *body* can be represented as the *head* without links which have the highest label. An example of a graph evolution rule is given in Figure 4.3(b). Do not get confused by the fact that body has less links than head. The naming convention follows the one used for rules in logic. Considering the



(a) Relative time patterns. (b) Graph evolution rule.

Figure 4.3: Examples of relative time patterns and graph evolution rules. Node labels correspond to the number of authors.

definition of the evolution rule, we can represent any evolution rule uniquely with its head. That is why relative time patterns in Figure 4.3(a)(1)-(3) are also graph evolution rules. However, the relative time pattern in Figure 4.3(a)(4) cannot be regarded as an evolution rule since both link labels equal zero.

To estimate frequency of the relative time pattern in the given network, we use *minimum image-based support*. It roughly equals to the minimum number of different nodes in this network to which one of the nodes in the pattern can be matched. The *support* of the evolution rule, $sup(r)$, is equal to the support of its head. The *confidence* of this rule, $conf(r)$, is equal to the ratio of the supports of the head and the body.

The graph evolution rule from Figure 4.3(b) has a minimum image based support 2 in the citation network from Figure 4.1. The support of its body is also 2. Therefore, confidence of this rule is 1. We can interpret this rule the following way: if the body of this rule embeds into the citation network to a specific node at time t, then this node is likely to get a new citation at time $t + 1$. We assume that the likelihood of such event is proportional to the confidence of the rule. To determine all graph evolution rules in a network, we need to employ a graph pattern mining procedure.

Since graph pattern mining is computationally hard, two additional constraints are used to speed up the process of mining graph evolution rules in a network. We mine only those rules which have support not less than *minSupport*, and which have number of links not more than *maxSize*. The higher *minSupport* or the lower *maxSize*, the faster the graph pattern mining process will finish and the less patterns we will obtain. In case we have node labels in the network, we will also often arrive at better running times compared to the case when no labels are used over the nodes. Among the uncovered patterns, we identify graph evolution rules. In other words, we look for the patterns which have at least two different values on the links. Furthermore, we consider only those graph evolution rules where body and head differ in one link. In Figure 4.3 all rules, except for a(3), correspond to this condition. Finally, we obtain a set $\mathcal{R}$ of graph evolution rules.

4.4.2 Calculating GERscore

To calculate the GERscore, we modify the procedure from [21]. We need to do this modification since the suggested approach in the previous work is a link prediction method, whereas we need to adapt it to our problem. The main task is to aggregate the information about the obtained graph evolution rules for a scientific publication. For each publication n in the citation network we identify rules from the set $\mathcal{R}$ which can be applied to it. We say that a graph evolution rule can be applied to the node n if its body can be embedded into the network so that one of the matched nodes is n. We obtain a set $\mathcal{R}_n \subset \mathcal{R}$ of rules applicable to the node n. Our assumption is that an evolution rule occurs in the future proportional to its confidence. That is why we put the GERscore equal to $c * conf(r)$, where c measures the proportion of rule's applicability.

We define three ways to calculate c. In the first case, we simply take $c = 1$. In the second case, we assume that evolution rules with higher support are more likely to happen, i.e., $c = sup(r)$. These two scores are also used for the link prediction problem in [21]. Lastly, if the evolution rule r contains more links, it provides more information relevant to the node n. We assume that such rule should be more likely to occur than the one with less edges. Since evolution rules are limited in their size by $maxSize$, we put $c = size(r)/maxSize$. Thus, we obtain three different scores:

1. $score_1(n, r) = conf(r)$,
2. $score_2(n, r) = sup(r) * conf(r)$, and
3. $score_3(n, r) = conf(r) * (size(r)/maxSize)$.

In the previous work the authors also experiment with different score calculation techniques, and they show that the best results for the link prediction problem are obtained by using $score_2(n, r)$ [21]. However, we will still run experiments with all three scores since we solve a different problem.

Finally, we use two functions to accumulate the final GERscore for the node n:

- $\text{GERscore}_{1,i}(n) = \sum_{r \in \mathcal{R}_n} score_i(n, r)$,
- $\text{GERscore}_{2,i}(n) = \max_{r \in \mathcal{R}_n} score_i(n, r)$.

Here $score_i(n, r)$ corresponds to one of three possible score calculations. Therefore, we obtain six possible scores for our new feature. Throughout the paper, whenever we use the word "score", we always refer to one of the possible calculations for the GERscore.

Both aggregation techniques, maximum and summation, are used in [21]. The authors show that summation leads to better results. Though it might be intuitive to select the rule with the maximum score (which corresponds to the usage of the maximum as an aggregation function), but taking into consideration all rules, which can be applied to the node, might provide better estimation about the evolution. However, if it turns out that graph evolution rules with the highest support are the determinants of future citations, it has good implications in the sense that we can set the support threshold for the graph pattern mining procedure very high, thus reducing the running time.

High values of the GERscore can mean two things: either many rules or rules with very high confidence measures are applicable to the node. In either case, the assumption is that this node is very likely to get a high amount of citations. We may have the situation when different rules correspond to the appearance of the same link. For example, in Figure 4.3 rules (b) and (a1) are subgraphs of rule (a2). It might happen that these rules correspond to the creation of the same link. Still we consider all three rules, since we are interested to approximate the likelihood of increase in citation counts. With this regard, the constructed GERscore is similar to the network measures discussed in the work of Shi et al. [101]. However, our feature is based on a link prediction method which makes it distinct from the measures in this work.

4.5 Experimental Results

4.5.1 Experimental Data

We use two real datasets to evaluate the GERscore: *HepTh* and *ArnetMiner*. The first dataset covers arXiv papers from the years $1992 - 2003$ which are categorized as High Energy Physics Theory [83]. We mine graph evolution rules for the network up to year 1996 which has 9,151 nodes and 52,846 links. The second dataset contains papers from major Computer Science publication venues [117]. By taking papers up to year 2000, we obtain a sub-network with 90,794 nodes and 272,305 links.

We introduce two additional properties for papers: *grouped number of references* and *grouped number of authors*. For the first property the intervals are $0-1, 2-5, 6-14, 15 \leq$. The references here do not correspond to all references of the paper, but only to those which are found within the dataset. We select the intervals $1, 2, 3, 4-6, 7 \leq$ for the second property.

We construct several graphs from the described sub-networks which differ in node labels. It is good to have a label over nodes because this speeds up graph pattern mining. Since we are not sure which label setting is better, we use either the grouped number of references, or the grouped number of authors, or no label. The choice of the first two label settings is motivated by the uncovered citing patterns in [83]. Bringmann et al. show that graph evolution rules with no node labels lead to good results when solving the link prediction problem [21]. Also, they use the node degree as a label to obtain labeled graph evolution rules. In our case, it makes more sense to use the out-degree (or the number of references) since it does not change over time. Since it does not make sense to have continuous values as node labels (possible rules will be too rare and their interpretation will get harder), we group the values into categories. Bringmann et al. use also the grouped values of node's degree as labels [21].

In Table 4.1 we show the amount of graph evolution rules obtained with the help of the tool *GERM* [1] for different label settings for our two datasets. It is clear that the amount of evolution rules is considerably smaller than the amount of mined frequent patterns: not every relative time pattern is a graph evolution rule and we consider only graph

[1] http://www-kdd.isti.cnr.it/GERM/

Table 4.1: Results of graph pattern mining.

Network	*Setting*	*min support*	*max size*	*# patterns*	*# evolution rules*
Hep-Th	no label	1,000	5	1,412	230
upto	# authors	500	5	7,441	886
1996	# references	500	5	6,565	426
Arnet	no label	5,000	5	6,742	4,108
upto	# authors	2,000	5	4,838	968
2000	# references	2,000	5	4,366	1,004

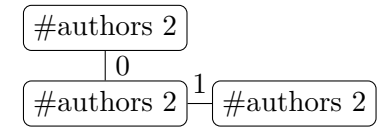
evolution rules where body and head differ in one link. We obtain 230 evolution rules in the dataset HepTh, and 4,108 in the dataset ArnetMiner for the unlabeled case (when no label over the nodes is used). We have 886 rules in HepTh, and 968 in ArnetMiner for the grouped number of authors. For the grouped number of references the numbers are 426 and 1,004 correspondingly. For both datasets we mine rules of the size up to five. However, support thresholds are set different since the datasets have considerably different amount of nodes. In our experiments we identified that this combination of input parameters is feasible enough to obtain results within one month for both datasets. The most crucial parameter is the size of the evolution rules, and it drastically affects the running times.

Figure 4.4 contains examples of graph evolution rules which we obtain for the citation network with grouped # authors as node labels. As we mention earlier, there are two main measures to estimate the frequency for each evolution rule: support and confidence. Thus, Figures 4.4(a) and 4.4(b) contain the rules which have the highest support in HepTh and ArnetMiner correspondingly. In both cases the rules have the same structure and same node labels, but they have different supports and confidence measures: a lower support in HepTh than in ArnetMiner, but a higher confidence at the same time. However, the rules with the highest confidences are different for our datasets (see Figures 4.4(c), 4.4(d)). Though they both have five links, they differ in structure, node and link labels. Furthermore, the rule for ArnetMiner (Figure 4.4(d)) has higher support as well as higher confidence. Such information already indicates that there are differences in the temporal evolution of the considered citation networks. Firstly, the amount of mined rules is considerably less for HepTh. Secondly, the differences in confidence measures will affect the probabilities of link formation.

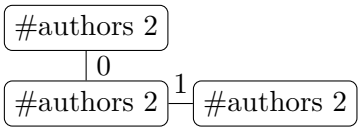
4.5.2 Experimental Setting

For a comprehensive study we perform two experiments. In the first experiment we aim at classifying papers into quartiles according to the future citation counts. We consider the following models for the classification task:

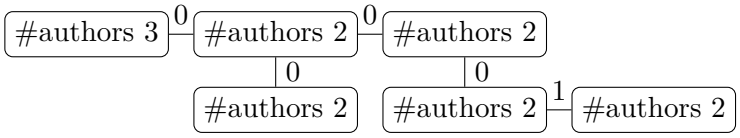
1. Multinomial Logistic Regression (mLR) which is a generalization of logistic regres-



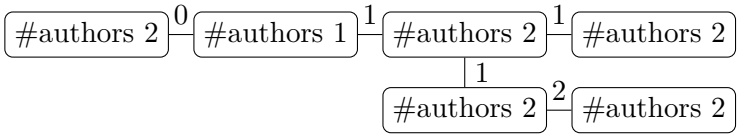
(a) support: 926, confidence: 0.342;



(b) support: 4336, confidence: 0.192;



(c) support: 619, confidence: 0.600;



(d) support: 2026, confidence: 0.767;

Figure 4.4: Examples of graph evolution rules mined from HepTh and ArnetMiner datasets using number of authors as node label: (a) rule with highest support for HepTh, (c) rule with highest confidence for HepTh, (b) rule with highest support for ArnetMiner, (d) rule with highest confidence for ArnetMiner.

sion for the case of more than two discrete outcomes,

2. Multi-class Support Vector Machines (mSVM) which construct a hyperplane or a set of hyperplanes to separate the training instance with the largest distance to the nearest data point of any class [25],

3. Conditional Inference Trees (CIT) which recursively perform univariate splits of the dependent variable and use a significance test to select variables [58].

We predict the real future citation counts for papers in the second experiment. Here, we consider such models for the regression task:

1. Linear Regression (LR) which approximates the dependent variable linearly based on the independent variables and intercept,

2. Support Vector Regression (SVR) which is an adaptation of SVM to perform the regression task [25],

3. Classification and Regression Tree (CART) which recursively perform univariate splits of the dependent variable and use the Gini coefficient to select variables [19].

We use the implementation of these models in R [97]. We look at a variety of models because they make different assumptions about the original data. Therefore, it is not guaranteed that features perform equally well in different models.

For each of the learning tasks (experiments) we consider two scenarios for evaluation which differ in the way we construct training and testing datasets. In Scenario 1 we train the models on the papers published one year before. A similar evaluation approach is undertaken in [80, 83, 86]. In Scenario 2 the training and testing datasets are constructed on the same pool of papers, for example, as it is done by Yan et al. in their works [116, 117].

In both scenarios we use a slightly modified 5×2 cross-validation [35], where each fold contains a stratified selection of scientific publications, i.e., 1,000 instances for HepTh and 10,000 for ArnetMiner. We choose such approach to remain in line with the previous works where 10,000 papers are chosen both into the training and testing datasets for ArnetMiner [116, 117]. We do not perform complete cross-validation procedure in Scenario 1: the training dataset contains papers from the year t and the testing dataset has papers from the year $t+1$, so changing the training and testing datasets does not make sense here. We use $\Delta t = 1$ both for classification and regression tasks which corresponds to the prediction of citation counts for the next year. Additionally, we perform five year prediction in the case of the regression task to provide a more comprehensive comparison to the previous works.

To compare performance of our new feature, we calculate several state-of-the-art features: *Author Rank*, *Total Past Influence for Authors (TPIA)* (TPIA), *Maximum Past Influence for Authors (MPIA)*, *Venue Rank*, *Total Past Influence for Venue (TPIV)*, *Maximum Past Influence for Venue (MPIV)* and *Recency* [116, 117]. To obtain Author Rank, for every author we calculate the average citation counts in the previous years and assign a rank among the other authors based on this number. We identify the author with the maximum citation counts in the previous years and put this total citation count as MPIA for the paper. TPIA is equal to the sum of citation counts for the previous papers of the authors. Venue Rank, TPIV and MPIV are calculated the same way using the venue of the paper. Recency is the absolute difference in years between the publication and current years, and it is used only in Scenario 2. Though recency is not a good feature, we want to verify its performance in the classification task. Livne et al. introduce several features based on the references of the paper in their work [80]. We do not use all the features since we do not aim at constructing a comprehensive model for citation count prediction, but we rather want to show the viability of our new feature for this task. Since we are not sure which feature performed the best in their work, we select three features. Following their work and preserving the naming convention of the earlier features, we calculate *References Rank*, *Total Past Influence for References* (TPIR) and *Maximum Past Influence for References* (MPIR). All these features are used as the baseline to compare our new feature.

In total, we obtain 18 different scores for each paper: $GERscore_{1,i}^{(j)}$ for summation and $GERscore_{2,i}^{(j)}$ for maximum, where i equals 1, 2, or 3 depending on the score calculation, and j corresponds to a specific label setting:

$j = 1$ corresponds to the grouped number of authors as node labels;

$j = 2$ stands for the unlabeled case;

$j = 3$ is for the grouped number of references.

We report results only for one score for each label setting, because the scores exhibit similar behavior. Since our new score $score_3$ provides slightly better results, we choose the $GERscore^{(j)}_{1,3}$ and $GERscore^{(j)}_{2,3}$. Additionally, feature *GERscore* is the combination of our new 18 scores.

In Figure 4.5 we illustrate the dependence between the features author rank, venue rank, recency and GERscore on one hand and average citation counts on the other hand for our two datasets. There is a similar dependence between author ranks and average citation counts in HepTh and ArnetMiner datasets (see Figures 4.5(a) and 4.5(d)). The dependence is clearer for the dataset ArnetMiner: papers with higher rank (which corresponds to the lower value of the variable author rank) have on average higher citation counts. However, there is even more obvious dependence between venue rank and average citation count for both datasets: papers with higher venue rank (which corresponds to the lower value of the variable venue rank) have considerably higher citation counts. The dependence between recency and average citation counts does not seem to be of a specific character (see Figures 4.5(c) and 4.5(f)) which can cause its poor performance in the learning tasks. The last feature, $GERscore^2_{1,3}$, shows an inverse trend compared to Author Rank and Venue Rank: papers with a higher score have on average higher citation count. This observation supports the intuition of the score construction.

4.5.3 Classification task

In this experiment we compare how the calculated features perform with regard to classifying academic publications according to future citation counts. We assign class labels with intervals $1, 2-5, 6-14, >14$ of citation counts. Such intervals are chosen in correspondence to the previous work [83]. There the classes were specified for HepTh dataset. It might occur that such distribution is not optimal for ArnetMiner. Shi et al. define classes dynamically for each dataset [101]. Such approach ensures that class distribution is the same across different datasets, but the disadvantage is that class boundaries are not fixed and may vary even over time. Therefore, we take the approach from the work of McGovern et al. [83]. In Table 4.2 we summarize the distribution of instances according to the chosen classes for the training and testing datasets. As we see, it is the case that the class distribution is extremely skewed for ArnetMiner, especially in Scenario 1. We do not change the intervals because we want to have the same setting for both datasets. To construct training and testing datasets, we randomly select 1,000 papers from Year 1996 into the training dataset, and from Year 1997 into the testing dataset in Scenario 1 for HepTh. 1,000 instances are selected from Year 1997 into the training data in Scenario 2 for HepTh, and another 1,000 instances are selected from the rest into the testing data. For ArnetMiner the procedure is the same, except that we select 10,000 papers from years 2000 and 2001 correspondingly. In all cases we construct stratified folds and repeat the procedure 5 times.

We use *average accuracy* and *precision* to evaluate the performance in the classification task. If we put tp_i true positives, tn_i true negatives, fp_i false positives, and fn_i false

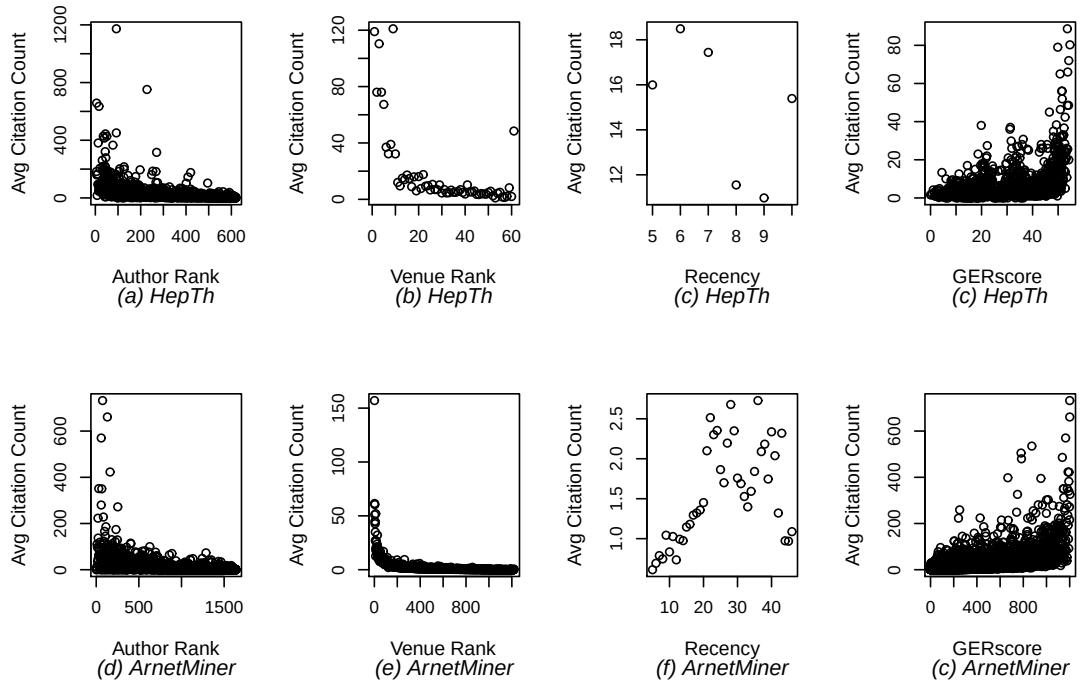


Figure 4.5: Correlation between average citation count and features: author rank, venue rank, recency and $GERscore^2_{1,3}$.

negatives for class i, then average accuracy of the classifier is:

$$Accuracy = \frac{1}{l} * \sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fp_i + fn_i + tn_i}.$$

If class distribution is unbalanced, then precision is better suited for the evaluation [103]:

$$Precision = \frac{1}{l} * \sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}.$$

The performance of the features for the classification task is presented in Tables 4.3 and 4.4. We report average accuracy and precision for the new feature GERscore and the baseline features: Author Rank, MPIA, TPIA, Venue Rank, MPIV, TPIV, References Rank, MPIR, TPIR and Recency. We mark in bold the features which lead to the highest performance measure in each column. In both scenarios we obtain that the highest accuracy and precision are for the GERscore.

However, due to a highly unbalanced distribution (Table 4.2), we observe only 1% advantage in accuracy for ArnetMiner in Scenario 2 and almost none in Scenario 1. Moreover, we obtain that the average accuracy for ArnetMiner is very high and there is

Table 4.2: Distribution of instances according to classes (% Total).

Citation Class	**HepTh**			**ArnetMiner**		
	Scenario 1		*Scenario 2*	*Scenario 1*		*Scenario 2*
	Year 1996	Year 1997	Year 1997	Year 2000	Year 2001	Year 2001
Class 1	42.9%	40.33%	34.09%	97.27%	96.69%	88.86%
Class 2	29.81%	26.64%	30.32%	2.51%	3.13%	7.75%
Class 3	13.70%	18.77%	19.85%	0.19%	0.18%	2.40%
Class 4	13.58%	14.27%	15.74%	0.03%	0.01%	0.99%
Total Amount	2,459	2,579	12,113	30,000	25,919	399,647

Table 4.3: Accuracy (%) and Precision (%) for the different features for the Classification Task in Scenario 1.

	Feature	Accuracy						Precision					
		HepTh			**ArnetMiner**			**HepTh**			**ArnetMiner**		
		mLR	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*
New	*GERscore*	**75.93**	**75.35**	**75.3**	**98.35**	**98.35**	98.34	**44.67**	**36.54**	**39.65**	**38.91**	**36.29**	**31.33**
Baseline	*Author Rank*	73.86	73.95	73.75	98.31	98.34	98.34	32.2	33.64	28.49	24.17	24.17	24.17
	MPIA	73.76	73.39	72.99	98.31	98.34	98.34	33.46	27.9	31.59	24.17	24.17	24.17
	TPIA	73.91	73.86	73.09	98.31	98.34	98.34	34.84	32.66	31.94	24.17	25.84	24.17
	Venue Rank	71.37	71.74	71.9	98.31	98.34	98.34	29.8	26.98	22.97	24.17	24.17	24.17
	MPIV	66.44	69.83	63.24	98.31	98.34	98.34	25.23	12.52	20.72	24.17	24.17	24.17
	TPIV	70.87	69.82	67.93	98.31	98.34	98.34	27.89	22.39	22.48	24.17	24.17	24.17
	References Rank	70.8	69.14	71.72	98.31	98.34	98.34	22.03	23.66	24.68	24.17	24.17	24.17
	MPIR	72.9	71.79	71.76	98.31	98.34	98.34	29.3	26.3	25.8	28.19	24.17	24.17
	TPIR	73.69	73.31	72.25	98.31	98.34	98.34	28.69	28.38	32.51	28.35	29.17	24.17

Table 4.4: Accuracy (%) and Precision (%) for the different features for the Classification Task in Scenario 2.

	Feature	Accuracy						Precision					
		HepTh			**ArnetMiner**			**HepTh**			**ArnetMiner**		
		mLR	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*
New	*GERscore*	**76.95**	**75.07**	**76.35**	**95.81**	**95.51**	**95.54**	**51.32**	**47.16**	**52.27**	**67.91**	**66.58**	**66.13**
Baseline	*Author Rank*	72.88	73.3	72.97	94.18	94.44	94.39	41.85	43.32	42.03	38.77	29.51	30.38
	MPIA	70.1	70.88	70.79	94.4	94.43	94.43	35.84	34.13	30.81	33.56	24.72	22.22
	TPIA	70.95	71.36	71.47	94.43	94.43	94.42	38.49	35.89	36.9	22.22	22.22	23.26
	Venue Rank	69.98	70.09	69.98	94.43	94.43	94.44	28.47	29.23	27.93	22.22	22.22	27.23
	MPIV	68.79	69.08	69.02	94.31	94.41	94.43	22.88	27.97	21.28	24.29	22.22	23.16
	TPIV	69.74	69.74	70.01	94.43	94.43	94.43	27.21	27.21	27	22.22	22.22	22.22
	Recency	67.77	67.36	67.37	94.39	94.3	94.4	16.78	16.79	13.27	22.17	22.17	22.16
	References Rank	69.01	68.91	69.04	94.36	94.43	94.43	26.94	24.91	31.28	27.47	22.22	22.22
	MPIR	69.06	69.08	69.23	94.38	94.41	94.43	25.21	27.97	26.5	28.74	28.51	22.22
	TPIR	69.4	69.83	69.73	94.4	94.43	94.43	28.55	30.51	30.74	34.15	31.48	23.17

almost no difference in the performance measures for different features. If the classifier puts all observations into the first class in Scenario 1, we arrive already at an average accuracy of around 97% and a precision rate of about 24%. Therefore, it is rather difficult to draw conclusions about the performance of the studied features on ArnetMiner dataset.

In the case of HepTh, the GERscore is at least 2% better in accuracy than the other features in both scenarios. The increase is more obvious in terms of precision. Recency, as expected, is not good for predicting future citation counts.

The full model with all considered features as independent variables is indicated in the row *"All"* in Tables 4.5 and 4.6. To be more precise, this model contains all features

Table 4.5: Accuracy (%) and Precision (%) for the full model with and without the GERscore for the Classification Task in Scenario 1.

	Accuracy						Precision					
Model	**HepTh**			**ArnetMiner**			**HepTh**			**ArnetMiner**		
	mLR	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*
All	77.08	74.85	75.8	98.37	98.36	98.36	50.05	47.62	40.83	41.84	35.31	31.7
-GERscore	74.69	70.44	73.53	98.31	98.35	98.33	43.74	36.24	37.19	24.17	30.01	26.79

Table 4.6: Accuracy (%) and Precision (%) for the full model with and without the GERscore for the Classification Task in Scenario 2.

	Accuracy						Precision					
Model	**HepTh**			**ArnetMiner**			**HepTh**			**ArnetMiner**		
	mLR	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*	*mLR*	*mSVM*	*CIT*
All	80.92	80.62	79.28	96.22	95.91	96.01	60.42	59.76	56.68	69.17	67.56	67.52
-GERscore	74.71	74.89	74.41	94.6	94.71	94.71	48.13	48.05	46.25	49.93	58.48	50.61

listed in Table 4.3 in Scenario 1 and all features listed in Table 4.4 in Scenario 2. To verify how much the GERscore improves the performance, we construct a full model without the new feature denoted as *"-GERscore"* in the tables.

Statistical analysis shows that the GERscore provides a significant improvement to the full model. The average accuracy does not improve considerably if we add the GERscore: for HepTh around 2% increase in Scenario 1 and less than 6% in Scenario 2; for ArnetMiner the increase is less than 2%. But if we compare precision rates, then we have that the full model with the GERscore ("All") is more than 10% better than without it ("-GERscore"). The best achieved accuracy for HepTh in Scenario 1 is 44% in previous work [83]. The accuracy of our full model mLR is 81%, but we cannot directly compare to the previous work since we take into account self-citations and the sampling technique is different. Still we see that constructed classification models provide accurate and rather precise results for both datasets. Suppose we are to identify relevant literature, then we could use classification as the first step to select potential papers and then perform regression on the selected papers to obtain a better ranking.

Overall, the results of the classification task indicate that the new feature is better than the baseline features and significantly improves the full model. Statistical significance of the improvement has been identified with analysis of variance (ANOVA) conducted for two models, "All" and "-GERscore", since these models are built on the same datasets. Surprisingly, mLR turns out to be the best performing method.

4.5.4 Regression task

In the second experiment we compare how the constructed features perform with regard to predicting real values of future citation counts for academic publications. To evaluate the performance in this task, we calculate the R^2 value as the *square of Pearson correlation coefficient* between the actual and predicted citation counts:

$$R^2 = \frac{1}{n-1}\sum_{i=1}^{n}\Big(\frac{X_i-\bar{X}}{s_X}\Big)\Big(\frac{Y_i-\bar{Y}}{s_Y}\Big),$$

where n is the sample size, $X = (X_1, ..., X_n)$ correspond to the real citation counts, $\bar{X}$ is the mean of X, s_X is the standard deviation of X, $Y = (Y_1, ..., Y_n)$ correspond to the predicted citation counts, $\bar{Y}$ is the mean of Y, and s_Y is the standard deviation of Y. R^2 measures how good the constructed model is in relative terms.

To measure the model's fitness also in absolute terms, we calculate the Root of Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\Big(X_i - Y_i\Big)^2}.$$

R^2 is a value between 0 and 1 while the value of $RMSE$ depends on the units of the response variable (in our case on the real citation counts). That is why we can compare R^2 for ArnetMiner and HepTh, but not $RMSE$: the ranges of the citation counts are quite different for them. R^2 measures how good the model is fit, hence, bigger values correspond to a better model. On the other hand, $RMSE$ estimates the non-fit of the model, therefore, it increases if the model becomes worse. Hence, we expect that the model with a higher R^2 would have a lower $RMSE$.

We summarize the performance of various features for the regression task in Tables 4.7, 4.8 in terms of R^2 and in Tables 4.9, 4.10 in terms of $RMSE$. The results are indicated for 1-year ($\Delta t = 1$) and 5-year prediction ($\Delta t = 5$). Again, we mark with bold font those features which give the highest performance measure in each column. If a feature has "NA" as a value for R^2, it means we are not able to calculate it because the standard deviation of the predicted citation counts is zero.

The GERscore leads to better R^2 values than the baseline features for ArnetMiner dataset. We showed that the GERscore is also the best performing feature for HepTh dataset in the classification task. However, it is no longer true in the regression task. An author-related feature, TPIA, yields the best R^2 results in Scenario 1 for HepTh in all cases (see Table 4.7).

Now if we examine Scenario 2, the best models (LR and SVR) are still constructed with the author-related features (see Table 4.8). But if we use CART as a learning method, we arrive at a better R^2 with our new feature than with the others. Interestingly, the results of $RMSE$ are not always coherent with the results of R^2. Depending on a learning method, we arrive at a lower $RMSE$ using our new feature (e.g., see Table 4.9). But it is still true that the lowest $RMSE$ is obtained for the author-related features using SVR.

Though author related features result in higher R^2 for HepTh, we obtain that the GERscore still brings additional value to the full models (Tables 4.11 and 4.12). The improvement is less obvious in Scenario 1, especially for the 5-year prediction for HepTh. To verify the statistical significance of the improvement, we conduct ANOVA for two models, "All" and "-GERscore". The analysis shows that the GERscore improves significantly the full model in all cases.

We also observe that it becomes harder to predict citation counts over longer periods. The only exception is ArnetMiner dataset in Scenario 2. In the previous work the authors also showed better performance over longer time periods in this setting [117].

Table 4.7: Performance measure R^2 for the different features for the Regression Task in Scenario 1.

	$\Delta t = 1$						$\Delta t = 5$					
Feature	**HepTh**			**ArnetMiner**			**HepTh**			**ArnetMiner**		
	LR	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*
$GERscore^{(1)}_{1,3}$	0.01	0.033	0.072	0.02	0.014	0.015	0.01	0.026	0.044	0.028	0.026	0.019
$GERscore^{(2)}_{1,3}$	0.075	0.11	0.122	0.111	0.095	0.073	0.06	0.083	0.084	0.123	0.13	0.101
$GERscore^{(3)}_{1,3}$	0.01	0.033	0.106	0.147	0.129	0.158	0.004	0.038	0.07	0.121	0.122	0.103
$GERscore^{(1)}_{2,3}$	0.003	0.017	0.023	0.026	0.029	0.04	0.002	0.015	0.015	0.022	0.028	0.022
$GERscore^{(2)}_{2,3}$	0.006	0.006	NA	0.108	**0.231**	**0.232**	0.004	0.004	NA	0.108	0.153	**0.166**
$GERscore^{(3)}_{2,3}$	0.067	0.111	0.104	0.109	0.131	0.182	0.04	0.073	0.069	0.117	0.125	0.123
GERscore	0.161	0.152	0.142	**0.221**	0.215	0.19	0.111	0.107	0.093	**0.165**	**0.155**	0.124
Author Rank	0.224	0.089	0.155	0.005	NA	0.003	0.183	0.076	0.136	0.008	0.004	0.009
MPIA	0.258	0.188	0.176	0.003	0.003	0.005	0.236	0.166	0.176	0.005	0.007	0.012
TPIA	**0.275**	**0.219**	**0.193**	0.001	0.001	0.01	**0.249**	**0.189**	**0.195**	0.002	0.006	0.013
Venue Rank	0.041	0.052	0.055	0.018	NA	0.029	0.03	0.054	0.035	0.063	0.062	0.063
MPIV	0.04	0.011	0.031	0.002	NA	0.034	0.037	0.001	0.027	0.004	0.011	0.035
TPIV	0.048	0.01	0.032	0.002	NA	0.021	0.037	0.002	0.022	0.001	0.026	0.027
References Rank	0.092	0.012	0.035	0.004	NA	0.029	0.073	0.004	0.029	0.02	0.023	0.026
MPIR	0.092	0.083	0.074	0.005	0.001	0.037	0.071	0.044	0.053	0.004	0.031	0.035
TPIR	0.076	0.1	0.095	0.008	0.001	0.037	0.061	0.066	0.058	0.006	0.028	0.033

Table 4.8: Performance measure R^2 for the different features for the Regression Task in Scenario 2.

	$\Delta t = 1$						$\Delta t = 5$					
Feature	**HepTh**			**ArnetMiner**			**HepTh**			**ArnetMiner**		
	LR	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*
$GERscore^{(1)}_{1,3}$	0.062	0.072	0.121	0.154	0.172	0.173	0.055	0.065	0.099	0.236	0.224	0.275
$GERscore^{(2)}_{1,3}$	0.126	0.226	**0.247**	0.382	**0.374**	0.428	0.106	0.17	**0.176**	0.447	**0.448**	0.472
$GERscore^{(3)}_{1,3}$	0.019	0.017	0.009	0.186	0.193	0.228	0.013	0.012	0.011	0.226	0.243	0.258
$GERscore^{(1)}_{2,3}$	0.027	0.036	0.049	0.066	0.092	0.101	0.024	0.033	0.045	0.095	0.128	0.132
$GERscore^{(2)}_{2,3}$	0.033	0.059	0.058	0.086	0.139	0.186	0.031	0.058	0.058	0.114	0.171	0.215
$GERscore^{(3)}_{2,3}$	0.006	0.015	NA	0.093	0.102	0.114	0.005	0.011	0.013	0.128	0.138	0.154
GERscore	0.188	0.201	0.217	**0.445**	0.298	**0.444**	0.14	0.161	0.157	**0.543**	0.355	**0.483**
Author Rank	0.201	**0.264**	0.179	0.118	0.121	0.17	0.164	0.21	0.17	0.133	0.181	0.159
MPIA	0.235	0.235	0.21	0.061	0.055	0.067	0.19	0.207	0.147	0.07	0.025	0.054
TPIA	**0.303**	0.239	0.236	0.002	0.067	0.056	**0.25**	**0.216**	0.167	0.004	0.062	0.071
Venue Rank	0.056	0.065	0.051	0.035	0.057	0.053	0.043	0.063	0.049	0.04	0.048	0.05
MPIV	0.046	0.056	0.044	0.031	0.027	0.035	0.032	0.043	0.038	0.025	0.009	0.039
TPIV	0.047	0.053	0.038	0.02	0.023	0.035	0.035	0.04	0.036	0.017	0.006	0.035
Recency	0.002	0.002	NA	0.006	NA	NA	0.002	0.003	0.004	0.002	NA	NA
References Rank	0.096	0.083	0.059	0.026	0.016	0.023	0.094	0.086	0.034	0.025	0.009	0.017
MPIR	0.094	0.088	0.086	0.018	0.022	0.02	0.098	0.092	0.065	0.018	0.014	0.019
TPIR	0.117	0.096	0.065	0.026	0.022	0.026	0.109	0.096	0.057	0.026	0.012	0.018

Table 4.9: Performance measure $RMSE$ for the different features for the Regression Task in Scenario 1.

	$\Delta t = 1$						$\Delta t = 5$					
Feature	**HepTh**			**ArnetMiner**			**HepTh**			**ArnetMiner**		
	LR	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*
$GERscore^{(1)}_{1,3}$	16.068	16.294	15.723	0.668	0.668	0.672	39.722	39.891	39.814	1.959	1.966	1.966
$GERscore^{(2)}_{1,3}$	15.495	15.864	**15.192**	0.633	0.639	0.702	**38.85**	39.301	**38.826**	1.848	1.834	2.034
$GERscore^{(3)}_{1,3}$	16.002	16.566	15.552	**0.624**	0.628	0.669	39.646	40.197	39.481	1.876	1.866	1.982
$GERscore^{(1)}_{2,3}$	16.062	16.624	15.915	0.677	0.662	0.663	39.702	40.383	39.554	1.97	1.967	1.964
$GERscore^{(2)}_{2,3}$	16.015	16.668	16.057	0.645	0.597	**0.583**	39.533	40.515	39.589	**1.834**	1.824	**1.811**
$GERscore^{(3)}_{2,3}$	15.729	15.739	15.566	0.625	0.634	0.631	39.58	39.063	39.504	1.844	1.869	1.881
GERscore	**15.317**	15.004	15.689	0.625	**0.593**	0.651	39.516	37.97	40.22	1.88	**1.811**	1.995
Author Rank	16.742	15.522	16.261	0.674	0.672	0.689	41.791	38.494	44.437	1.973	2.004	1.986
MPIA	17.892	14.683	16.301	0.677	0.671	0.68	48.113	**36.809**	43.368	1.979	1.999	1.994
TPIA	20.059	**14.636**	18.913	0.673	0.672	0.687	53.046	37.625	49.093	1.966	1.999	2.003
Venue Rank	15.941	16.297	15.753	0.667	0.672	0.672	39.768	39.711	39.478	1.875	1.949	1.951
MPIV	18.887	16.662	16.189	0.673	0.672	0.672	47.044	40.418	40.378	1.988	1.991	1.946
TPIV	18.21	16.369	16.355	0.673	0.672	0.674	45.499	40.167	41.119	1.983	1.984	1.981
References Rank	17.847	16.344	16.593	0.674	0.671	0.666	42.742	40.046	41.651	1.969	2.053	1.964
MPIR	19.653	15.653	16.108	0.672	0.672	0.667	46.93	39.345	40.496	1.977	1.955	1.952
TPIR	22.359	15.462	17.775	0.67	0.67	0.667	53.624	38.715	45.761	1.98	1.958	1.977

Table 4.10: Performance measure $RMSE$ for the different features for the Regression Task in Scenario 2.

	$\Delta t = 1$						$\Delta t = 5$					
Feature	**HepTh**			**ArnetMiner**			**HepTh**			**ArnetMiner**		
	LR	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*
$GERscore^{(1)}_{1,3}$	25.644	26.194	25.006	5.202	5.318	5.168	47.473	48.119	46.625	5.542	5.823	5.41
$GERscore^{(2)}_{1,3}$	24.962	24.631	**23.439**	4.43	**4.669**	4.248	46.371	46.453	**44.973**	4.725	**4.997**	4.605
$GERscore^{(3)}_{1,3}$	26.286	26.806	26.678	5.089	5.281	4.931	48.421	49.107	49.104	5.542	5.799	5.439
$GERscore^{(1)}_{2,3}$	26.113	26.641	25.84	5.431	5.485	5.334	48.012	48.804	47.636	5.998	6.075	5.883
$GERscore^{(2)}_{2,3}$	26.055	26.491	25.751	5.353	5.388	5.067	47.841	48.495	47.328	5.927	6.018	5.602
$GERscore^{(3)}_{2,3}$	26.383	26.854	26.422	5.345	5.444	5.283	48.417	49.152	48.519	5.9	6.048	5.812
GERscore	24.263	24.651	24.346	**4.198**	4.935	**4.201**	45.867	46.296	46.237	**4.287**	5.386	**4.571**
Author Rank	24.032	**23.994**	24.783	5.26	5.403	5.119	45.015	45.383	45.999	5.863	5.924	5.789
MPIA	23.462	24.344	24.085	5.43	5.589	5.42	44.636	45.686	46.001	6.066	6.245	6.133
TPIA	**22.512**	24.086	23.785	5.585	5.561	5.472	**44.45**	**44.874**	46.847	6.261	6.193	6.049
Venue Rank	25.818	26.378	25.892	5.494	5.557	5.466	47.725	48.342	47.454	6.154	6.22	6.131
MPIV	25.923	26.504	25.974	5.522	5.608	5.514	47.873	48.623	47.703	6.2	6.263	6.167
TPIV	25.922	26.495	26.051	5.544	5.613	5.513	47.824	48.642	47.72	6.224	6.267	6.175
Recency	26.441	27.024	26.422	5.574	5.648	5.589	48.428	49.332	48.438	6.266	6.274	6.274
References Rank	25.33	26.16	26.269	5.533	5.619	5.549	46.593	48.006	49.467	6.202	6.259	6.245
MPIR	25.332	26.07	25.467	5.547	5.606	5.553	46.568	47.813	47.469	6.223	6.25	6.228
TPIR	25.055	25.792	27.29	5.528	5.606	5.542	46.162	47.371	47.961	6.203	6.251	6.223

Table 4.11: Performance measures (R^2 and $RMSE$) for the full model with and without the GERscore for the Regression Task in Scenario 1.

		$\Delta t = 1$						$\Delta t = 5$					
	Model	**HepTh**			**ArnetMiner**			**HepTh**			**ArnetMiner**		
		LR	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*
R^2	*All*	0.3	0.19	0.218	0.224	0.197	0.193	0.273	0.147	0.154	0.173	0.162	0.136
	-GERscore	0.288	0.115	0.215	0.022	0.011	0.026	0.269	0.085	0.141	0.063	0.037	0.057
RMSE	*All*	21.984	14.586	21.584	0.626	0.599	0.644	57.694	37.471	58.282	1.872	1.796	1.954
	-GERscore	24.156	16.246	20.899	0.667	0.668	0.673	60.632	41.296	59.817	1.892	1.956	1.941

Table 4.12: Performance measures (R^2 and $RMSE$) for the full model with and without the GERscore for the Regression Task in Scenario 2.

		$\Delta t = 1$						$\Delta t = 5$					
	Model	**HepTh**			**ArnetMiner**			**HepTh**			**ArnetMiner**		
		LR	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*	*LR*	*SVR*	*CART*
R^2	*All*	0.346	0.28	0.366	0.27	0.452	0.474	0.269	0.285	0.226	0.562	0.312	0.527
	-GERscore	0.296	0.221	0.324	0.129	0.213	0.149	0.243	0.272	0.154	0.162	0.141	0.175
RMSE	*All*	22.483	23.438	21.674	4.977	4.166	4.089	44.485	43.258	45.939	4.198	5.478	4.415
	-GERscore	23.112	24.689	22.275	5.325	4.998	5.172	44.686	43.755	47.81	5.767	5.933	5.778

Our explanation is that it happens due to the specifics of the considered datasets and evaluation approaches. The average citation count for ArnetMiner dataset remains around one throughout years, while it gradually grows till twelve for HepTh. Thus, the performance of 5-year prediction for ArnetMiner does not drop so much as for HepTh. The same observation holds for the dataset from the work of Yan et al. [117].

If we study the results in terms of $RMSE$, we observe that 5-year prediction is harder: the values more than double for both datasets in Scenario 1 (see Table 4.11), but the relative drop in Scenario 2 is not so high, especially for ArnetMiner. Again, the reason is the difference between the average citation counts for these datasets: HepTh has a higher average citation count for papers than ArnetMiner. This also explains why the values of $RMSE$ are higher for HepTh.

Another interesting observation is that R^2 is higher in Scenario 2 compared to Scenario 1. We have expected that predicting citation counts for freshly published papers is more difficult since not much is known about them. However, if we compare $RMSE$ across scenarios, we notice that the values are lower in Scenario 1. The explanation is quite simple: average citation counts in Scenario 2 are higher than in Scenario 1, and this leads to a higher error. So, comparing our two scenarios in terms of $RMSE$ does not make sense.

4.6 Summary and Discussion

Overall our new feature GERscore significantly improves citation count prediction. The statistical significance of the improvement has been verified for the full models using ANOVA test. When classifying the future citations, the GERscore is better than the

baseline features in all cases. However, author-related features are still better in the regression task, but only for the dataset HepTh. HepTh provides better coverage of papers in the relevant domain, thus the citations are more complete. Another difference of HepTh from ArnetMiner is the domain: physics for the first and computer science for the latter. The last issue is the amount of mined graph evolution rules: we have only 230 unlabeled evolution rules for HepTh. We are not sure which of these differences leads to the disagreement in the best performing features. In the previous work the authors argue that such disagreement arises due to the nature of the relevant scientific domains [80]. However, additional investigation is required to draw a final conclusion.

We observe that CART performs the best for the regression task in Scenario 2 which agrees with the previous work [117]. However, LR provides better results in Scenario 1. In general, the performance in Scenario 1 is not as good as in Scenario 2. This means that it is much harder to predict citation counts for freshly published papers. It might be the reason why a simple linear regression with a better generalization ability performs well. Surprisingly, CART does not yield the best results for the 5-year prediction which contradicts to the previous work [117]. However, if we leave out the GERscore from the full model ("-GERscore" in Table 4.12), firstly, we have that non-linear models, namely SVR and CART, perform better, secondly, CART yields the best result for ArnetMiner for predicting citation counts over 5 years. For the full model it is the case that the performance drops for HepTh and increases for ArnetMiner over the longer time period. We face again the challenge that more datasets are required to determine whether the nature of the scientific domain influences these results.

Out of all scores which constitute the GERscore, the best results are gained for the scores calculated from the unlabeled graph evolution rules (see Tables 4.8 and 4.10). When aggregating separate scores, summation is a better choice compared to maximum. This is an unfortunate outcome since aggregation with maximum would allow us to speed up the graph pattern mining by setting a high support threshold. The decrease in running time is also gained through mining labeled graph evolution rules. Though $GERscore_{1,i}^{(2)}$ provides better results compared to other label settings and aggregation technique, we still receive that the other scores contribute to the combined GERscore.

Our results are coherent with Yan et al. for ArnetMiner in Scenario 2 which is the only setting that corresponds to theirs: Author Rank is better than Venue Rank [116, 117]. However, we show that the GERscore is even better in this case. Moreover, we arrive already at a better performance just by identifying graph evolution rules in the unlabeled citation network from the previous years.

We have constructed a new feature - GERscore - for estimation of future citation counts for academic publications. Our experiments show that the new feature performs better than ten state-of-the-art features in the classification task. Furthermore, the average accuracy of the classification is not affected much if we bring in other baseline features into the model. In the regression task the new feature outperforms the state-of-the-art features for the dataset of publications from computer science domain (ArnetMiner), though the latter still contribute to the performance of regression models. Thus, the application of graph pattern mining to the citation count prediction problem leads to better results.

However, for the dataset of publications from physics (HepTh) the GERscore is not as good as the author related features, i.e., author rank, MPIA and TPIA, though it does contribute to the increase of the performance. Additional investigation is required to identify the reason for the disagreement in the best performing features.

We have performed both classification and regression tasks for the prediction of citation counts in one year. Additionally, we predict the actual citation counts in 5 years. We observe that it becomes harder to predict citation counts over longer periods. Our results also indicate that the performance of the model does not always improve if we include more features. We have not included all features from the previous works in our evaluation framework, e.g., content related features [80, 116, 117] or network related features [80, 101]. Thus, an important aspect to investigate is the performance of various features on different datasets and their optimal combination where dimension reduction methods might be of help. Ultimately, we want to include our findings into a recommender system for academic publications.

Our future work includes thorough investigation how the mined evolution rules influence the predictive power of the GERscore. Here we want to investigate in several directions. The first issue is to study the influence of input parameters, minimum support (minSup) and maximum size (maxSize), and what is the best combination for them. We need to take into consideration that by setting maxSize high and minSupport low we will obtain more evolution rules, however the computational time will grow exponentially. Another issue is that real-world networks change considerably over time. It may lead to the fact that the evolution rules which are frequent and have high confidence at time t may become rudimentary in ten years and will not be predictive of the citation counts. Thus, we plan to investigate for how long mined evolution rules on average stay predictive. This is an important question also because mining graph evolution rules is computationally hard, and reducing the amount of re-learning GERscores is extremely important.

CHAPTER 5

Heterogeneous Networks

> In the digital universe, our personal history and its sense of narrative is succeeded by our social networking profile - a snapshot of the current moment. The information itself - our social graph of friends and likes - is a product being sold to market researchers in order to better predict and guide our futures.
>
> *Douglas Rushkoff*

The work from this chapter was performed under the supervision of Hannes Werthner and Ryutaro Ichise, and appeared in the companion proceedings of the International World Wide Web conference 2015 [98].

5.1 Motivation

An important problem in social networks, called link prediction, is to predict the appearance of a link between two disconnected people, or nodes, in the network. Nowadays link prediction has applications in many different domains, such as in communication or collaboration networks.

Earlier works on link prediction have focused on simple network models disregarding the temporal aspects and heterogeneity of human relationships [2, 88]. Temporal aspects have often been omitted due to insufficient data [31]. But nowadays, with the availability of Web APIs on many platforms, it is possible to collect data about social interactions over considerable time periods on a large scale. Recent works show that models which consider temporality better capture link formation processes in the network [30, 86, 118]. As for

the second property, heterogeneity can be frequently observed in real-world networks. For example, on Facebook we can study the formation of friendship ties and communication interactions between users. Classical methods for link prediction in homogenous networks cannot fully capture the complex structure in such scenarios. Hereby, the two easiest ways to adapt these methods to a heterogeneous setting are either to treat all link types equally or to consider only one link type and disregard the others. Either way we might lose valuable information which becomes even more critical in the light of sparse nature of many social networks.

We formulate link prediction as a binary classification problem, where a pair of nodes is classified as a positive case if they form a link. To efficiently combine information about various types of links in the network, we follow the approach of counting 3-node graphlets [31]. In this approach, we identify for a pair of nodes the type of a pattern/triad they form with their common neighbors and put a score proportional to the frequency of this triad in the whole network. We introduce three modifications to the original weighting scheme of triads.

The first modification is the introduction of time awareness which we achieve by using the time score [86]. The second modification is motivated by the work in the area of graph pattern mining where the stress is put on the fact that simple counts are not always a proper measure to estimate the frequency of patterns in the network [22]. Our last modification addresses the hypothesis that triad formation among more active (thus experienced) and ordinary users in the network might be different. Therefore, we differentiate the 3-node graphlets not only by link type, but also by node categories. We also raise the question whether there are certain network properties which might point out a suitable weighting scheme of triads for temporal link prediction.

The main contributions in this chapter are the following:

1. We study the performance of several supervised methods for temporal link prediction in heterogeneous social networks at several time points.

2. We suggest three modifications to the weighting scheme of 3-node graphlets [31].

3. Our experiments illustrate that network evolution cannot be explained by one specific feature at all time points which emphasizes the importance of combining different features into efficient models.

4. We observe that some network properties can point out which weighting scheme for 3-node graphlets is more effective for temporal link prediction.

The structure of this chapter is as follows. In the next section we provide a formal definition of the problem. In Section 5.3 we discuss related work. We explain the calculation of our new features in Section 5.4. In Section 5.5 we present datasets and experimental setting. We also report and discuss the results of our experiments. Finally, the conclusion is drawn and possible directions for future work are outlined.

5.2 Problem

We have a network $G = (V, E_1, E_2, \tau_1, \tau_2)$, where V is the set of nodes; E_1 and E_2 are the set of links of type 1 and 2 respectively; τ_1 assigns timestamps to links of the first type and τ_2 to links of the second type. The set of nodes does not need necessarily to be homogeneous, but we focus on the case when nodes represent the same type of objects, namely people. Let $G[t] = (V, E_1[t], E_2[t], \tau_1, \tau_2)$ denote the sub-network of G which exists at time $t \in T$. In other words, this sub-network contains only those links which have a timestamp not greater than t: $E_1[t] = \{e \in E_1 : \tau_1(e) \leq t\}$, $E_2[t] = \{e \in E_2 : \tau_2(e) \leq t\}$.

The link prediction problem consists either of finding hidden connections or predicting links which will appear in the future based on the previously observed network states [50]. The first problem is better studied in the link prediction community. The latter, predicting future links, has gained more interest recently. We focus on this problem in the following formulation: Given $G[t] = (V, E_1[t], E_2[t], \tau_1, \tau_2)$ predict $E_x[t']$ for some $t' > t$ and link type $x \in \{1, 2\}$.

5.3 Related Work

There are many methods developed for the link prediction problem in networks with one type of links. We enumerate some of them in Chapter 3. We focus on the category of link prediction methods which are based on topological patterns, since this category is related to our work. The aim here is to calculate a similarity score between a pair of nodes based on the topological metrics of the network. In the following we repeat the definitions of the methods which are of especial interest in our work.

Among the topological metrics, the neighborhood based scores form the biggest category. The *preferential attachment* (PA) score for a node pair (s, t) is the product of their degrees: $|N_s| * |N_t|$ [88]. Here, N_x denotes the set of immediate neighbors of the node x. The *common neighbors* (CN) counts the number of common neighbors: $|N_s \cap N_t|$ [88]. *Jaccard's coefficient* (JC) [99] is a normalized number of common neighbors:

$$\frac{|N_s \cap N_t|}{|N_s \cup N_t|}.$$

The *Adamic/Adar* (AA) measure [2] weights the impact of neighbors inversely according to their degrees:

$$\sum_{n \in N_s \cap N_t} \frac{1}{\log |N_n|}.$$

Unsupervised and supervised methods can be used to build a prediction model based on these scores. A survey of unsupervised methods is provided in [74]. Lichtenwalter et al. show that supervised learning provides better results due to the ability to reduce variance and to cope with high imbalance in the class distribution [77]. To overcome imbalance, they suggest to use skew-insensitive trees based on Hellinger distance or to undersample the negative class. We choose to undersample the negative class when constructing the

training and testing datasets since such approach decreases the time which is required for learning.

Supervised methods are appropriate if we observe a sufficient amount of links in the network which is not always true. That is why a set of methods have been developed to predict links across networks. There are a source network with observed links and a target network with missing links. These two networks can be completely different: with different nodes and different types of interactions (links). The idea is to learn a model on the source network and to apply it to the target network. Variations of a transfer learning framework which incorporate various social theories have been suggested to solve this problem [105, 37]. These works focus on generalizing the link prediction methods.

We want to efficiently combine information about different types of links in the same network to predict future links. For this purpose Davis et al. construct a weighting scheme for combinations of link types based on the frequencies of 3-node graphlets [31]. They define connected 3-node graphlets, following social theories similar to [37], but unlike the latter work where the patterns differ in node attributes (opinion leader or not), this work constructs patterns with possible combinations of link types. In the following work Lichtenwalter and Chawla extend the class of graphlets and introduce a new method for link prediction, called Vertex Collocation Profiles (VCP) [76]. Here the authors consider graphlets with more than 2 nodes. In particular they conduct experiments on 11 networks using graphlets with 3 and 4 nodes. Theirs results indicate that VCP outperforms the classical features in many cases. However, time information is not considered.

There are already works which incorporate time information into the calculation of similarity scores. For example, Munasinghe and Ichise suggest a time-score which combines the time and weight of links with common neighbors [86]. Their approach is based on two concepts: the strength of a link decreases with time, and the common neighbors are more effective if the interaction with them happens in a closer proximity of time. They show that the constructed score achieves better f-measures for temporal link prediction in the supervised setting.

Soares and Prudencio do not construct one single score for a node of pairs, but rather explore the evolution of topological metrics by constructing a time series of scores [30]. Based on the obtained time series for a specific neighborhood based score, they predict the next value of the series and use the predicted value in the link prediction model. They use both supervised and unsupervised neighborhood based methods (PA, CN, AA, JC) on two co-authorship networks and show that by applying one of time series techniques better AUROC results can be achieved. However, many works on temporal link prediction consider only homogeneous networks [30, 86].

We want to perform temporal link prediction in networks with two types of links. There is a recent work which solves this problem on a DBLP dataset by performing a multi-relational influence propagation [118]. To account for temporal dimension, the authors introduce several features. Firstly, they calculate recency as the time passed since a node made its last new link and activeness as the number of new links made in the last time step. These two features are somewhat similar to the time score [86]: instead of introducing time awareness for the links this work makes nodes time aware.

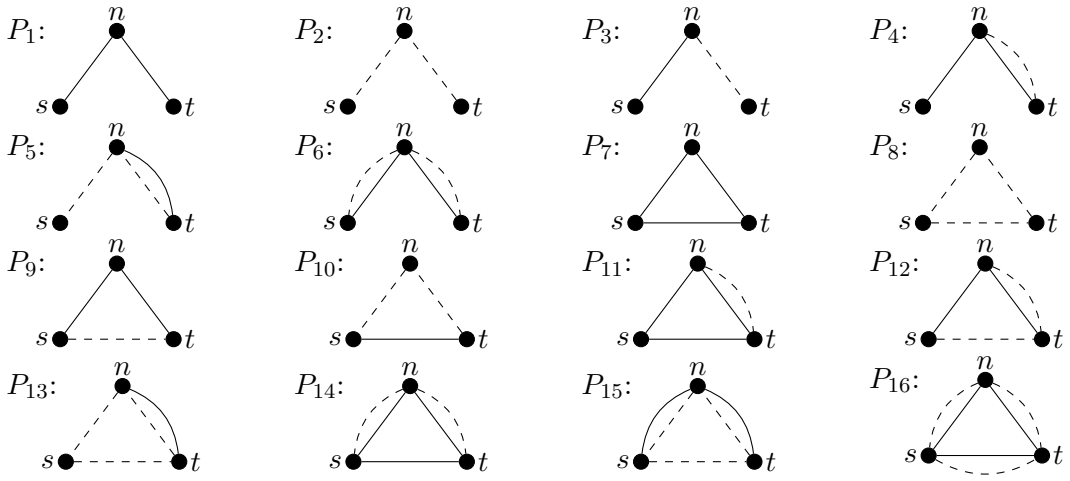


Figure 5.1: 3-node graphlets for a network with links of two types.

Secondly, the authors calculate a degree preferential vector which can be perceived as a time series approach [30]. However, they use a different probabilistic technique to calculate the final feature. Finally, they design two new models (Temporal and MRIP models) and compare their performance against three state-of-the-art models (Homo Model, VCP3U and VCP4U). They achieve the best AUROC for the Temporal model in the supervised setting.

Our approach is based on the works of Davis et al. and Munasinghe et al. [31, 86]. Therefore, we use these works as baselines to compare our approach. We do not study the combinations of features. The future work includes this aspect and a comparison with the other models [118, 76].

5.4 Time and Heterogeneity based scores

Davis et al. introduce a prediction score for an appearance of a link between two nodes based on the frequency of 3-node graphlets which these two nodes form with their common neighbors [31]. We suggest to extend this methodology in three ways:

(1) by introducing an additional weight based on time of interactions in the network;

(2) by using support as a measure to account for frequency of graphlets;

(3) by using node labels which indicate how experienced nodes are.

The original score which is introduced to predict a link of type x between nodes s and t is [31]: $score^{(i)}(s,t) = \sum_{n \in N_s \cap N_t} w_n^{(i)}$, where $w_n^{(1)}$ corresponds to the weighting scheme from the previous work and $w_n^{(2)}$ is our modification. To calculate the first score, we put:

$$w_n^{(1)} = \frac{\sigma \cdot |P(x) - P(x \subset \text{edge\_type}(s,t) | \text{pattern}(s,n,t))|}{P(\text{edge\_type}(s,n)) \cdot P(\text{edge\_type}(t,n))}$$

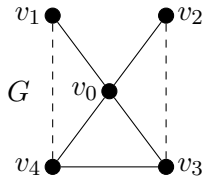


Figure 5.2: Example of a network G with two types of links.

in which pattern(s, n, t) describes one of possible 3-node graphlets formed by nodes s, n, t. The possible graphlets for a network with two types of links are illustrated in Figure 5.1. We do not regard the symmetrical cases separately. For example, the pattern P_4 has an isomorphic pattern where the link of the second type is between the nodes n and s. In our implementation both cases are captured with the pattern P_4.

To calculate the conditional probability of the link x to appear in pattern(s, n, t), in other words $P(x \subset \text{edge\_type}(s, t)|\text{pattern}(s, n, t))$, we count the occurrence of pattern(s, n, t) and the occurrence of this pattern with the added link type x. Then we divide the latter number by the former. In both cases we talk about graph patterns specified in Figure 5.1. $P(x)$ stands for the probability of the link type x in the network , and edge_type(s, t) determines the link type between nodes s, t. N_s, N_t stand for the sets of neighbors for nodes s, t correspondingly. The value of $\sigma \in \{-1, 0, 1\}$ is determined upon statistical comparison of $P(x \subset \text{edge\_type}(s, t)|\text{pattern}(s, n, t))$ and $P(x)$ for a given link type x:

$$\sigma = \begin{cases} 1 & \text{if } P(x \subset \text{edge\_type}(s, t)|\text{pattern}(s, n, t)) > P(x) \\ 0 & \text{if } P(x \subset \text{edge\_type}(s, t)|\text{pattern}(s, n, t)) = P(x) \\ -1 & \text{if } P(x \subset \text{edge\_type}(s, t)|\text{pattern}(s, n, t)) < P(x). \end{cases}$$

We conduct statistical comparison the following way. We perform a two-tailed two proportion z test for sample proportions $p_1 = P(x \subset \text{edge\_type}(s, t)|\text{pattern}(s, n, t))$ and $p_2 = P(x)$. We put sample sizes $n_1 = count(\text{pattern}(s, n, t) + x)$ and $n_2 = count(x)$. Then, pooled sample proportion $p = (p_1 * n_1 + p_2 * n_2)/(n_1 + n_2)$, and standard error $SE = \sqrt{p * (1 - p) * [\frac{1}{n_1} + \frac{1}{n_2}]}$. Now z-score is calculated as $z = (p_1 - p_2)/SE$. We calculate p-value and critical values using z-score and significance level 0.01.

The approach of 3-node graphlets is loosely related to the approach of mining graph evolution rules [21] which we used in Chapter 4 to solve the citation count prediction problem. Unlike counting apriori known graphlets, Bringmann et al. discover frequent graph patterns in a network which are limited in their size. The frequency of the graph patterns is measured by the minimum image-based support. It roughly equals to the minimum number of different nodes in the network to which one of the nodes in the pattern can be matched. Bringmann and Siegfried argue that support is a better measure to estimate the frequency of a pattern than a simple count [22]. Consider the network G in Figure 5.2, where the first type of link is represented by the full line, and the second type is shown by the dashed line. 3-node graphlet P_1 has a count 16 in this network,

and the corresponding closed triad P_4 has a count 6. The conditional probability of the appearance of the link of the first type between nodes v_1 and v_2 using counts [31]: $P(x \subset \text{edge\_type}(v_1, v_2)|\text{pattern}(v_1, v_0, v_2)) = \frac{6}{16}$. The minimum image-based support for P_1 is three, since the node n can be matched only to $v_0, v_3, v4$ in the network G. P_4 has the same support. Hence, the conditional probability of the appearance of the link of the first type between nodes v_1 and v_2 using support is 1. Note that such difference in these two calculation approaches arises if we have a high-degree node which connects otherwise disconnected nodes. Such kind of nodes are also referred to as structural holes. We do not know how exactly the temporal evolution is affected by structural holes in the network, but a previous work points out that structural holes can indicate the relationship type [105]. To account for such behavior in social networks, we calculate $w_n^{(2)}$ using the support [21]:

$$w_n^{(2)} = \frac{support^2(x \subset \text{edge\_type}(s,t), \text{pattern}(s,n,t))}{support(\text{pattern}(s,n,t))}.$$

This score has proven to perform well not only for link prediction in social networks [21], but also for prediction of citation counts over time [92].

To consider the temporal aspect of node interactions, we use the time score [86]: $TS = \sum_{n \in N_s \cap N_t} \frac{H_m * \beta^k}{|t_1 - t_2| + 1}$, where t_1, t_2 are recent interactions between n, s and n, t correspondingly, $k = now - \max(t_1, t_2)$, H_m is a harmonic mean of weights of these two interactions. We put β equal to 0.5. This score is based on two concepts: the strength of a link decreases with time, and the common neighbors are more effective if the interaction with them happens in a closer proximity of time. The time score can be naturally applied to each 3-node graphlet separately. Then we put a new time-dependent score:

$$tscore^{(i)}(s,t) = \sum_{n \in N_s \cap N_t} TS_n \cdot w_n^{(i)}.$$

The last extension is motivated by the work on link prediction across heterogeneous networks [37]. A more fine-grained mechanism to study network growth is proposed. First, nodes are categorized in two groups (elite and ordinary users) based on their PageRank score. Then triads (3-node graphlets) with respect to node categories are enumerated and weighted with regard to their frequencies. We also introduce two node categories, ordinary and experienced users, which we estimate based on user's engagement in the network. We obtain a new score:

$$tscore_l^{(i)}(s,t) = \sum_{n \in N_s \cap N_t} TS_n \cdot W_n^{(i)}.$$

$W_n^{(i)}$ is calculated like $w_n^{(i)}$, except that patterns and link types include labels of the nodes.

From the perspective of social theory, we incorporate three social patterns in our methodology:

Table 5.1: Properties of Dota2 network over time.

	team mate link					*friend link*				
year-week	# nodes	# links	$\bar{C}$	T	# CC	# nodes	# links	$\bar{C}$	T	# CC
201146	4293	11710	0.22	0.112	214	66786	76613	0.053	0.018	1854
201147	7901	17544	0.216	0.112	584	69505	81733	0.058	0.021	1831
201148	12323	28669	0.217	0.099	618	72198	86666	0.064	0.023	1783
201149	18763	47145	0.214	0.084	668	75756	92656	0.069	0.026	1786
201150	24347	64333	0.209	0.074	660	79432	99164	0.076	0.028	1731
201151	28639	81237	0.208	0.069	526	81858	103983	0.079	0.03	1602
201152	31768	96876	0.208	0.064	416	83975	108091	0.082	0.032	1501
201201	34839	114426	0.205	0.061	271	86272	112574	0.083	0.033	1432
201202	37818	144961	0.199	0.061	142	88245	117137	0.086	0.034	1309
201203	40104	176195	0.196	0.063	88	90010	121424	0.087	0.035	1214
201204	41946	205395	0.194	0.065	62	91565	125374	0.089	0.037	1125
201205	43701	232289	0.192	0.07	48	93295	129767	0.092	0.038	1043
201206	45238	256900	0.192	0.073	40	94961	134082	0.094	0.039	963
201207	46645	276768	0.19	0.076	34	96534	138030	0.095	0.039	903
201208	47772	291577	0.19	0.077	27	97984	141522	0.096	0.04	862

1. social balance which is based upon a principle that "the friend of my friend is my friend";
2. microscopic mechanism which investigates human social interactions and agency on a small scale;
3. time awareness which assumes that the strength of a relationship weakens with time.

The first social pattern is supported by any link prediction method which is based on the triad-closing model. For example, AA or CN. The new time and heterogeneity based scores support the second social pattern since we study triad closing with regard to the node labels and link types. The last social pattern is ensured due to the usage of the time score.

The new scores are similar to AA, since these features are based on the triad formation. However, they all provide different weighting scheme for a given triad. This scheme is constructed with the goal to capture the network formation over time in the best possible way. We hypothesize that our new weighting scheme will better capture the network evolution over time.

5.5 Experimental Results

5.5.1 Datasets

We conduct experiments on two datasets: the *gaming network Dota2* and the *collaboration network HepTh*. In the *gaming network Dota2* nodes correspond to users of Steam; links correspond to - (1) friendship relationship if two users are friends on Steam; this relationship is timed and undirected; (2) team relationship if two users played in the same team in the same match of the game Dota2; this relationship is weighted, timed and undirected. In the *collaboration network HepTh* nodes correspond to authors of scientific

Table 5.2: Properties of HepTh network over time.

	peer link					*colleague link*				
year	# nodes	# links	$\bar{C}$	T	# CC	# nodes	# links	$\bar{C}$	T	# CC
1993	529	1618	0.435	0.314	30	965	1185	0.521	0.592	217
1994	1494	9008	0.441	0.265	27	1776	2497	0.492	0.473	263
1995	2500	21845	0.461	0.245	23	2579	3988	0.482	0.409	289
1996	3280	40913	0.47	0.243	16	3248	5482	0.475	0.357	313
1997	4063	68510	0.471	0.248	14	3902	7173	0.472	0.313	318
1998	4690	102892	0.469	0.254	15	4463	8827	0.462	0.285	319
1999	5325	140528	0.477	0.259	16	5025	10622	0.463	0.263	327
2000	5941	182550	0.48	0.255	16	5583	12425	0.468	0.246	332
2001	6677	241091	0.485	0.252	16	6240	14534	0.466	0.229	349
2002	7318	301300	0.49	0.247	13	6814	16594	0.467	0.217	342
2003	7978	368323	0.493	0.25	15	7416	18793	0.467	0.206	349

publications from HepTh; links correspond to (1) colleague relationship if two authors collaborated on a publication; this relationship is timed, weighted and undirected; (2) peer relationship if one of the authors cited the other; this relationship is weighted, timed and undirected.

Gaming Network. The dataset was crawled using Steam API (friendship information) and Dota2 API (team membership information) from Valve. The team membership information is obtained from the matches of Dota2 game which occurred during 2011 and first two months of 2012. The data crawler was implemented in Python. It stores data in a PostgreSQL database. We show the entity relationship diagram of this database in Figure 5.3. The database contains seven tables. The data for two of these tables is obtained by using the Steam Web API. The rest is populated via the Dota2 Web API. For the work in this thesis we do not use information from all the tables. In particular, from the data, which is crawled via the Dota2 Web API, we use only tables "match_players" and "matches".

A detailed description of the dataset is available in the works where we studied the factors for a team of five players to win in a match of the game Dota2 [95, 94]. Based on the crawled data, we construct a network the following way. Nodes correspond to users of Steam in both cases. The *mate* link indicates that two players were team mates in a Dota2 match, and the *friend* link means that two players are friends on Steam. The mate link has a weight as the number of matches where two players were team mates. We consider only links with weight>1. We make this restriction for two reasons: firstly, if the weight is more than one, than the interaction is less likely to be random; secondly, to decrease the network volume. There are 309,612 "mate" links. 46,574 out of these "mate" links have corresponding "friend" links. We introduce a weight on the friend links as the current friendship age. We select one calendar week as the time unit. The users in this network are classified at each time point in two groups based on their experience: if they played more than 60 matches, they are marked as "experienced", otherwise "ordinary". The threshold is identified by the distribution of users' experience in the week 201145. 1% Top users are selected as experienced.

Collaboration network. The collaboration network is constructed from 27,732 papers from the years $1992-2003$ from the arXiv which are categorized as High Energy

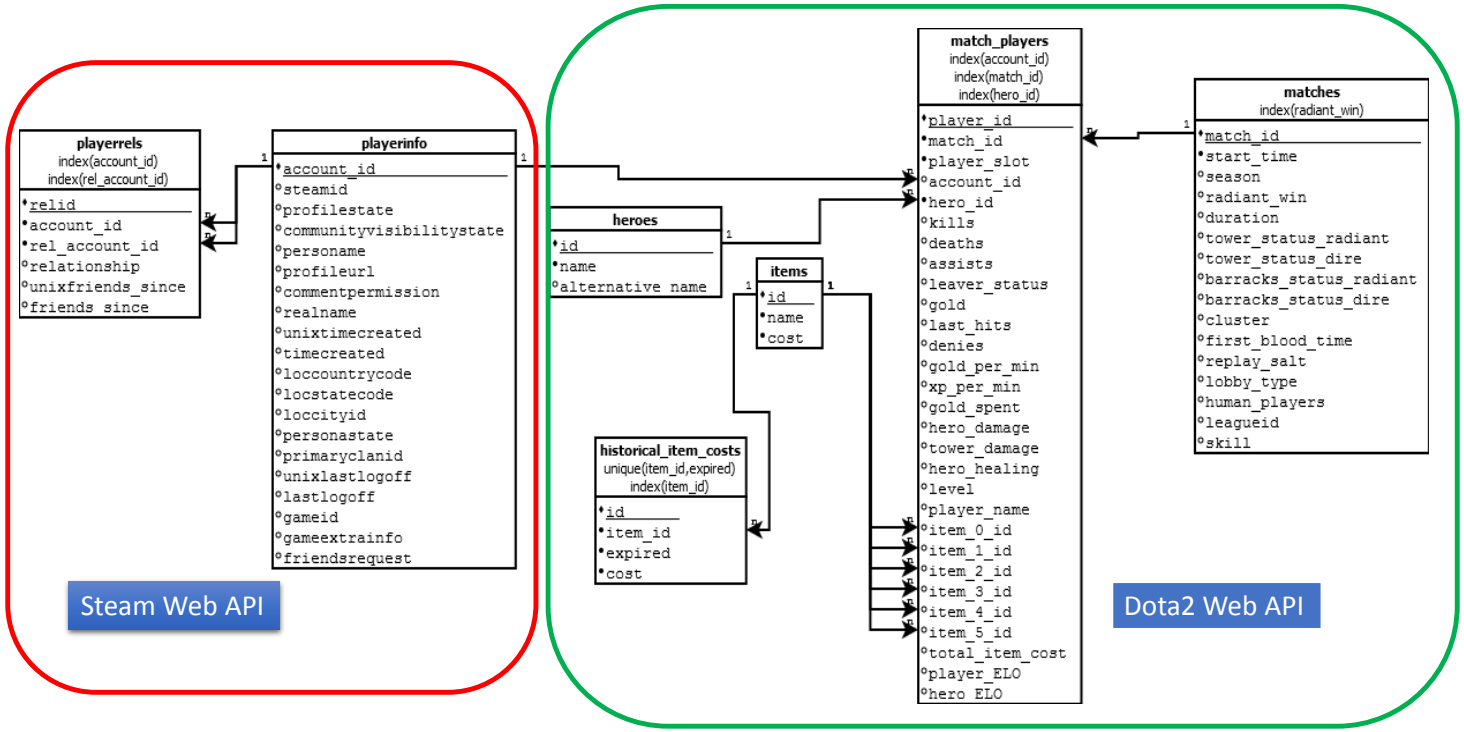


Figure 5.3: The entity relationship diagram of PostgreSQL database which stores data for the gaming network Dota2.

Physics Theory. We obtain 8,204 nodes which correspond to distinct authors, 19,454 links of type "colleague" which correspond to being co-authors on one of the papers, and 388,677 links of type "peer" which indicates if one of the authors cites the other one. Among these links, there are 16,464 links where both types are present. Nodes correspond to authors of these publications. There are *colleague* and *peer* types of links. If a scientist collaborates with another one on some paper, it indicates that they are colleagues and know each other on a personal level. If a scientist cites one of the works of the other author, it indicates that both of them are working on related topics, and, thus, they may be considered as peers. That is also the reason why we ignore the direction of this relationship. We delete self-loops. There are weights on both types of links which indicate how many times the corresponding interaction between the authors takes place. We select one year as the time unit. The authors are classified at each time point in two groups based on how many papers they wrote before: if they authored more than 10 papers, they are marked as "experienced". The threshold is identified by the distribution of users' experience in the year 1993. Again, 1% top users are selected as experienced.

We present the properties for each link type in Tables 5.1 and 5.2 over the selected time periods. These properties include number of nodes (#nodes), number of links (#links), average clustering coefficient ($\bar{C}$), transitivity (T) and number of connected components (#CC). The transitivity T of a graph is based on the relative number of triangles in the graph, compared to total number of connected triples of nodes. The clustering coefficient of an undirected graph is a measure of the number of triangles in a graph. The clustering coefficient of a graph is based on a local clustering coefficient for each node. Both transitivity and clustering coefficient measure the relative frequency of triangles. However, average clustering coefficient places more weight on the low degree nodes, while the transitivity ratio places more weight on the high degree nodes.

5.5.2 Experimental setting

We select several state-of-the-art scores as our baseline: AA, CN, JC and PA which are described in Section 5.3. We also consider TS [86] and $score^{(1)}$ [31]. We predict each link type separately, thus features AA, CN, JC, PA and TS are calculated for the network with one link type. We construct the training and testing datasets based on link existence at different time units: t and $t+1$ correspondingly. If a link appears between two nodes at time t, then it is a positive class in the training dataset. The same rule is used for the testing dataset, except that we consider time $t+1$. The features are calculated based on $G[t-1]$ for the training dataset and $G[t]$ for the testing dataset. We predict links for the testing dataset at time $t+1$. We undersample the negative class to balance the class distributions [77, 31, 30, 76]. We choose to undersample the negative class so that the positive class represents 25% observations [31, 76]. Another crucial factor is the neighborhood in which we undersample the majority class. We consider only those pairs of nodes which have common neighbors in the network.

We incorporate bagging into our model [31]. We build ten folds for the training datasets: each fold contains all observations of the positive class, but we take different observations of the negative class to reach the balanced ratio of classes; we use the

same testing dataset for each fold of the training dataset. We experimented with three classification methods: Logistic Regression (LR), conditional inference trees (CIT) and Random Forests (RF). Due to consistently better performance of CIT, we report the results of this method.

Recent works indicate that in the case of a highly imbalanced class distribution Area under Precision-Recall curve (AUPR) is a better performance measure [31, 32, 76, 119]. If we put tp true positives, tn true negatives, fp false positives, and fn false negatives, then: precision is calculated as $tp/(tp + fn)$ and recall equals $tp/(tp + fn)$. Precision-Recall curve plots recall versus precision at different cutoff levels. Since many works report only AUROC or top N results, we also report Area under ROC curve (AUROC). We use R packages in our learning tasks: party and PerfMeas [97].

5.5.3 Results

We report AUPR results in Tables 5.3-5.4 for Dota2 network and in Tables 5.5-5.6 for HepTh network. The AUROC results are indicated in tables 5.7-5.10. The best performing feature for each time period in the network (i.e., each row in the tables) is marked in bold. We noticed that, in case AUPR is more than 0.99, the learning algorithm fails to create a classification model (AUROC equals 0.5 in such cases). Closer analysis reveals that in such cases the distributions of the calculated scores are almost identical for positive and negative classes, resulting in all instances being classified into the negative class. Thus, we mark the corresponding numbers in the tables in italic and identify the second best results. Our general expectation is that the new time-dependent scores $tscore^{(i)}$ and $tscore_l^{(i)}$ will perform better than $score^{(i)}$. We do not expect $score^{(2)}$ to outperform $score^{(1)}$, but we want to define the cases when it does. Lastly, we expect that including additional information about users' experience will improve the performance, i.e., $tscore_l^{(i)}$ is better than $tscore^{(i)}$.

The results indicate that there is no definite winner in all cases. The performance of features varies over time and across link types. However, if we compare the average of AUPR over the considered time periods, we obtain that $tscore_l^{(1)}$ provides the highest value for 3 out of 4 link types (we exclude cases with AUPR>0.99 from averaging). Prediction of friend links in Dota2 network is the only case when JC yields better average results. It is also the case when the new scores based on the support calculation outperform almost at all time points the scores which use counts for graphlets. In Table 5.1 we notice that the network based on friend links has very low average clustering coefficient and transitivity, especially compared to three other networks. These two network measures provide an insight how well the social balance theory is fulfilled within the network. Apparently, such configuration leads to the better performance of JC and PA which are not based on this theory. Furthermore, it also provides an explanation why using support for frequency calculation is a better choice in this situation. Remember the example network G in Figure 5.1 where the conditional probability of link appearance is higher if we use the support measure. Similar situation will arise every time we have structural holes in the network. Tang et al. include the information about structural

Table 5.3: AUPR for predicting mate links.

week	AA	CN	JC	PA	TS	$score^{(1)}$	$score^{(2)}$	$tscore^{(1)}$	$tscore^{(2)}$	$tscore_l^{(1)}$	$tscore_l^{(2)}$
201146	0.344	0.426	0.362	0.431	0.288	0.187	**0.675**	0.282	0.66	0.244	0.606
201147	0.382	0.409	0.138	0.309	**0.505**	0.152	0.223	0.191	0.335	0.143	0.31
201148	0.192	0.21	*0.999*	**0.489**	0.201	0.141	0.126	0.113	0.169	0.117	0.191
201149	0.123	0.184	0.219	0.181	0.142	0.134	**0.431**	0.113	0.288	0.119	0.121
201150	0.082	0.092	**0.168**	0.08	0.146	0.095	0.129	0.087	0.092	0.094	0.143
201151	0.093	0.114	0.096	0.12	0.142	0.087	0.105	0.101	**0.204**	0.094	0.185
201152	0.083	0.129	**0.828**	0.121	0.128	0.103	0.085	*0.999*	0.108	*0.999*	0.087
201201	0.102	0.112	0.177	0.096	0.122	0.08	0.12	0.215	0.17	**0.216**	0.215
201202	0.074	0.079	0.109	0.076	0.19	0.11	0.081	0.122	0.082	**0.19**	0.095
201203	0.095	0.095	0.176	0.084	0.107	0.119	0.09	0.163	0.083	**0.368**	0.094
201204	0.078	0.09	0.069	0.102	0.088	0.112	0.08	0.237	0.108	**0.336**	0.112
201205	0.078	0.084	0.084	0.082	0.076	0.093	0.08	0.121	0.082	**0.361**	0.08
201206	0.067	0.07	0.117	0.084	0.123	0.068	0.072	0.18	0.07	**0.338**	0.066
201207	0.072	0.069	0.134	0.071	0.068	0.073	0.071	**0.211**	0.072	0.172	0.073
Average	0.097	0.115	0.204	0.138	0.133	0.104	0.127	0.145	0.132	**0.223**	0.126

Table 5.4: AUPR for predicting friend links.

week	AA	CN	JC	PA	TS	$score^{(1)}$	$score^{(2)}$	$tscore^{(1)}$	$tscore^{(2)}$	$tscore_l^{(1)}$	$tscore_l^{(2)}$
201146	0.282	0.528	**0.733**	0.674	0.213	0.302	0.276	0.201	0.273	0.2	0.309
201147	0.423	0.498	0.547	**0.675**	0.168	0.315	0.288	0.344	0.421	0.266	0.597
201148	0.276	0.496	**0.797**	0.503	0.27	0.26	0.346	0.465	0.398	0.389	0.385
201149	0.249	0.343	**0.941**	0.594	0.195	0.35	0.154	0.379	0.711	0.329	0.259
201150	0.203	0.182	0.694	0.604	0.14	0.244	0.339	0.307	**0.799**	0.325	0.5
201151	0.546	0.524	0.323	0.649	0.212	0.276	0.115	0.363	**0.969**	*0.999*	0.817
201152	0.385	0.383	0.831	0.767	0.626	0.274	0.74	0.359	**0.956**	0.174	0.949
201201	0.195	0.284	**0.953**	0.915	0.458	0.195	0.07	*0.999*	0.939	0.4	0.943
201202	0.355	0.204	0.923	0.616	0.924	*0.999*	0.622	0.199	**0.942**	0.19	0.769
201203	0.179	0.203	0.601	0.799	0.258	0.214	0.237	0.173	**0.962**	0.273	0.215
201204	0.152	0.226	**0.956**	0.444	0.318	0.152	0.615	0.178	0.379	0.264	0.504
201205	0.23	0.249	0.57	0.361	0.385	0.195	**0.653**	0.228	0.251	0.192	0.343
201206	0.194	0.202	0.681	0.667	0.181	0.183	**0.697**	0.227	0.262	0.282	0.17
201207	0.213	0.21	**0.968**	0.517	0.894	0.192	0.593	0.278	0.244	0.315	0.218
Average	0.27	0.3	**0.752**	0.629	0.361	0.234	0.417	0.288	0.688	0.282	0.532

Table 5.5: AUPR for predicting peer links.

year	AA	CN	JC	PA	TS	$score^{(1)}$	$score^{(2)}$	$tscore^{(1)}$	$tscore^{(2)}$	$tscore_l^{(1)}$	$tscore_l^{(2)}$
1994	0.202	0.225	0.095	0.168	0.167	0.13	0.325	0.265	0.195	**0.336**	0.131
1995	0.223	0.23	0.166	0.18	0.196	0.163	0.327	0.183	0.27	**0.423**	0.109
1996	0.208	0.21	0.116	0.154	0.212	0.169	0.239	0.241	0.242	**0.418**	0.219
1997	0.203	0.19	0.221	0.141	0.181	0.171	0.193	0.242	0.193	**0.715**	0.235
1998	0.182	0.183	0.168	0.143	0.196	0.182	0.185	0.202	0.184	**0.328**	0.19
1999	0.198	0.195	0.179	0.155	0.194	0.184	0.165	0.214	0.197	**0.342**	0.203
2000	0.192	0.194	0.217	0.154	0.184	0.186	0.172	**0.257**	0.199	0.249	0.205
2001	0.243	0.242	0.227	0.18	0.234	0.245	0.22	0.332	0.236	**0.718**	0.244
2002	0.203	0.201	0.212	0.159	0.189	0.189	0.196	0.186	0.19	**0.666**	0.194
Average	0.206	0.209	0.174	0.159	0.195	0.179	0.228	0.242	0.215	**0.441**	0.192

Table 5.6: AUPR for predicting colleague links.

year	AA	CN	JC	PA	TS	$score^{(1)}$	$score^{(2)}$	$tscore^{(1)}$	$tscore^{(2)}$	$tscore_l^{(1)}$	$tscore_l^{(2)}$
1994	0.161	0.356	0.146	*0.991*	0.287	**0.813**	0.579	0.566	0.546	0.359	0.772
1995	0.154	0.195	*0.992*	0.224	0.254	0.805	0.641	**0.86**	0.573	*0.993*	0.835
1996	0.296	0.37	0.261	0.226	0.541	0.562	0.289	0.591	0.294	*0.995*	**0.648**
1997	0.276	0.214	0.148	0.164	0.216	**0.468**	0.311	0.215	0.298	*0.996*	0.358
1998	0.182	0.136	0.138	0.186	0.109	0.4	0.214	0.138	0.321	**0.769**	0.31
1999	0.385	0.199	0.123	0.138	0.145	0.271	0.123	0.145	0.189	**0.585**	0.204
2000	0.376	0.186	**0.692**	*0.997*	0.136	0.11	0.135	0.1	0.158	*0.997*	0.127
2001	0.116	0.223	0.166	*0.997*	0.122	0.306	**0.792**	0.17	0.085	*0.997*	0.086
2002	0.111	0.243	0.195	*0.992*	0.429	0.289	*0.991*	0.53	**0.959**	*0.992*	0.21
Average	0.243	0.235	0.239	0.188	0.226	0.467	0.385	0.348	0.308	**0.571**	0.418

holes for the task of link prediction across networks [105]. Our results indicate that such information might further improve the link prediction and help to choose the suitable weighting scheme.

We make another interesting observation for the network based on colleague links. This is the only case where the number of connected components grows over time (Table 5.2). It might be explained by the trend of writing papers within specified groups at hosting institutes and of having little collaboration with outside groups. Though the features based on our new scores perform overall good, there is no consistency in results with $tscore_l^{(1)}$ leading often to extreme AUPR values. Including more information about authors, e.g., which institute they belong to, might improve the performance like it was done in some previous works [118].

Overall, introducing the time awareness improves the performance. However, not in all cases $tscore^{(i)}$ yields better AUPR compared to $score^{(i)}$, especially on the network with colleague links. Still if we use 3-node graphlets which distinguish between ordinary and experienced authors, we gain advantage over $score^{(i)}$. Nevertheless, in case of the Dota2 network with friend links $tscore_l^{(i)}$ reduces the average AUPR in comparison to $tscore^{(i)}$. We think that the gaming experience of users does not really impact the friendship formation among them. It might be more effective to categorize users in this network with regard to being a structural hole. Judging from results both for Dota2 and HepTh networks, we believe that the choice of node categories fits well team mate and peer link types, but we could introduce better node categories for friend and colleague link types, thus reducing the inconsistency.

Friendship and colleague networks are sparse compared to team mate and peer networks. We see that the new score performs well on the latter while the results for the former are inconsistent. Previous works outline that a technique based on counting graphlets is better suited for information networks [31]. It might be the explanation for such inconsistency, however, we show that even in such cases the new scores perform on average well and might further improve supervised models in combination with other features.

Our results lead us to the conclusion that the network evolution is more complex and is not completely captured by one specific feature. Even within the same type of network we observe quite a lot of variance in the performance of features. Similar observation has been already outlined by Davis et al. [31]. However, we want to stress that even combining efficiently the considered features on one dataset at a specific time point does not guarantee that this model will perform equally well over time. It is worth noting that except for the network structure and content we do not use any additional information. However, there can be outside factors which influence the network evolution. For example, Valve (the company which develops Dota2 and Steam) performed many marketing activities to attract new players to their game Dota2 in 2011. Note that we have currently no means to include them into our model. This fact might explain why till the beginning of 2012 we observe quite drastic changes in the performance of features (see Table 5.3).

To complete the picture about the performance of various features for temporal link

prediction, we present AUROC results in the following. In Tables 5.7 and 5.8 we present the results for the Dota2 network. Tables 5.9 and 5.10 show results for the HepTh network. As we see, there is indeed a considerable difference between AUROC and AUPR which is explained by the extreme imbalance between the positive and negative classes when solving the link prediction problem [119]. Often the AUROC results do not show such a drastic difference between the performance of two features which we can clearly observe when studying the AUPR results.

We show ROC and PR curves for the Dota2 network for two time periods, i.e., weeks 201146 and 201203 (see Figure 5.4). In the top of this figure we have the charts with ROC curves. In the bottom we see the charts with PR-curves. The curves provide an opportunity to study the performance of methods on a more fine-grained level. For example, in Figure 5.4(ii) we show the curves for the model which predicts friend links in the Dota2 network for the week 201204 based on the network snapshot up to week 201203. We observe that there is almost no difference in the performance of four methods (namely, JC, PA, $tscore^l_1$ and $tscore^l_2$) in terms of ROC. But if we look at the PR-curve, we see that there is a considerable difference in the performance of these methods. Such difference happens because of the extreme imbalance in the class distribution: the negative class is overrepresented.

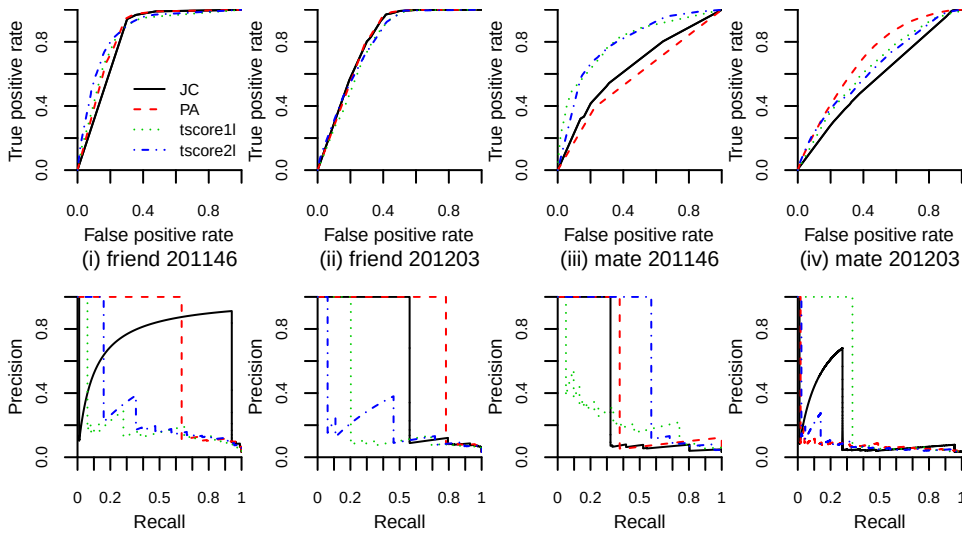


Figure 5.4: ROC and PR curves for dota2 network.

5.6 Summary and Discussion

We have studied the performance of several state-of-the-art and 5 new scores for temporal link prediction in social networks with two types of links. We have performed experiments

Table 5.7: AUROC for predicting mate links.

week	AA	CN	JC	PA	TS	$score^{(1)}$	$score^{(2)}$	$tscore^{(1)}$	$tscore^{(2)}$	$tscore_l^{(1)}$	$tscore_l^{(2)}$
201146	0.688	0.557	0.642	0.583	0.796	0.686	0.513	0.748	0.794	**0.799**	0.793
201147	0.645	0.59	0.53	0.475	0.664	0.645	0.474	0.667	0.679	**0.681**	0.663
201148	0.661	0.621	0.5	0.577	0.672	0.658	0.534	**0.679**	0.665	0.67	0.604
201149	0.652	0.609	0.529	0.581	0.669	0.493	0.567	0.66	0.646	**0.68**	0.643
201150	0.634	0.598	0.525	0.585	0.638	0.567	0.49	0.643	0.627	**0.649**	0.628
201151	**0.641**	0.613	0.527	0.622	0.639	0.606	0.545	0.634	0.613	0.624	0.62
201152	0.637	0.636	0.517	**0.702**	0.634	0.49	0.552	0.5	0.621	0.5	0.62
201201	0.645	0.644	0.537	**0.704**	0.65	0.602	0.646	0.573	0.64	0.611	0.631
201202	0.635	0.636	0.549	**0.689**	0.643	0.601	0.633	0.59	0.619	0.615	0.624
201203	0.66	0.658	0.569	**0.7**	0.673	0.619	0.66	0.601	0.653	0.639	0.633
201204	0.654	0.647	0.585	**0.684**	0.668	0.598	0.647	0.599	0.65	0.602	0.632
201205	0.655	0.647	0.573	**0.685**	0.646	0.617	0.645	0.598	0.65	0.639	0.64
201206	0.633	0.622	0.553	**0.663**	0.627	0.601	0.619	0.587	0.63	0.586	0.623
201207	0.618	0.601	0.538	**0.642**	0.641	0.577	0.578	0.589	0.612	0.625	0.607
Average	0.647	0.62	0.548	0.635	**0.662**	0.597	0.579	0.619	*0.65*	0.637	0.64

Table 5.8: AUROC for predicting friend links.

week	AA	CN	JC	PA	TS	$score^{(1)}$	$score^{(2)}$	$tscore^{(1)}$	$tscore^{(2)}$	$tscore_l^{(1)}$	$tscore_l^{(2)}$
201146	0.838	0.659	0.833	0.853	0.828	0.774	0.72	0.85	0.857	0.845	**0.868**
201147	0.827	0.642	0.833	0.839	0.8	0.748	0.294	0.793	0.829	0.772	**0.839**
201148	0.828	0.641	0.825	0.831	0.809	0.783	0.304	0.816	0.828	0.806	**0.844**
201149	0.806	0.645	0.804	0.829	0.82	0.779	0.436	0.813	0.827	0.798	**0.835**
201150	0.798	0.65	0.795	0.815	0.813	0.773	0.544	0.817	0.811	0.8	**0.82**
201151	0.793	0.652	**0.83**	0.821	0.825	0.778	0.272	0.798	0.783	0.5	0.805
201152	0.798	0.682	**0.819**	0.818	0.808	0.787	0.455	0.786	0.777	0.802	0.778
201201	0.78	0.661	0.795	**0.803**	0.781	0.761	0.306	0.5	0.776	0.801	0.773
201202	0.772	0.659	0.793	0.808	0.775	0.5	0.416	**0.82**	0.772	0.796	0.788
201203	0.782	0.665	**0.811**	0.809	0.81	0.767	0.47	0.794	0.746	0.786	0.799
201204	0.772	0.655	0.777	0.804	0.8	0.774	0.758	0.806	0.793	**0.811**	0.803
201205	0.764	0.66	0.796	0.803	0.798	0.77	0.437	0.789	0.792	**0.81**	0.805
201206	0.77	0.672	0.798	0.8	0.801	0.761	0.748	0.795	0.8	**0.804**	0.804
201207	0.757	0.66	0.773	0.799	0.761	0.772	0.424	**0.812**	0.8	0.805	0.808
Average	0.792	0.657	0.806	**0.817**	0.802	0.752	0.47	0.785	0.799	0.781	*0.812*

Table 5.9: AUROC for predicting peer links.

year	AA	CN	JC	PA	TS	$score^{(1)}$	$score^{(2)}$	$tscore^{(1)}$	$tscore^{(2)}$	$tscore_l^{(1)}$	$tscore_l^{(2)}$
1993	**0.706**	0.693	0.603	0.533	0.691	0.5	0.645	0.5	0.67	0.5	0.584
1994	0.744	0.741	0.659	0.644	**0.753**	0.49	0.696	0.693	0.744	0.751	0.617
1995	0.758	0.754	0.683	0.678	**0.759**	0.699	0.669	0.758	0.755	0.752	0.687
1996	0.776	0.774	0.714	0.71	**0.781**	0.659	0.736	0.756	0.777	0.779	0.763
1997	**0.8**	0.796	0.751	0.732	0.797	0.784	0.714	0.771	0.794	0.735	0.795
1998	**0.812**	0.809	0.761	0.752	0.81	0.801	0.717	0.767	0.809	0.808	0.809
1999	**0.832**	0.827	0.774	0.778	0.827	0.81	0.647	0.825	0.825	0.827	0.827
2000	**0.832**	0.828	0.776	0.776	0.829	0.823	0.717	0.829	0.827	0.828	0.831
2001	**0.859**	0.854	0.81	0.801	0.854	0.845	0.732	0.855	0.851	0.833	0.855
2002	**0.833**	0.827	0.789	0.766	0.824	0.826	0.768	0.782	0.823	0.808	0.827
Average	**0.795**	0.79	0.732	0.717	*0.792*	0.724	0.704	0.754	0.787	0.762	0.759

Table 5.10: AUROC for predicting colleague links.

year	AA	CN	JC	PA	TS	$score^{(1)}$	$score^{(2)}$	$tscore^{(1)}$	$tscore^{(2)}$	$tscore_l^{(1)}$	$tscore_l^{(2)}$
1993	0.49	0.595	0.579	0.494	**0.62**	0.5	0.5	0.5	0.5	0.5	0.5
1994	0.559	0.566	0.525	0.5	0.635	0.551	0.64	**0.668**	0.651	0.562	0.642
1995	0.541	0.579	0.5	0.549	0.582	0.578	0.605	0.564	**0.623**	0.5	0.601
1996	0.561	0.563	0.501	0.529	**0.617**	0.538	0.58	0.616	0.6	0.5	0.585
1997	0.567	0.572	0.548	0.541	0.611	0.581	0.538	**0.629**	0.576	0.5	0.591
1998	0.587	0.572	0.526	0.533	**0.635**	0.582	0.592	0.634	0.557	0.581	0.566
1999	0.586	0.574	0.528	0.491	**0.616**	0.55	0.529	0.611	0.544	0.601	0.548
2000	0.609	0.574	0.549	0.5	0.625	0.598	0.532	**0.643**	0.547	0.5	0.567
2001	0.596	0.56	0.567	0.5	**0.626**	0.564	0.523	0.611	0.536	0.5	0.539
2002	**0.659**	0.602	0.557	0.5	0.651	0.575	0.5	0.628	0.512	0.5	0.549
Average	0.575	0.576	0.538	0.514	**0.622**	0.562	0.554	*0.61*	0.565	0.524	0.569

on two real-world social networks: a gaming network Dota2 with team mate and friend links; a co-author network HepTh with colleague and peer links. We have confirmed once again that considering the temporal aspect of links when studying network evolution is important and leads to the improved link prediction performance. However, our results indicate that the performance of link prediction methods varies over time, and in two cases there is considerable inconsistency in results within the same network type.

We have noticed that the methods which use some variation of a triad counting technique do not perform well on the network with very low average clustering and transitivity coefficients. By using a support measure to estimate the frequencies of graphlets we achieve better performance for temporal link prediction. Another interesting observation is the usage of node labels in the graphlets.

The results of our work point out several directions for the future work. First of all, the categorization of nodes according to structural hole spanning could lead to the further improvement in performance of our new scores $tscore_l^{(i)}$. Secondly, we could introduce time series techniques [30, 118]. We assume that frequencies of 3-node graphlets in the future are the same as they are in the present. As we see from Figures, this assumption is not always true. A way to circumvent this issue is to introduce time series for frequencies and instead of using the present value of the frequency employ the predicted one [30]. Lastly, by efficiently combining the new features we could compare their performance with the state-of-the-art models like HPLP, MRIP and VCP [76, 118].

There is currently a trend to develop supervised models for link prediction by combining classical unsupervised scores (e.g., AA, JC, PA, CN) with new features [76, 118]. These models are then tested on a variety of datasets with the goal to fit as many as possible. We noticed that there might be innate network properties (e.g, average clustering coefficient, presence of structural holes) which could point out appropriate features to explain its future evolution. Instead of designing a general model to fit different networks, we could guide the process of model development with regard to these properties. For example, the methodology based on 3-node graphlets, which we applied in our current work, and more generally the approach of VCPs [76] provide considerable flexibility as to how much additional information (besides the local structure among nodes) is captured by the graphlets, like node labels, link labels, frequency estimation and weighting scheme. For interaction-based networks (like gaming) we might take user activeness to categorize nodes while for relationship-based networks (like co-authorship) considering geographical location of users would lead to better performance of link prediction methods. However, to tune the parameter configuration, we need a bigger pool of networks.

CHAPTER 6

Tools for Graph Pattern Matching

A bad workman always blames his tools.

Proverb

When predicting citation counts or solving the temporal link prediction problem in heterogeneous networks, we need to perform graph pattern matching which is a computationally hard problem. To resolve this issue, we require efficient tools. In this chapter we will discuss such tools. The chapter is based on the joint work with Stefan Ruemmele, Sebastian Skritek and Hannes Werthner. Stefan Ruemmele and Sebastian Skritek provided technical support with the database systems Clingo and Jena TDB. The work appeared in the proceedings of the 25th international conference on Database and Expert Systems Applications 2014 [96].

6.1 Motivation

Networks are one of the most generic data structures and therefore used to model data in various application areas. Their importance has increased, especially because of the social web and big data applications that need to store and analyze huge amounts of highly interconnected data. One well-known task when dealing with graphs is the so-called *graph pattern matching*. Thereby the goal is to find inside a given graph a smaller subgraph, called *pattern*. This allows to explore complex relationships within networks as well as to study and to predict their evolution over time [21]. Indeed, graph pattern matching has lots of applications, for example in software engineering [10], in social networks [39, 67, 122], in bioinformatics [67, 122] and in crime investigation & prevention [39, 115].

Solving graph pattern matching tasks can be done in two ways. The first way is to use specialized algorithms. A comparison of various specialized algorithms for graph pattern matching has been done recently [67]. The second way is to express this problem in the query language of a database system. In the database area the comparison of different systems is an important topic and has a long tradition. Therefore, there exist already studies comparing databases in the context of graph queries [9, 57, 109]. Thereby the authors compare the performance of various databases for different query types, like adjacency queries, reachability queries, and summarization queries. But we want to point out that these works do not study graph pattern matching queries, which are computationally harder. To the best of our knowledge, there is no work on the comparison of database systems for this type of queries.

Additionally, most papers that compare database systems with respect to graph problems, evaluate relational and graph database systems. A family of database systems that is less known in this area is the family of *deductive database systems*. These systems are widely used in the area of artificial intelligence and knowledge representation & reasoning. They are especially tailored for combinatorial problems of high computational complexity. Since graph pattern matching is an NP-complete problem, they lend themselves as a candidate tool for the problem at hand. Another family of database systems that are suitable for the task at hand are RDF-based systems. These databases are tailored to deal with triples that can be seen as labeled edges. Hence, graph problems can be naturally expressed as queries for these systems.

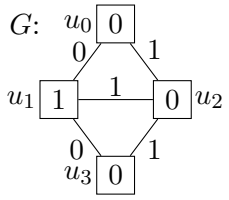
To close the mentioned gaps, we conduct an in-depth comparison of the viability of relational databases, graph databases, deductive databases, and RDF-based systems for solving the graph pattern matching problem. The results of this work are the following:

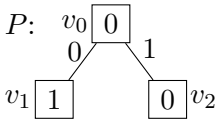
- We build a benchmark set including both synthetic and real data. The synthetic data is created using two different graph models while the real-world datasets include a citation network and a global terrorist organization collaboration network.

- We create sample graph patterns for the synthetic and real-world datasets. Again, part of these patterns are generated randomly. The second set of patterns is created using frequent graph pattern mining. This means we select specific patterns which are guaranteed to occur multiple times in our benchmark set.

- We express the graph pattern matching problem in the query languages of the four database systems we use. These are SQL for relational databases, Cypher for graph databases, ASP for deductive databases, and SPARQL for RDF-based systems.

- We conduct an experimental comparison within a uniform framework using PostgreSQL as an example of relational database system, Jena TDB representing RDF-based systems, Neo4j as a representative for graph databases systems, and Clingo as a deductive database. Based on our experimental results we draw conclusions and offer some general guidance for choosing the right database system.





$M_1 = \{(v_0, u_0), (v_1, u_1), (v_2, u_2)\}$
$M_2 = \{(v_0, u_3), (v_1, u_1), (v_2, u_2)\}$

Figure 6.1: Examples data graph G, pattern graph P, and resulting embeddings M_1, M_2.

6.2 Problem

In this work we deal with *undirected, simple networks*, that means networks without self-loops and with not more than one link between two nodes. We denote a (labeled) network G by a triple $G = (V, E, \lambda)$, where V denotes the set of *nodes*, E is the set of *links*, and λ is a *labeling function* which maps nodes and/or links to a set of labels, e.g. the natural numbers $\mathbb{N}$. A link is a subset of V of cardinality two.

Let $G = (V_G, E_G, \lambda_G)$ and $P = (V_P, E_P, \lambda_P)$ be two networks. An *embedding* of P into G is an injective function $f : V_P \to V_G$ such that for all $x, y \in V_P$:

1. $\{x, y\} \in E_P$ implies that $\{f(x), f(y)\} \in E_G$;
2. $\lambda_P(x) = \lambda_G(f(x))$; and
3. $\lambda_P(\{x, y\}) = \lambda_G(\{f(x), f(y)\})$.

This means if two nodes are connected in P then their images are connected as well. Note that there is no requirement for the images to be disconnected if the original nodes are. This requirement would lead to the notion of *subgraph isomorphism*.

Instead, the problem of *graph pattern matching* is defined as follows: Given two networks, G and P, where P is the smaller one, called *pattern* or pattern graph, the task is to compute all embeddings of P into G. Figure 6.1 shows an example of a network G, a pattern graph P and all possible embeddings of P into G. In practice, we may stop the pattern matching after the first k embeddings are found.

The problem of deciding whether an embedding exists is NP-complete, since a special case of this problem is to decide if a graph contains a clique of certain size, which was shown to be NP-complete [60]. However, if the pattern graph is restricted to a class of graphs of bounded treewidth, then deciding if an embedding exists is fixed-parameter tractable with respect to the size of P, i.e., exponential with respect to the size of P but polynomial with respect to the size of G [5].

6.3 Related Work

Related work includes theoretical foundations of graph pattern matching and benchmarks comparing different databases.

6.3.1 Pattern Matching

Subgraph isomorphism queries are among the most important queries for graph databases [112]. A pattern match query is more flexible than the subgraph isomorphism query and more informative than a simple shortest-path or reachability query. Pattern match queries have gained a lot of attention recently with different extensions being introduced [45]. For example, Zou et al. [122] introduce a distance pattern match query which extends embeddings so that edges of the pattern graph are matched to paths of bounded length in the data graph. In other words, if two vertices are connected in the pattern graph, the shortest path between their images in the data graph has to be bounded by some natural number k. If k equals one, we get the definition of pattern match query from the previous section. The authors demonstrate how such distance pattern match queries can be used in the analysis of friendship, author collaboration and biological networks. Still the problem of deciding whether there is an embedding for a distance pattern match query remains NP-complete. Fan et al. [39] introduce a graph pattern match query with regular expressions as edge constraints. In their work, if two nodes are connected in the pattern graph, their images have to be connected in the data graph within k hops, where k can be either bounded or unbounded. They construct a polynomial time algorithm for this kind of queries and demonstrate the applications in the studies of drug trafficking, assembly network service, recommendation networks and terrorist organization collaboration networks. There is much confusion in the literature with regard to the definitions of subgraph isomorphism and graph pattern match. So, Lee et al. call the graph pattern matching query, which we define here, as a subgraph isomorphism problem [67]. Lee et al. [67] compare several state-of-the-art algorithms for solving graph pattern matching. They compare performance and scalability of algorithms such as VF2, QuickSI, GraphQL, GADDI and SPath. Each of these algorithms uses a different data structure to store graphs.

We investigate how available database systems perform with regard to graph pattern matching. To the best of our knowledge, there is no experimental work on this issue.

6.3.2 Benchmarking database systems for graph analysis

Social web and big data applications deal with highly interconnected data. Modeling this type of data in a relational database causes a high number of many-to-many relations. That is why a number of the so-called NoSQL databases have been developed [107]. Among them, graph databases are especially interesting since they often offer a proper query language. Nevertheless, since these databases are young compared to relational databases, and their query optimizers are not mature enough, it is an open question whether it is worth switching from a relational database to a dedicated graph database. Angles [8] outlines four types of queries on graphs:

- adjacency queries, e.g., list all neighbors of a vertex;
- reachability queries, e.g., compute the shortest path between two nodes;

- pattern matching queries, which are the focus of this paper; and
- summarization queries, e.g., aggregate vertex or edge labels.

There are already several works comparing the performance of current database systems for graph analysis. Dominguez et al. [36] study several graph databases (Neo4j, Jena, HypergraphDB and DEX) as to their performance for reachability and summarization queries. They find that DEX and Neo4j are the most efficient databases.

The other works include not only graph databases, but also relational database systems. Vicknair et al. [109] compare Neo4j, a graph database, and MySQL, a relational database, for a data provenance project. They conclude that Neo4j performs better for adjacency and reachability queries, but MySQL is considerably better for summarization queries. In the comparison they also take into account subjective measures, like maturity, level of support, ease of programming, flexibility and security. They conclude that, due to the lack of security and low maturity in Neo4j, a relational database is preferable. It is worth noting that they used Neo4j v1.0 which did not have a well developed query language and was much less mature than MySQL v5.1.42.

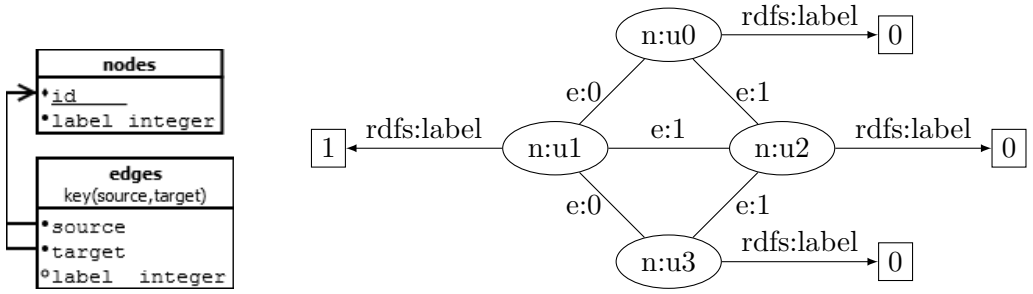
Holzschuher and Peinl [57] show that Neo4j v1.8 is much more efficient for graph traversals than MySQL. Angles et al. [9] extend the list of considered databases and include two graph databases (Neo4j and DEX), one RDF-based database (RDF-3X), and two relational databases (Virtuoso and PostgreSQL) in their benchmark. They show that DEX and Neo4j are the best performing database systems for adjacency and reachability queries. However, none of these works consider graph pattern matching queries.

6.4 Benchmark for Graph Pattern Matching

Our benchmark consists of three main components: database systems, datasets and query sets. The first component includes the choice of the systems, their setup, the used data representation and encodings of pattern graphs in a specific query language. The second component consists of data graphs that are synthetically generated according to established graph models or chosen from real-world data. The last component contains the construction of pattern graphs which are then transformed into queries according to the first component and used on the datasets from the second component.

6.4.1 Database Systems

The database systems, which we compare, are PostgreSQL, Neo4j, Clingo, and Jena TDB. These systems have in common that they are open source and known to perform well in their respective area. But they differ considerably in the way they store data and the algorithms used to execute queries. However, all four systems allow to execute the four mentioned types of graph queries. We present in this section the data schema in each of these systems as well as the query statements in four different query languages. The used data schemas are general purpose schemas for graph representation and are



(a) Entity relationship diagram of a graph in PostgreSQL.

(b) RDF representation for the example data graph in Figure 6.1.

Figure 6.2: Data schema for PostreSQL and Jena TDB.

not specifically tailored to graph pattern matching. We do not use embedded database systems, but rather server/client like in real applications. Client application is in Python.

PostgreSQL (SQL)

PostgreSQL is an open-source *object-relational database system* with SQL being the query language. We use PostgreSQL server v9.1.9. The data schema we use for a graph consists of two tables: nodes and edges (see Figure 6.2(a)). The primary key of the table "nodes" is the attribute "id" and it contains an index over the attribute "label". The attribute "label" is also indexed in the table "edges". The attributes "source" and "target" constitute the primary key of this table. Since we are dealing with undirected graphs, the edge table contains two entries per edge. The SQL query we use to find all embeddings of the pattern graph in Figure 6.1 is the following:

```
select v0.id, v1.id, v2.id
from nodes v0, nodes v1, nodes v2, edges e0, edges e1
where v0.label=0 and v1.label=1 and v2.label=0 and
      v0.id<>v2.id and
      e0.source=v0.id and e0.target=v1.id and e0.label=0 and
      e1.source=v0.id and e1.target=v2.id and e1.label=1;
```

Listing 6.1: SQL query for the pattern graph from Figure 6.1.

As we see, we need to do many joins for the tables "nodes" and "edges" corresponding to the amounts of vertices and edges in the pattern graph. The way the query is written, we leave it up to the database query optimizer to define the join order.

It is possible to use a denormalized data schema when we have only one table which contains all information about vertices and edges. One can also manually optimize the query. However, this is not the scope of the paper. The same data schema has been used in previous benchmarking works [9, 109]. Also, our focus is at how the database engine can optimize the query execution. Such setting corresponds to a typical production scenario when an average user is not an expert.

Jena TDB (SPARQL)

In RDF (Resource Description Framework), entities comprise of triples (*subject*, *predicate*, *object*). The interpretation is that the subject applies a predicate to the object. RDF-based data can be also regarded as a graph where the subject and the object are nodes, and predicate corresponds to a link between them. There exist several RDF-based database systems. We choose Jena which is an open-source framework for building RDF based applications and provides several methods to store and interact with the data. For our purposes, the SPARQL server Fuseki provides the database server that is used for query answering. We use Fuseki 1.0.1, with the included Jena TDB for the persistent storage of the data. There are two main reasons for choosing TDB over SDB. Firstly, TDB was repeatedly reported to provide better performance. Secondly, especially for the comparison with PostgresSQL, it is convenient to use the native triple store TDB instead of SDB that is backed up by an SQL database.

We encode the graph in RDF by representing each node of the graph as a resource (i.e., we generate an IRI for every node). The edge relation is described as properties of these resources that state to which other nodes a node is connected. Since the choice of the node that appears in the subject position of the RDF triple implies an order on the edges, we create two triples for each edge where we switch the subject and object positions. In the presence of edge labels, we introduce a separate property for each label. As a result, the structure of the RDF data resembles the original graph (cf. Figure 6.2(b)).

The following query looks for embeddings of the graph pattern from Figure 6.1:

```
SELECT ?X0 ?X1 ?X2
WHERE {?X0 e:0 ?X1 . ?X0 e:1 ?X2 .
  ?X0 a t:node . ?X0 rdfs:label ``0'' .
  ?X1 a t:node . ?X1 rdfs:label ``1'' .
  ?X2 a t:node . ?X2 rdfs:label ``0'' .
  FILTER ( (?X0 ≠ ?X1) && (?X0 ≠ ?X2) && (?X1 ≠ ?X2))}
```

Listing 6.2: SPARQL query for the pattern in Figure 6.1 (omitting prefix definitions).

Neo4j (Cypher)

Neo4j is a *graph database system* with Cypher being its own query language. Like SQL, Cypher is not only a query language but also allows data manipulation like updates and deletes. Neo4j does not rely on a relational data layout, but a network model storage that natively stores nodes (vertices), relationships (edges) and attributes (vertex and edge labels). Neo4j is written in Java and has a dual free/commercial license model. Neo4j is fully written in Java and can be deployed on multiple systems. It supports transactions and fulfills the ACID consistence properties. Currently Neo4j can handle graphs with up to 2^{35} vertices and 2^{35} edges. We use Neo4j v1.9 which introduced considerable enhancements and optimization to the query language Cypher.

It is possible to access and to modify data in Neo4j either with Cypher queries or directly via a Java API. Additionally, Neo4j can be embedded into the application or accessed via REST API. Experimental results show that Cypher via REST API performs

slower than Cypher in embedded mode [57]. It is clear that queries using an embedded instance are faster than those accessing the database over the network. However, since we deploy the relational and RDF databases as servers and send queries from a client application, we also use REST API to send Cypher queries to the Neo4j server. Moreover, such a configuration models most real-world scenarios more accurately. The data schema of a graph in Neo4j corresponds to the representation in Figure 6.1. Vertex and edge labels are indexed with Lucene.

There are several ways to express a pattern match query with Cypher. Since Cypher is the least mature query language in our benchmark, we use two encodings of patten graphs in Cypher. The most straightforward way is to start with matching one vertex from the pattern graph, and match all edges in one "MATCH" statement of the Cypher query. We cannot specify all vertices as the starting nodes in Cypher, since it results in a completely different set of answers. This shortcoming is unfortunate since the user is left with the task to choose the most appropriate starting node.

```
start v0 = node:my_nodes(label='0')
match v0-[e0]-v1, v0-[e1]-v2
where v1.label=1 and v2.label=0 and
      id(v0)<>id(v2) and
      e0.label=0 and e1.label=1
return id(v0), id(v1), id(v2);
```

Listing 6.3: Cypher query for the pattern graph from Figure 6.1.

As an alternative, it is possible to write nested queries in Cypher. This allows to match the pattern graph one edge at a time and transfer the intermediate results to the next level:

```
START v0 = node:my_nodes(label='0') MATCH v0-[e0]-v1
WHERE v1.label=1 and e0.label=0
WITH v0, v1 MATCH v0-[e1]-v2
WHERE v2.label=0 and id(v0)<>id(v2) and e1.label=1
RETURN id(v0), id(v1), id(v2);
```

Listing 6.4: Nested Cypher query for the pattern graph from Figure 6.1.

The Neo4j developers mention that the nested queries might be especially good if the pattern graph is complicated. In both cases, straightforward and nested, we could further improve the queries by choosing more intelligently the starting node and the order in which we match the edges. Again, we do not apply these modifications since the scope of this work is on how well the database system itself can optimize the query execution. Due to space restrictions and readability issues, we report only the performance of the nested Cypher query since it shows consistently better results on our benchmark than the straightforward implementation.

Clingo (ASP)

Answer-set programming (ASP) is a paradigm for declarative problem solving with many applications, especially in the area of artificial intelligence (AI) and knowledge

representation & reasoning (KR). In ASP a problem is modeled in the form of logic program in a way such that the so-called stable models of the program correspond to the solutions of the problem. The stable model semantics for logic programs can be computed by ASP solvers like Clingo [47], DLV [68], Smodels [104], or others. We use Clingo v4.2.1 because of its performance at various ASP competitions [6].

In database terminology, Clingo is a *deductive database*, supporting ASP as query language. Data is represented by facts (e.g., vertices and edges) and rules from which new facts (e.g., the embeddings) are derived. For example, the data graph from Figure 6.1 is given as a text file of facts in the following form.

$v(0,0).\ v(1,1).\ v(2,0).\ v(3,0).\ e(0,1,0).\ e(0,2,1).\ e(1,2,1).\ e(1,3,0).\ e(2,3,1).$

The first argument of the vertex predicate v indicates the node ID, the second one the label. For the edge predicate e the first two arguments represent the ID's of the connected nodes and the third argument corresponds to the edge label: Note that we have omitted here half of the edges. To model undirected edges, we have two facts corresponding to each edge. For example, the dual version of the first edge fact above would be $e(1,0,0)$.

The ASP encoding for our running example is shown below:

```
1 {match(0,X) : v(X,0)} 1.
1 {match(1,X): e(Y,X,0)} 1 ← match(0,Y).
← match(1,X), not v(X,1).
1 {match(2,X): e(Y,X,1)} 1 ← match(0,Y).
← match(2,X), not v(X,0).
← node(K,X), node(L,X), K ≠ L.
```

Listing 6.5: ASP query for the pattern graph from Figure 6.1.

In this encoding, we derive a new binary predicate *match* where the first argument indicates the ID of a vertex in the pattern graph and the second argument corresponds to the ID of a vertex in the data graph.

This encoding follows the "guess and check" paradigm. The *match* predicates are guessed as follows. The rule in Line 1 states that from all variables X such that we have a fact $v(X,0)$, i.e. all vertices with label 0, we choose exactly one at random for our pattern node 0. The rule in Line 2 states that from all variables X such that there exists an edge with label 0 to a node Y which we have chosen as our pattern node 0, we choose exactly one at random for our pattern node 1. Finally, we have constraints in this encoding, which basically throw away results where the guess was wrong. For example, Line 3 is such a constraint which states that a guess for variable X as our pattern node 1 is invalid if the corresponding vertex in the data graph does not have label 1.

Datalog is more expressive than SQL. ASP is a further extension which means that this language is even more expressive. It is possible to express Datalog and SQL queries in ASP. ASP has proven to be especially good for combinatorial problems.

6.4.2 Datasets

We use both synthetic and real data. The synthetic datasets include two types of networks: small-world and erdos renyi networks. *Erdos Renyi Model (ERM)* is a classical random

Table 6.1: Summary of the datasets.

Dataset	*Synthetic data*				*Real data*	
	ERM 1000	ERM 10000	PAM 1000	PAM 10000	GTON	HepTh
# vertices	1,000	10,000	1,000	10,000	335	9,162
# edges	4,890	500,065	3,984	39,984	335	52,995
avg degree	9.78	100.01	7.97	8	1.98	11.57
max degree	22	143	104	298	13	430
# vertex labels	2	2	2	2	2	5
# edge labels	–	–	–	–	2	5

graph model. It defines a random graph as n vertices connected by m edges, chosen randomly from the $n(n-1)/2$ possible edges. The probability for edge creation is given by the parameter p. We use parameter $p = 0.01$. This graph is connected.

Preferential Attachment Model (PAM), or small-world model, grows a graph of n nodes by attaching new nodes each with m edges that are preferentially attached to existing nodes with high degree. We use $m = 4$. We choose this graph generation model, since it has been shown that many real-world networks follow this model [12]. In both cases, we generate graphs with 1000 and 10,000 nodes. Vertex labels are assigned randomly. We do not produce edge labels for the synthetic datasets.

The real-world datasets include a citation network and a terrorist organization collaboration network. *The terrorist organization collaboration network (GTON)* is constructed on the basis of Global Terrorism Database[1] which contains 81,800 worldwide terrorist attack events in the last 40 years. In this network, each vertex represents a terrorist organization, and edges correspond to the collaboration of organizations in the common attacks. Vertices are assigned two labels according to the number of recorded events: either 0 if the organization conducted less than 2 attacks, or 1 otherwise. Edges have also two labels depending on the amount of common attacks: either 0 if two organizations collaborated less than twice, or 1 in the other case.

The citation network (HepTh) covers arXiv papers from the years 1992–2003 which are categorized as High Energy Physics Theory. This dataset was part of the KDD Cup 2003 [48]. In this network, a vertex corresponds to a scientific publication. An edge between two vertices indicates that one of the papers cites the second one. We ignore the direction of the citation, and consider the network undirected. As vertex labels, we use the grouped number of authors of the corresponding paper. The edge label corresponds to the absolute difference between publication years of the adjacent vertices.

We summarize the statistics of the constructed data graphs in Table 6.1. Except for the standard graph measures, we also report how much space each data graph occupies in a specific database on the disk. It is not surprising that Clingo databases are the smallest since there is no indexing and each database is a separate text file. It is though striking that databases of smaller graphs have a smaller size in Neo4j than in PostreSQL, but the bigger graphs on the contrary occupy less space in PostgreSQL.

We use Gephi to visualize some of the constructed networks [14]. We apply the Louvain

[1] `http://www.start.umd.edu/gtd`

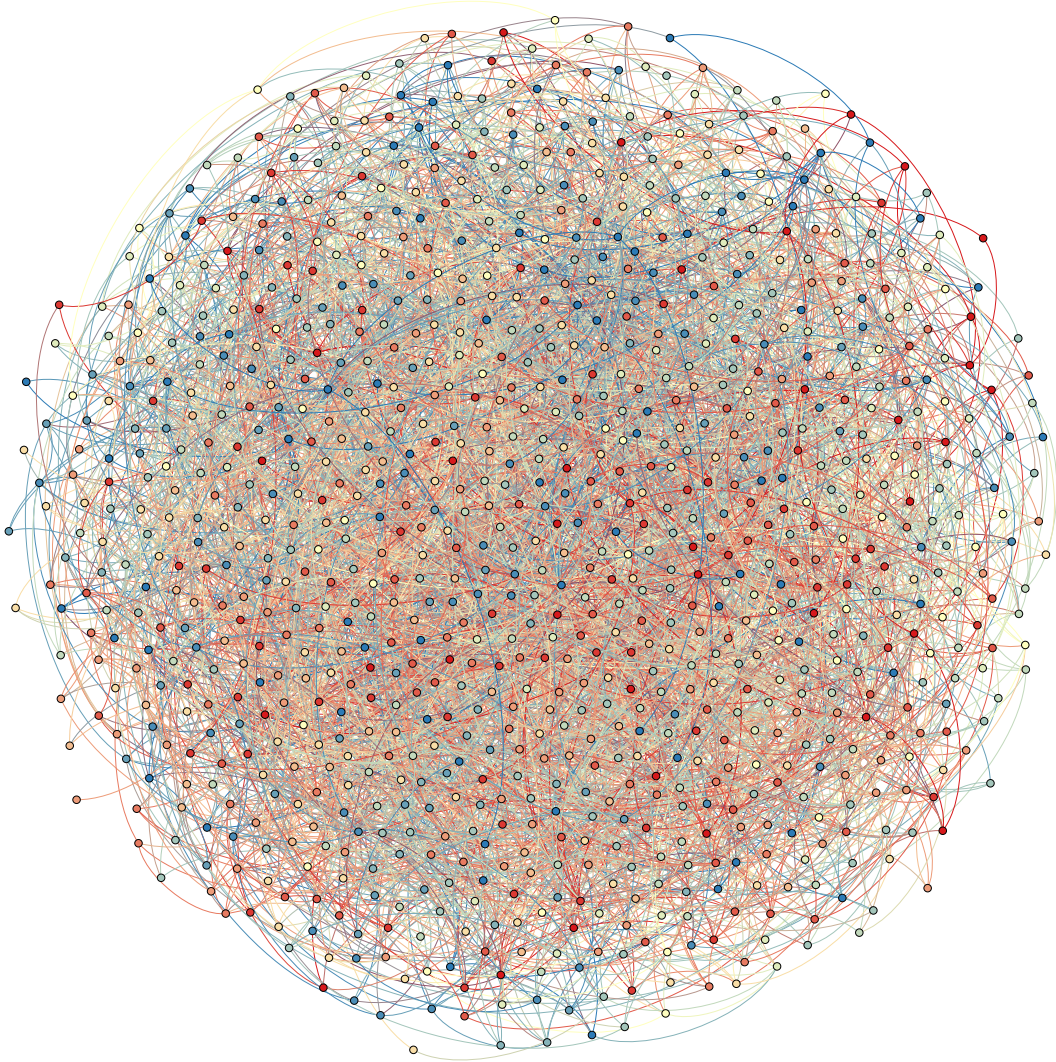


Figure 6.3: Visualization of a network generated via Erdos Renyi model with 1000 nodes. Colors of the nodes correspond to the detected communities in the network.

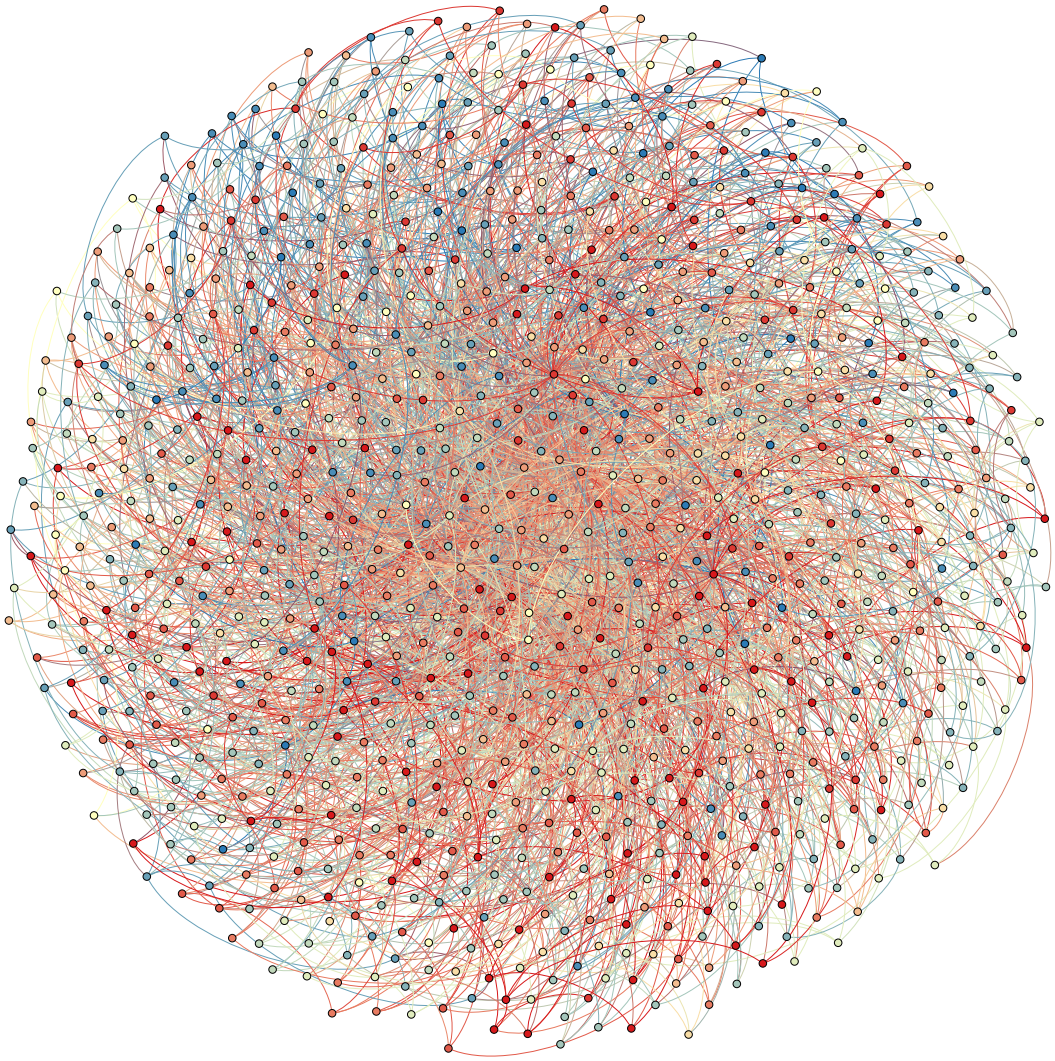


Figure 6.4: Visualization of a network generated via preferential attachment model with 1000 nodes. Colors of the nodes correspond to the detected communities in the network.

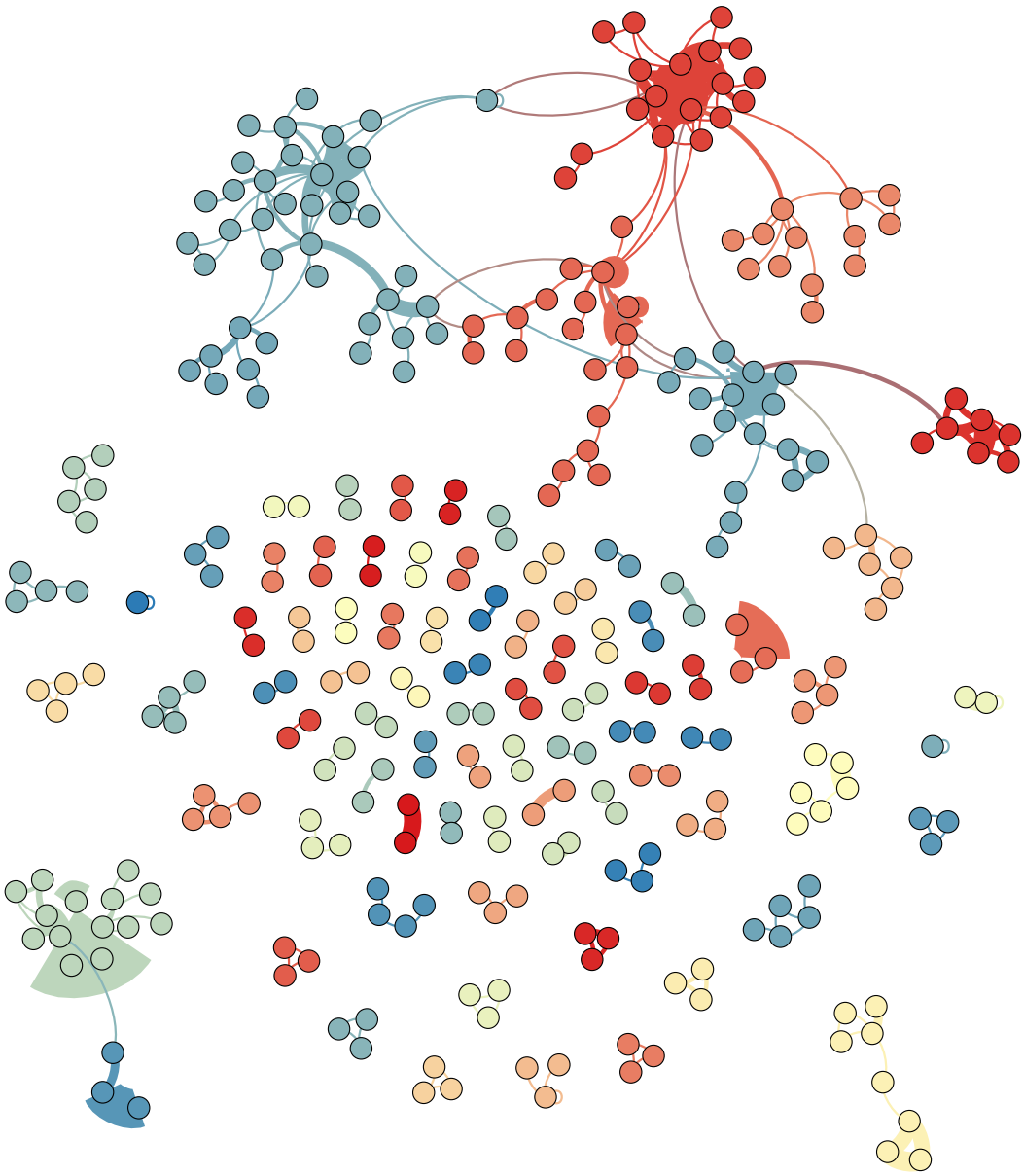


Figure 6.5: Visualization of the terrorist organization collaboration network. Colors of the nodes correspond to the detected communities in the network.

method to discover communities in each network and color the nodes in the networks according to the community they belong to [18]. In Figure 6.3 we show the obtained result for the network generated via Erdos Renyi model with 1000 nodes. Figure 6.4 contains the visualization of the network generated via the preferential attachment model. We show the structure of the global terrorist collaboration network in Figure 6.5. Since this is a weighted network, the width of the links in the figure is proportional to the weight of the link.

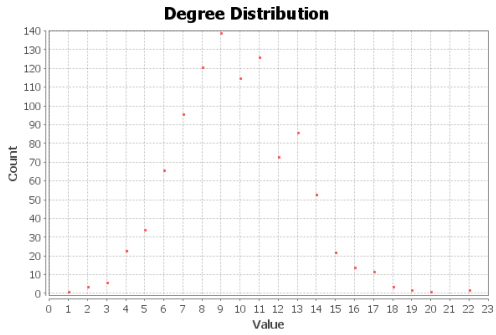
By analyzing the visualizations, we can see that the real-world network is much more sparse than the generated networks (in other words, it has a much smaller amount of links). This network is disconnected unlike the generated networks. In the case of the generated networks we could have done them disconnected by putting together two networks generated independently via the corresponding model. Since we do not loose any generality power, we consider only connected networks which are generated via the models.

As mentioned in Chapter 2, degree distribution of the nodes is an important characteristic of the network. We show the degree distributions of three aforementioned networks in Figure 6.6. We can see that the degree distribution of ERM follows the normal distribution while for the other two networks the distributions correspond to the power-law degree distribution.

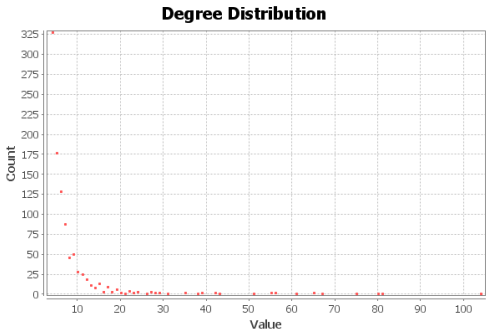
Another interesting aspect which we can observe in the visualized networks is the community structure (see Chapter 2 for definition). We clearly see tightly-knit communities in GTON and the bridges which connect different communities. Though the degree distribution of PAM1000 is also power-law, we do not observe such a clear community structure in this network. In PAM1000 we can see the few highly-connected nodes in the center while the rest of nodes have much lower degrees. In ERM1000 there is a big amount of nodes which have approximately the same degree, and on the periphery we see not so many nodes with low degrees. Thus, we have networks with quite different topologies in our benchmark which allows us to estimate how the structure of the network influences the performance of a graph pattern matching algorithm.

6.4.3 Query Sets

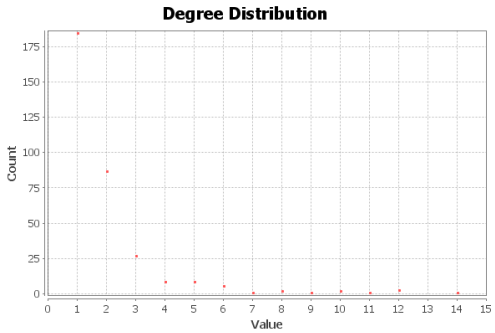
We produce two sets of pattern graphs which are then used in graph pattern matching queries. All the generated pattern graphs are connected. The first set is generated synthetically with a procedure which takes as input the number of vertices and number of edges. The queries in this set have only vertex labels. In the first run of the procedure, we generate queries with five vertices and vary the number of edges from 4 till 10. In the second run, we fix the number of edges to ten and vary the number of vertices from 5 till 11. We generate 20 pattern graphs for each parameter configuration in both cases. We construct the synthetic queries this way in order to verify how the performance of database systems is influenced by the size of the pattern graph: first, we focus on the dependence on the number of edges, and second, on the number of vertices. We call this set of pattern graphs *synthetic patterns*.



(a) ERM1000.



(b) PAM1000.



(c) GTON.

Figure 6.6: Degree distributions of three constructed networks.

The second set of queries is generated specifically for real-world data using graph pattern mining. Thereby we look for patterns which occur at least five times in our data graphs. In this set we specify not only vertex labels but also edge labels. The reason for this set of queries is twofold. First, we can study the performance on pattern graphs with guaranteed embeddings. Second, frequent graph pattern mining is a typical application scenario for graph pattern matching [45]. For example, graph pattern mining together with matching is used to predict citation counts for HepTh dataset in Chapter 4.

6.5 Experimental Results

6.5.1 Experimental setup

The server and the client application are hosted on the same 64 bit Linux machine. It has four AMD Opteron CPUs at 2.4GHz and 8GB of RAM. The client application is written in Python and uses wrappers to connect to the database systems. In our scenario we reduce the run time by hosting the client application on the server, and thus saving the overhead due to the network communication. The warm-up procedure for PostgreSQL, SPARQL and Neo4j consists of executing several pattern match queries. Since Clingo

does not have a caching mechanism, we do not perform a warm-up procedure for it. Since the problem at hand is NP-complete, we limit the execution time for all queries and database systems to three minutes to avoid long-lasting queries and abnormal memory consumption. We use the same order of edges in pattern graphs when encoding them into queries in different database languages. Except for our smallest dataset, GTON, we query only for the first 1000 embeddings. As Lee et al. [67] point out, this limit on embeddings is reasonable in practice. We would like to stress that all systems provide correct answers and agree on the number of discovered embeddings.

6.5.2 Synthetic data

We report the performance of the database systems for synthetic queries on the four synthetic datasets in Figures 6.7 and 6.8. The performance of PostgreSQL is labeled by "SQL". The label "Cypher" stands for the nested Cypher query. Label "SPARQL" and "ASP" show the performance of Jena TDB and Clingo correspondingly. The performance is measured in terms of the execution time per query, and we plot the execution times on a logarithmic scale for better visualization. We also consider how many queries the database system manages to execute within three minutes.

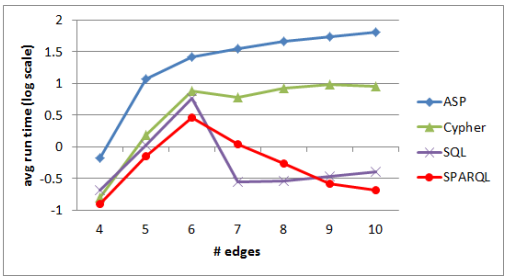
Charts (a) and (c) in the figures correspond to the case where we use pattern graphs with five vertices and change the number of edges from four till ten. We have 20 distinct queries for each category, this means each data point corresponds to the average execution time over 20 queries. Since we consider only connected patterns, all patterns with five vertices and four edges are acyclic. With increasing number of edges, the number of cycles in patterns increases. Patterns with ten edges are complete graphs. Moreover, for these patterns the principle of containment holds: pattern graphs with less edges are subgraphs to some pattern graphs with more edges. Hence, the number of found embeddings can only drop or remain the same as we increase the number of edges.

In charts (b) and (d) from Figures 6.7 and 6.8 we start with the same complete pattern graphs as in the last category in charts (a) and (c). Then, by fixing the number of edges to 10, we increase the number of vertices till 11. Pattern graphs with 11 vertices and 10 edges are again acyclic graphs. By construction, the principle of containment does not hold here. Hence, in charts (a) and (c) we investigate how the increasing number of edges influences the performance of the database systems. For charts (b) and (d), the focus is on the dependence between the number of vertices and execution time.

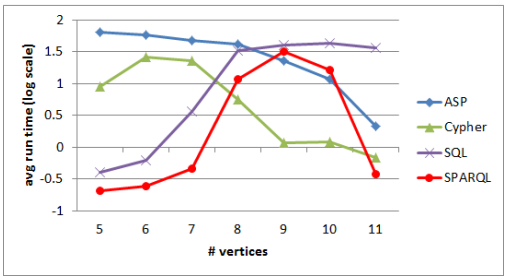
Another aspect studied is the scalability of the database systems with regard to the size of the data graph. Thus, in Figures 6.7 and 6.8 charts (a) and (b) correspond to smaller graphs while charts (c) and (d) show results for bigger ones. Since we observe that the performance of the systems also depends on the structure of the pattern, we present the dependence of the run time on the number of cycles in the pattern in Figure 6.9.

The results indicate that SPARQL is on average better than the others. We observe that the performance of the database systems depends on the following factors: (I) size of the data graph; (II) size of the pattern graph; and (III) structure of the pattern graph.

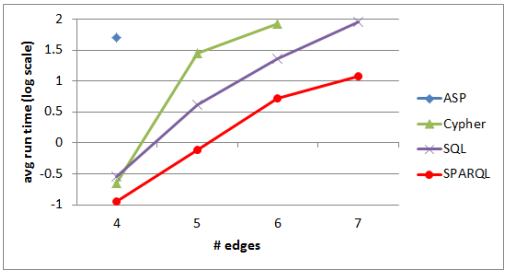
PostgreSQL shows incoherent results. For example, we can observe a peak in the average run time for the pattern graphs with five vertices and six edges for small datasets



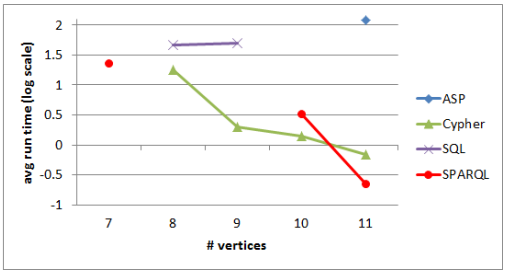
(a) Patterns with 5 vertices and varying number of edges for ERM1000.



(b) Patterns with 10 edges and varying number of vertices for ERM1000.



(c) Patterns with 5 vertices and varying number of edges for ERM10000.



(d) Patterns with 10 edges and varying number of vertices for ERM10000.

Figure 6.7: Average run time in seconds on logarithmic scale for ERM.

in PostgreSQL (see Figure 6.7(a),6.8(a)). One reason for this behavior is that in some cases the query optimizer fails in determining the best join strategy (we recall that PostgreSQL offers nested loop-, hash-, and merge join). For example, by disabling the usage of the nested loop join in the configuration of PostgreSQL, we arrive at an average run time of two seconds instead of eight for the dataset PAM1000 for synthetic patterns. However, this trick works only for the smaller graphs.

Overall, PostgreSQL does not scale with regard to the size of the pattern graph. We can see it especially in Figure 6.7(b). Furthermore, none of the queries in Figure 6.7(d) finished within 3 minutes. The reason is that the database query optimizer decides the join order in the execution plan as well as the join strategy for each pattern query. There are three join strategies in PostgreSQL. The nested loop join is the most fundamental join algorithm. Here, the outer loop fetches the results from one table, and the second (inner) loop retrieves for each row from the outer loop the corresponding data from the other table. The hash join, instead, loads the candidate rows from one side of the join into a hash table that can be probed very quickly for each row from the other side of the join. Hash joins are very fast, but require more memory. The third strategy is called merge join. Thereby the two tables are sorted first and then scanned in parallel. Depending on the choice of the join order and strategy selected by the query optimizer, the run times may differ considerably. For example, we can observe a peak in the average run time for the pattern graphs with five vertices and six edges for small datasets in PostgreSQL

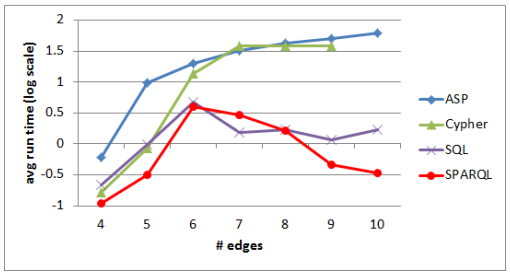
(see Figure 6.7(a),6.8(a)). By influencing the query optimizer it is possible to improve the execution time in this case. For example, by disabling the usage of the nested loop in the configuration settings of PostgreSQL, we arrive at an average run time of two seconds instead of eight for the dataset PAM1000 for synthetic patterns. However, this trick works only for the smaller graphs. Overall, PostgreSQL does not scale with regard to the size of the pattern graph. We can see it especially in Figure 6.7(b). Furthermore, none of the queries in Figure 6.7(d) finished within 3 minutes.

Surprisingly, SPARQL shows the best performance in almost all cases. This complements previous works ([9] and [36]) where RDF-based systems are among the worst performing systems for the tasks of graph analysis, except our case of graph pattern matching. Our understanding is that, besides solving a different problem, the authors use native access to Neo4j in [36], and access to Neo4j through Cypher via REST API has been shown to be much slower [57]. Also, Renzo et al. [9] chose RDF-3X which seems to perform worse than Jena TDB. Like PostgreSQL, Jena TDB shows incoherence, e.g., there is a peak in average run time for patterns with five vertices and six edges (see Figures 6.7(a) and 6.8(a)). In Figure 6.9 we observe that SPARQL can handle pattern graphs with two cycles considerably worse than the others. It seems that the query optimizer of Jena TDB makes worse choices for the matching order of edges, however, it is unclear if such behavior can be configured since the query optimizer is not transparent.

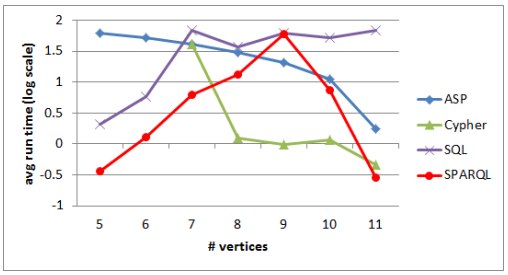
Though SPARQL shows better run times, it cannot handle efficiently complete patterns on ERM10000 like the other database systems (see Figure 6.7(d)). SPARQL turned out to be very sensitive towards the changes in the data schema. If we put edges as resources in Jena TDB, it becomes the worst performing database system. Such change in the data schema might be relevant if we introduce more than one property for the edges or we have a string property associated with the edges. At the same time, such drastic changes to the data schema are not required for the other database systems.

Our results show that Clingo is not a scalable system with regard to the size of the data graph. It cannot execute any query for cyclic patterns within three minutes on ERM with 10000 vertices (Figure 6.7(c),6.7(d)), but can only handle acyclic patterns on these datasets. Though the size of the pattern affects the run time of Clingo, the decrease in the run time happens mainly due to the growth of cycles in the pattern graph (Figure 6.9).

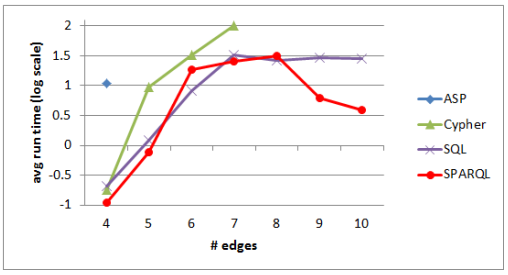
Like for Clingo, cyclic patterns pose the main problem for Neo4j. The average run times grow with the increase of the number of edges in charts (a) and (c), and then drop with the increase of the number of vertices in charts (b) and (d) in Figures 6.7 and 6.8. Thus, the worst run time is for complete patterns. This trend is illustrated in Figure 6.9. Unlike ASP, the dependence between the run time and number of cycles is not linear for Cypher. The issue is that both Cypher and ASP rely on backtracking algorithms. Hence, the starting vertex in the queries is very important for the performance. We could further optimize the queries by choosing the vertex with the least frequent label as the starting point. This change is especially crucial for Neo4j, since its query optimizer is not as mature as the one for PostgreSQL or Clingo. When analyzing the change of average run times between the small and big data graphs in each figure, we see that the performance



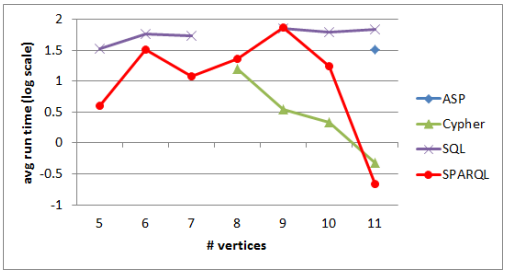
(a) Patterns with 5 vertices and varying number of edges for PAM1000.



(b) Patterns with 10 edges and varying number of vertices for PAM1000.



(c) Patterns with 5 vertices and varying number of edges for PAM10000.



(d) Patterns with 10 edges and varying number of vertices for PAM10000.

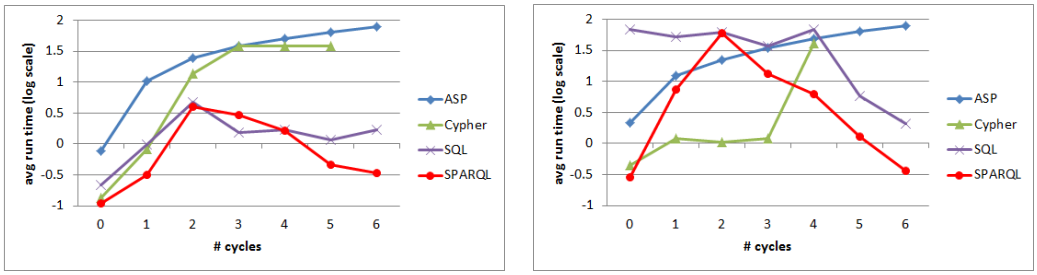
Figure 6.8: Average run time in seconds on logarithmic scale for PAM.

of ASP drastically drops while Cypher shows the least change in performance among the considered systems. This trend is especially clear for ERM (Figure 6.7).

As mentioned, database systems not always manage to execute our queries within three minutes. PostgreSQL finishes not all queries with 10 edges on the small PAM and ERM graphs within the time limit. Especially problems arise with patterns which have more than 9 vertices. For example, PostgreSQL finishes only two queries out of 20 on the small PAM graph for pattern graphs with 10 edges and 11 vertices. Furthermore, on both big graphs (PAM10000 and ERM10000) PostgreSQL can execute only a fraction of queries for pattern graphs with more than six edges. Clingo, Neo4j and SPARQL execute all queries for acyclic patterns within the established time limit irrelevant of the size of data and pattern graphs. An interesting observation is that Clingo either executes all queries within the specified category, or none at all. While Neo4j can handle more cyclic patterns compared to Clingo, the number of executed queries within three minutes still drops with the increase of the number of cycles.

6.5.3 Real world data

Synthetic patterns do not have edge labels, and for many of them no embedding exists in the data graphs. Therefore, we construct another set of patterns by ensuring that each of them occurs at least 5 times in the dataset. Hence, the sets of generated patterns



(a) Patterns with 5 vertices and varying number of cycles.

(b) Patterns with 10 edges and varying number of cycles.

Figure 6.9: Average run time in seconds on log-scale for synthetic patterns on PAM1000.

differ for GTON and HepTh. In both cases graph patterns have from 5 up to 10 edges, and almost all of them are acyclic. More precisely, the generated patterns have either the same number of vertices as edges, or they have one vertex more than edges. The performance for this setting is shown in Figure 6.10. Again, we use 20 pattern graphs in each category, which corresponds to a data point in the charts, and report the average run time for each category on the logarithmic scale in the figure.

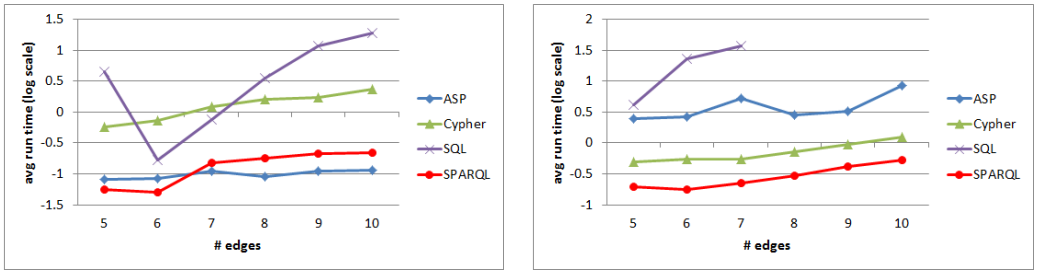
Clingo and Jena TDB show equally good performance on our smallest dataset GTON (Figure 6.10(a)) with Clingo being better for the bigger pattern graphs. Neo4j and PostgreSQL are considerably worse in this case. However, ASP drops in its performance on the dataset HepTh (Figure 6.10(b)). Clingo does not use any indexes on labels and has no caching mechanism. This leads to a considerable drop in the performance when the data graph is especially big. At the same time, SPARQL shows the best run times on the bigger dataset, though compared to GTON the run times increase.

Surprisingly, PostgreSQL does not provide the best performing results in any case on the real-world data. We can observe a clear trend that the average run time for PostgreSQL considerably grows with the increase of the number of edges in the pattern graphs. Moreover, PostgreSQL executes only five queries out of 20 for pattern graph with 7 edges on HepTh. This observation proves once again that PostgreSQL does not scale with regard to the size of the pattern graphs. Since the more edges there are in the pattern graph, the more joins for tables PostgreSQL has to do.

In terms of scalability with the size of the data graph, we conclude that Neo4j shows the best results. Judging from the results on GTON, we may conclude that there is a lot of space for improvement for Neo4j. We believe that the query optimizer in Neo4j could be tuned to narrow the gap between Cypher and SPARQL in this case.

6.5.4 Recommendation

As a result, we can provide the following insights. In general, Jena TDB is better for graph pattern matching with regard to the data schema provided in our benchmark. If we have a very small data graph, Clingo is a good choice. If we have a big data graph



(a) Frequent pattern queries for GTON. (b) Frequent pattern queries for HepTh.

Figure 6.10: Average run time in seconds on logarithmic scale for GTON and HepTH.

and pattern graphs are mainly acyclic, Neo4j provides good results. However, in case of big data graphs and cyclic pattern graphs with more than seven edges, none of the studied database systems perform well.

6.6 Summary and Discussion

We have studied how four database systems perform with regard to the problem of graph pattern matching. In our study we use a relational database PostgreSQL, a graph database Neo4j, a deductive database Clingo and RDF-based database Jena TDB. The most mature system among these four is PostgreSQL with Neo4j being the youngest. By conducting extensive experiments on synthetic and real-world datasets we come to the following conclusions.

Clingo does not scale with the size of the data graph. Its performance drastically drops when the data graph becomes bigger. Though the performance of Clingo is not much influenced by the size of the pattern graph, it worsens with the growth of the cycles in the pattern graph. Neo4j cannot efficiently handle cyclic pattern graphs. However, it scales very well with regard to the size of the data graph as well as the size of the pattern graph. The performance of PostgreSQL oscillates generally due to the changes in the join order chosen in the execution plan. Though PostgreSQL shows good performance for cyclic patterns, it does not scale well with regard to the size of the pattern graphs and the size of the data graphs. Jena TDB is in general better than the other database systems on our benchmark. However, it turned out to be the most sensitive system to the changes in the data schema.

In our opinion, the efficiency of databases for solving graph pattern matching tasks is not yet good enough for information systems that deal with real-world big data scenarios. The database systems should consider implementing the state-of-the-art algorithms for this task [67]. For example, the query optimizer of Neo4j cannot ensure the most optimal choice of the starting node, and of the join order for the links and nodes. The latter holds also for PostgreSQL which can be optimized by tuning up the configuration settings of the server. Furthermore, in all database systems we can configure the servers to achieve

better results, but it is unclear what is the best configuration if we need to perform a variety of graph queries and not just graph pattern matching. This calls for further investigation. We plan to investigate how the efficiency can be increased by influencing the matching order of the links. Future work includes the integration of other types of graph queries into our benchmark.

CHAPTER 7

Conclusion

> "Begin at the beginning," the King said gravely, "and go on till you come to the end: then stop."
>
> *Lewis Carroll*, Alice in Wonderland

Link prediction is an important problem in the field of network analysis which has attracted much attention from academic and industrial researchers in the recent years. It is a young area of research within computer science dating back to the seminal work by Liben-Nowell and Kleinberg in 2004 [74] Hence, there are many open issues, including proper evaluation, application scenarios and best methods to solve the problem. The first link prediction methods have been developed to infer missing links in networks since in many scenarios complete data is not available. Our focus is on predicting links in the network over time.

7.1 Summary

The goal of this thesis was to advance the state-of-the-art of graph pattern based temporal link prediction. Hereby we regarded the methods which calculate a likelihood score of link formation for a pair of nodes based on the local topological embedding of the considered two nodes with regard to the specified graph patterns. We focused on the following three important aspects: (i) quality of prediction; (ii) efficiency of the involved computations; and (iii) application areas. To achieve our goal, we stated three research questions: (1) How can we predict citation counts for academic publications using link prediction methods? (2) Can we improve temporal link prediction in heterogeneous networks? (3) Which tools can be used to efficiently solve graph pattern matching problems? In the following we summarize the results which were obtained while answering the above-mentioned questions.

7.1.1 Citation Count Prediction

We showed that citation count prediction is a novel application area for link prediction. Firstly, we have demonstrated how to formulate the citation count prediction problem as a link prediction problem. Secondly, by adopting score calculation based on the graph evolution rules of Bringmann et al. [21] we have introduced a new feature GERscore for solving the citation count prediction problem. We have also proposed a new score calculation. The mentioned graph evolution rules are a special type of graph patterns which capture network evolution over time at a microscopic level and are mined from the network by using a graph pattern mining procedure [21]. Thirdly, we have designed an extended evaluation framework which we have applied not only to the new feature, but also to several state-of-the-art features. Our experiments show that the new feature GERscore performs better than ten state-of-the-art features in the classification task. Furthermore, the average accuracy of the classification is not affected much if we bring in other baseline features into the model. In the regression task the new feature outperforms the state-of-the-art features for the dataset of publications from computer science domain (we call this dataset ArnetMiner), though the latter still contribute to the performance of regression models. Thus, the application of graph pattern mining to the citation count prediction problem leads to better results.

However, for the dataset of publications from physics (we call this dataset HepTh) the GERscore is not as good as the author related features, i.e., author rank, MPIA and TPIA, though it does contribute to the increase of the performance. HepTh provides better coverage of papers in the relevant domain, thus the citations are more complete. Another difference of HepTh from ArnetMiner is the domain: physics for the first and computer science for the latter. The last issue is the amount of mined graph evolution rules: we have only 230 unlabeled evolution rules for HepTh. We are not sure which of these differences leads to the disagreement in the best performing features. In the previous work the authors argue that such disagreement arises due to the nature of the relevant scientific domains [80]. However, additional investigation is required to draw a final conclusion.

7.1.2 Heterogeneous Networks

We have studied the performance of several state-of-the-art and five new scores for temporal link prediction in social networks with two types of links. The new scores consider the temporality and heterogeneity of links. By using a Web API provided by the gaming company Valve, we have collected data about the gaming experience and friendship within an online community. Based on the collected data, we have constructed the gaming network Dota2. We have performed experiments on two real-world social networks. The first network is the aforementioned gaming network Dota2 with links indicating team mates and friends respectively. The second network is a collaboration network. It is constructed on the dataset of scientific publications HepTh, where nodes represent authors with colleague and peer links. We have confirmed once again that considering the temporal aspect of links when studying network evolution is important

and leads to an improved link prediction performance. However, our results indicate that the performance of link prediction methods varies over time, and in two cases there is considerable inconsistency in results within the same network type.

We have noticed that the methods which use some variation of a triad counting technique do not perform well on the network with very low average clustering and transitivity coefficients. By using a support measure to estimate the frequencies of graphlets we achieve better performance for temporal link prediction. We observe that some network properties can point out which weighting scheme for 3-node graphlets is more effective for temporal link prediction.

Our experiments illustrated that network evolution cannot be explained by one specific feature at all time points which emphasizes the importance of combining different features into efficient models.

7.1.3 Tools for Graph Pattern Matching

We have studied how four database systems perform with regard to the problem of graph pattern matching. In our study we use a relational database PostgreSQL, a graph database Neo4j, a deductive database Clingo and RDF-based database Jena TDB. The most mature system among these four is PostgreSQL with Neo4j being the youngest. By conducting extensive experiments on synthetic and real-world datasets we come to the following conclusions.

Jena TDB is in general better than the other database systems on our benchmark. However, it turned out to be the most sensitive system to the changes in the data schema. Clingo does not scale with the size of the data graph. Its performance drastically drops when the data graph becomes bigger. Though the performance of Clingo is not much influenced by the size of the pattern graph, it worsens with the growth of the cycles in the pattern graph. Neo4j cannot efficiently handle cyclic pattern graphs. However, it scales very well with regard to the size of the data graph as well as the size of the pattern graph. The performance of PostgreSQL oscillates generally due to the changes in the join order chosen in the execution plan. Though PostgreSQL shows good performance for cyclic patterns, it does not scale well with regard to the size of the pattern graphs and the size of the data graphs.

7.2 Future Work and Open Issues

In this thesis we have shown that by using graph pattern matching we can better capture the formation of links in heterogeneous networks over time and we can also solve new interesting problems with the help of link prediction methods. Nevertheless, there are still many open questions and directions for future work which we outline in this section.

We have performed both classification and regression tasks for the prediction of citation counts in one year. Our results indicate that the performance of the model does not always improve if we include more features. Moreover, we have not included all features from the previous works in our evaluation framework, e.g., content related

features [80, 117, 116] or network related features [80, 101]. Thus, an important aspect to investigate is the performance of various features on different datasets and their optimal combination. Dimension reduction methods might be of help for this task.

One of the ultimate goals of citation count prediction is to construct a recommender system for academic publications. Preliminary work in this direction has been already done by Yan et al. [116]. Future work is to include the new features and to improve the recommendation.

Thorough investigation is required to understand how the mined evolution rules influence the predictive power of the GERscore. Here we want to investigate in several directions. The first issue is to study the influence of input parameters, minimum support (minSup) and maximum size (maxSize), and what is the best combination for them. The advantage of maxSize high and minSupport low is that we will obtain more evolution rules. On the flip side, the computational time will grow exponentially. Another issue is that real-world networks change considerably over time. It may lead to the fact that the evolution rules which are frequent and have high confidence at time t may become rudimentary in ten years and will not be predictive of the citation counts. Thus, we plan to investigate for how long mined evolution rules on average stay predictive. This is an important question also because mining graph evolution rules is computationally hard, and reducing the amount of re-learning GERscores is extremely important.

There is currently a trend to develop supervised models for link prediction by combining classical unsupervised scores (for example, AA, JC, PA, CN) with new features [76, 118]. These models are then tested on a variety of datasets with the goal to fit as many as possible. We noticed that there might be innate network properties (e.g, average clustering coefficient, presence of structural holes) which could point out appropriate features to explain its future evolution.

Instead of designing a general model to fit different networks, we could guide the process of model development with regard to these properties. For example, the methodology based on 3-node graphlets, which we applied in Chapter 5, and more generally the approach of VCP [76], provide considerable flexibility as to how much additional information (besides the local structure among nodes) is captured by graph patterns, like node labels, link labels, frequency estimation and weighting scheme. For interaction-based networks (like gaming) we might take user activeness to categorize nodes while for relationship-based networks (like co-authorship) considering geographical location of users would lead to better performance of link prediction methods. The categorization of nodes according to structural hole spanning could lead to the further improvement in performance of our new scores $tscore_l^{(i)}$. However, to tune the parameter configuration, we need a bigger pool of networks.

Another direction for the future work is the introduction of time series techniques which have been shown to improve the prediction of links over time [30, 118]. Lastly, by efficiently combining the new features we could compare their performance with the state-of-the-art models like HPLP, MRIP and VCP [76, 118]. Here we face an issue with the proper evaluation of link prediction methods which has been recently raised by Yang et al. [119]. The problem is not only which evaluation metrics to use (for example,

AUPR or AUROC) or how to construct training and testing datasets, but also which methods should be the baseline to compare the new methods with. Currently new link prediction methods are being developed each year, and often they are compared against the established neighborhood-based or random walk based methods like AA, CN, JC or PF. To the best of our knowledge, there is no work comparing, for example, algebraic methods with probabilistic models. Lichtenwalter and Chawla have developed one of the first multi-core link prediction platforms which includes some of the most commonly used methods [75]. However, their platform again does not cover any algebraic method or probabilistic model. Hence, there is an opportunity for future work.

In Chapter 4 we used graph evolution rules to solve the problem of temporal link prediction. In contrast, in Chapter 5 we applied graph patterns with three nodes, different link types and node labels. It is unclear which approach better captures the temporal evolution of networks. Additionally, in both cases we could introduce further information to the considered graph patterns. The question is how fine-grained the patterns should be to obtain the optimum results in the temporal link prediction task.

The biggest bottleneck for graph pattern based link prediction is the problem of graph pattern matching. In our opinion, the efficiency of databases for solving graph pattern matching tasks is not yet good enough for information systems that deal with real-world big data scenarios. Graph database systems should consider implementing the state-of-the-art algorithms for this task [67]. Especially query optimizers could profit from the insights of these algorithms since they provide guidelines with regard to the optimal choice of the starting node, and of the join order for the links and nodes. The latter holds also for PostgreSQL. We plan to investigate how the efficiency of the database systems to solve the graph pattern matching task can be increased by influencing the matching order of the links. Furthermore, in all database systems from our benchmark one can change various configurations of the servers to achieve better results (for example, increase cache memory), but it is unclear what is the best configuration if we need to perform a variety of graph queries and not just graph pattern matching. This calls for further investigation and requires the integration of other types of graph queries into our benchmark.

List of Publications

[1] Nataliia Rümmele, Ryutaro Ichise, and Hannes Werthner. Exploring supervised methods for temporal link prediction in heterogeneous social networks. In *Proceedings of the International World Wide Web Conference (Companion Volume)*, WWW (Companion Volume), 2015.

[2] Nataliia Pobiedina and Ryutaro Ichise. Citation count prediction as a link prediction problem. *Journal of Applied Intelligence*, 42(3):1–17, 2015.

[3] Julia Neidhardt, Nataliia Pobiedina, and Hannes Werthner. What can we learn from review data? In *Proceedings of ENTER 2015: Volume 6*, 2015.

[4] Nataliia Pobiedina, Stefan Rümmele, Sebastian Skritek, and Hannes Werthner. Benchmarking database systems for graph pattern matching. In *Proceedings of the 25th International Conference on Database and Expert Systems Applications*, DEXA, pages 226–241, 2014.

[5] Nataliia Pobiedina and Ryutaro Ichise. Predicting citation counts for academic literature using graph pattern mining. In *Proceedings of the 27th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*, IEA/AIE, pages 109–119, 2014.

[6] Nataliia Pobiedina, Julia Neidhardt, Maria del Carmen Calatrava Moreno, Laszlo Grad-Gyenge, and Hannes Werthner. On successful team formation: Statistical analysis of a multiplayer online game. In *Proceedings of the IEEE 15th Conference on Business Informatics*, CBI, pages 55–62, 2013.

[7] Nataliia Pobiedina, Julia Neidhardt, Maria del Carmen Calatrava Moreno, and Hannes Werthner. Ranking factors of team success. In *Proceedings of the 22nd International World Wide Web Conference (Companion Volume)*, WWW (Companion Volume), pages 1185–1194, 2013.

[8] Anna Fensel, Julia Neidhardt, Nataliia Pobiedina, Dieter Fensel, and Hannes Werthner. Towards an intelligent framework to understand and feed the web. In *Proceedings of the Business Information Systems Workshops - International Workshops and Future Internet Symposium*, pages 255–266, 2012.

[9] Nataliia Pobiedina. Modeling the flow and change of information on the web. In *Proceedings of the 21st World Wide Web Conference (Companion Volume)*, WWW (Companion Volume), pages 173–178, 2012.

List of Figures

1.1 Information Systems Research Framework [55]. 5
1.2 Design Science Framework of this thesis. 6
1.3 Classification of our artifacts according to Design Science framework [55] . . 7

2.1 Example of a small network with eight nodes and nine links. 14
2.2 Example of a citation network. 15
2.3 Example of a collaboration network between authors of scientific publications. 16
2.4 Friendship network between members of a karate club. Colors of the nodes correspond to the detected communities. Numbers in the nodes correspond to their ids. 17
2.5 Degree distribution for the karate network. 17
2.6 A connected triple (P_1) and a closed triad (P_2). 18
2.7 Example of a multi-relational network with two types of links: solid and dashed lines represent these types. 19
2.8 Illustration of the link prediction problem in a small friendship network. . . . 20
2.9 Examples data graph G, pattern graph P, and resulting embeddings M_1, M_2. 23

3.1 Number of published papers in Elsevier library which are related to the link prediction problem (taken from [111]). 25
3.2 The general framework for link prediction [111]. 26

4.1 Example of a citation network. 42
4.2 Process flow for predicting citation counts using a modified link prediction method. 46
4.3 Examples of relative time patterns and graph evolution rules. Node labels correspond to the number of authors. 47
4.4 Examples of graph evolution rules mined from HepTh and ArnetMiner datasets using number of authors as node label: (a) rule with highest support for HepTh, (c) rule with highest confidence for HepTh, (b) rule with highest support for ArnetMiner, (d) rule with highest confidence for ArnetMiner. 51
4.5 Correlation between average citation count and features: author rank, venue rank, recency and $GERscore^2_{1,3}$. 54

5.1 3-node graphlets for a network with links of two types. 67

5.2 Example of a network G with two types of links. 68
5.3 The entity relationship diagram of PostgreSQL database which stores data for the gaming network Dota2. 72
5.4 ROC and PR curves for dota2 network. 78

6.1 Examples data graph G, pattern graph P, and resulting embeddings M_1, M_2. 85
6.2 Data schema for PostreSQL and Jena TDB. 88
6.3 Visualization of a network generated via Erdos Renyi model with 1000 nodes. Colors of the nodes correspond to the detected communities in the network. . 93
6.4 Visualization of a network generated via preferential attachment model with 1000 nodes. Colors of the nodes correspond to the detected communities in the network. 94
6.5 Visualization of the terrorist organization collaboration network. Colors of the nodes correspond to the detected communities in the network. 95
6.6 Degree distributions of three constructed networks. 97
6.7 Average run time in seconds on logarithmic scale for ERM. 99
6.8 Average run time in seconds on logarithmic scale for PAM. 101
6.9 Average run time in seconds on log-scale for synthetic patterns on PAM1000. 102
6.10 Average run time in seconds on logarithmic scale for GTON and HepTH. . . 103

List of Tables

3.1 Social theories behind link prediction techniques. 27
3.2 Link prediction techniques. 29

4.1 Results of graph pattern mining. 50
4.2 Distribution of instances according to classes (% Total). 55
4.3 Accuracy (%) and Precision (%) for the different features for the Classification Task in Scenario 1. 55
4.4 Accuracy (%) and Precision (%) for the different features for the Classification Task in Scenario 2. 55
4.5 Accuracy (%) and Precision (%) for the full model with and without the GERscore for the Classification Task in Scenario 1. 56
4.6 Accuracy (%) and Precision (%) for the full model with and without the GERscore for the Classification Task in Scenario 2. 56
4.7 Performance measure R^2 for the different features for the Regression Task in Scenario 1. 58
4.8 Performance measure R^2 for the different features for the Regression Task in Scenario 2. 58
4.9 Performance measure $RMSE$ for the different features for the Regression Task in Scenario 1. 59
4.10 Performance measure $RMSE$ for the different features for the Regression Task in Scenario 2. 59
4.11 Performance measures (R^2 and $RMSE$) for the full model with and without the GERscore for the Regression Task in Scenario 1. 60
4.12 Performance measures (R^2 and $RMSE$) for the full model with and without the GERscore for the Regression Task in Scenario 2. 60

5.1 Properties of Dota2 network over time. 70
5.2 Properties of HepTh network over time. 71
5.3 AUPR for predicting mate links. 75
5.4 AUPR for predicting friend links. 75
5.5 AUPR for predicting peer links. 76
5.6 AUPR for predicting colleague links. 76
5.7 AUROC for predicting mate links. 79

5.8 AUROC for predicting friend links. 79
5.9 AUROC for predicting peer links. 80
5.10 AUROC for predicting colleague links. 80

6.1 Summary of the datasets. 92

Bibliography

[1] Sisay Fissaha Adafre and Maarten de Rijke. Discovering missing links in Wikipedia. In *Proceedings of the 3rd International Workshop on Link Discovery*, LinkKDD, pages 90–97, 2005.

[2] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[3] Santa Agreste, Pasquale De Meo, Emilio Ferrara, Sebastiano Piccolo, and Alessandro Provetti. Analysis of a heterogeneous social network of humans and cultural objects. *IEEE T. Systems, Man, and Cybernetics: Systems*, 45(4):559–570, 2015.

[4] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, 9:1981–2014, 2008.

[5] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[6] Mario Alviano, Francesco Calimeri, Günther Charwat, Minh Dao-Tran, Carmine Dodaro, Giovambattista Ianni, Thomas Krennwallner, Martin Kronegger, Johannes Oetsch, Andreas Pfandler, Jörg Pührer, Christoph Redl, Francesco Ricca, Patrik Schneider, Martin Schwengerer, LaraKatharina Spendier, JohannesPeter Wallner, and Guohui Xiao. The fourth answer set programming competition: Preliminary report. In *Proceedings of the 12th International Conference on Logic Programming and Nonmonotonic Reasoning*, LPNMR, pages 42–53, 2013.

[7] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Effects of user similarity in social media. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, WSDM, pages 703–712, 2012.

[8] Renzo Angles. A comparison of current graph database models. In *Workshops Proceedings of the IEEE 28th International Conference on Data Engineering, ICDE*, pages 171–177, 2012.

[9] Renzo Angles, Arnau Prat-Pérez, David Dominguez-Sal, and Josep-Lluis Larriba-Pey. Benchmarking database systems for social network applications. In *First International Workshop on Graph Data Management Experiences and Systems, GRADES, co-located with SIGMOD/PODS*, page 15, 2013.

[10] Yudistira Asnar, Elda Paja, and John Mylopoulos. Modeling design patterns with description logics: A case study. In *Proceedings of the 23rd International Conference on Advanced Information Systems Engineering*, CAiSE, pages 169–183, 2011.

[11] Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna. Four degrees of separation. In *Proceedings of the 3rd Annual ACM Web Science Conference*, WebSci, pages 33–42, 2012.

[12] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science Magazine*, 286(5439):509–512, 1999.

[13] Albert-Laszlo Barabasi, Reka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(4):69 – 77, 2000.

[14] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the Third International Conference on Weblogs and Social Media*, ICWSM, 2009.

[15] J. Beel and B. Gipp. Google scholar's ranking algorithm: The impact of citation counts (an empirical study). In *Proceedings of the Third IEEE International Conference on Research Challenges in Information Science*, RCIS, pages 439–446, 2009.

[16] Steven Bethard and Dan Jurafsky. Who should I cite: learning literature search models from citation behavior. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*, CIKM, pages 609–618, 2010.

[17] Prantik Bhattacharyya, Ankush Garg, and ShyhtsunFelix Wu. Analysis of user keyword similarity in online social networks. *Social Network Analysis and Mining*, 1(3):143–158, 2011.

[18] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[19] Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[20] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998.

[21] Bjoern Bringmann, Michele Berlingerio, Francesco Bonchi, and Arisitdes Gionis. Learning and predicting the evolution of social networks. *IEEE Intelligent Systems*, 25:26–35, 2010.

[22] Björn Bringmann and Siegfried Nijssen. What is frequent in a single graph? In *Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining*, PAKDD, pages 858–863, 2008.

[23] R.S. Burt. *Structural holes: The social structure of competition.* Harvard University Press, Cambridge, MA, 1992.

[24] M. Callaham, R.L. Wears, and E. Weber. Journal prestige, publication bias, and other characteristics associated with citation of published studies in peer-reviewed journals. *JAMA: the journal of the American Medical Association*, 287(21):2847–2850, 2002.

[25] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):1–27, May 2011.

[26] Li Chen, Wei Zeng, and Quan Yuan. A unified framework for recommending items, groups and friends in social media environment via mutual resource fusion. *Expert Syst. Appl.*, 40(8):2889–2903, 2013.

[27] Fan Chung and Wenbo Zhao. Pagerank and random walks on graphs. In *Fete of Combinatorics and Computer Science*, volume 20 of *Bolyai Society Mathematical Studies*, pages 43–62. Springer Berlin Heidelberg, 2010.

[28] Aaron Clauset, Cristopher Moore, and Mark E.J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2008.

[29] David Crandall, Dan Cosley, Daniel Huttenlocher, Jon Kleinberg, and Siddharth Suri. Feedback effects between similarity and social influence in online communities. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 160–168, 2008.

[30] P.R. da Silva Soares and R. Bastos Cavalcante Prudencio. Time series based link prediction. In *Proceedings of the International Joint Conference on Neural Networks*, (IJCNN), pages 1–7, 2012.

[31] Darcy Davis, Ryan Lichtenwalter, and Nitesh V. Chawla. Supervised methods for multi-relational link prediction. *Social Network Analysis and Mining*, 3(2):127–141, 2013.

[32] Jesse Davis and Mark Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML, pages 233–240, 2006.

[33] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer, 1996.

[34] Fereshteh Didegah and Mike Thelwall. Determinants of research citation impact in nanoscience and nanotechnology. *JASIST (JASIS)*, 64(5):1055–1064, 2013.

[35] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[36] David Dominguez-Sal, P. Urbón-Bayes, Aleix Giménez-Vañó, Sergio Gómez-Villamor, Norbert Martínez-Bazan, and Josep-Lluis Larriba-Pey. Survey of graph database performance on the HPC scalable graph analysis benchmark. In *Web-Age Information Management - WAIM International Workshops: IWGD, XMLDM, WCMT*, pages 37–48, 2010.

[37] Yuxiao Dong, Jie Tang, Sen Wu, Jilei Tian, N.V. Chawla, Jinghai Rao, and Huanhuan Cao. Link prediction and recommendation across heterogeneous social networks. In *Proceedings of the IEEE 12th International Conference on Data Mining*, ICDM, pages 181–190, Dec 2012.

[38] Beyza Ermis, Evrim Acar, and A.Taylan Cemgil. Link prediction in heterogeneous data via generalized coupled tensor factorization. *Data Mining and Knowledge Discovery*, 29(1):203–236, 2015.

[39] Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Yinghui Wu. Adding regular expressions to graph reachability and pattern queries. *Frontiers of Computer Science*, 6(3):313–338, 2012.

[40] Anna Fensel, Julia Neidhardt, Nataliia Pobiedina, Dieter Fensel, and Hannes Werthner. Towards an intelligent framework to understand and feed the web. In *Proceedings of the Business Information Systems Workshops - International Workshops and Future Internet Symposium*, pages 255–266, 2012.

[41] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010.

[42] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. on Knowl. and Data Eng.*, 19(3):355–369, 2007.

[43] L.C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1:215–239, 1979.

[44] Valerio Freschi. A graph-based semi-supervised algorithm for protein function prediction from interaction maps. In *Learning and Intelligent Optimization*, pages 249–258, 2009.

[45] Brian Gallagher. Matching structure and semantics: A survey on graph-based pattern matching. In *Proceedings of the AAAI Fall Symposium on Capturing and Using Patterns for Evidence Detection*, 2006.

[46] Eugene Garfield. Impact factors, and they won't go away. *Science*, 411(6837):522–522, 2001.

[47] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Thomas Schneider. Potassco: The Potsdam answer set solving collection. *AI Commun.*, 24(2):107–124, 2011.

[48] Johannes Gehrke, Paul Ginsparg, and Jon Kleinberg. Overview of the 2003 KDD cup. *SIGKDD Explorations*, 5(2):149–151, December 2003.

[49] Lisa Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of link structure. *J. Mach. Learn. Res.*, 3:679–707, March 2003.

[50] Lise Getoor and Christopher P. Diehl. Link mining: A survey. *SIGKDD Explor. Newsl.*, 7(2):3–12, 2005.

[51] M. Girvan and Mark E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[52] M. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973.

[53] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *Proceedings of the SDM workshop on Link Analysis, Counterterrorism and Security*, 2006.

[54] Mohammad Al Hasan and Mohammed J. Zaki. A survey of link prediction in social networks. In Charu C. Aggarwal, editor, *Social Network Data Analytics*, pages 243–275, 2011.

[55] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–106, 2004.

[56] J.E. Hirsch. An index to quantify an individual's scientific research output. In *Proc. the National Academy of Sciences of the United States America*, page 102(46):16569, 2005.

[57] Florian Holzschuher and René Peinl. Performance of graph query languages: comparison of Cypher, Gremlin and native access in Neo4j. In *Joint EDBT/ICDT Conferences, Workshop Proceedings*, pages 195–204, 2013.

[58] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Unbiased recursive partitioning: A conditional inference framework. *J. Comp. Graph. Stat.*, 15(3):651–674, 2006.

[59] Zan Huang, Xin Li, and Hsinchun Chen. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL, pages 141–142, 2005.

[60] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[61] H. Kashima and N. Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *Proceedings of the 6th International Conference on Data Mining*, ICDM, pages 340–349, 2006.

[62] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[63] Przemyslaw Kazienko, Katarzyna Musial, and Tomasz Kajdanowicz. Multidimensional social network in the social recommender system. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 41(4):746–759, 2011.

[64] Abhaya V. Kulkarni, Jason W. Busse, and Iffat Shams. Characteristics associated with citation rate of the medical literature. *PLOS one*, 2(5), 2007.

[65] Jérôme Kunegis and Andreas Lommatzsch. Learning spectral graph transformations for link prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML, pages 561–568, 2009.

[66] P. Lazarsfeld and R.K. Merton. Friendship as a social process: A substantive and methodological analysis. *Freedom and Control in Modern Society*, pages 18–66, 1954.

[67] Jinsoo Lee, Wook-Shin Han, Romans Kasperovics, and Jeong-Hoon Lee. An in-depth comparison of subgraph isomorphism algorithms in graph databases. *PVLDB*, 6(2):133–144, 2012.

[68] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.

[69] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. In *Proceedings of the 7th ACM conference on Electronic commerce*, EC, pages 228–237, 2006.

[70] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 497–506, 2009.

[71] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD, pages 462–470, 2008.

[72] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM TKDD*, 1(1), 2007.

[73] Xin Li and Hsinchun Chen. Recommendation as link prediction: A graph kernel-based machine learning approach. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL, pages 213–216, 2009.

[74] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[75] Ryan N. Lichtenwalter and Nitesh V. Chawla. LPmade: Link prediction made easy. *J. Mach. Learn. Res.*, 12:2489–2492, 2011.

[76] Ryan N. Lichtenwalter and Nitesh V. Chawla. Vertex collocation profiles: Subgraph counting for link analysis and prediction. In *Proceedings of the 21st World Wide Web Conference*, WWW, pages 1019–1028, 2012.

[77] Ryan N. Lichtenwalter, Jake T. Lussier, and Nitesh V. Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 243–252, 2010.

[78] Haifeng Liu, Zheng Hu, Hamed Haddadi, and Hui Tian. Hidden link prediction based on node centrality and weak ties. *EPL (Europhysics Letters)*, 101(1):18004, 2013.

[79] Yan Liu and Zhenzhen Kou. Predicting who rated what in large-scale datasets. *SIGKDD Explorations*, 9(2):62–65, 2007.

[80] Avishay Livne, Eytan Adar, Jaime Teevan, and Susan Dumais. Predicting citation counts using text and graph mining. In *The iConference Workshop on Computational Scientometrics: Theory and Applications*, 2013.

[81] Linyuan Lü and Tao Zhou. Link prediction in weighted networks: The role of weak ties. *EPL (Europhysics Letters)*, 89(1):18001, 2010.

[82] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150 – 1170, 2011.

[83] Amy Mcgovern, Lisa Friedl, Michael Hay, Brian Gallagher, Andrew Fast, Jennifer Neville, and David Jensen. Exploiting relational structure to understand publication patterns in high-energy physics. *SIGKDD Explorations*, 5:2003, 2003.

[84] M. McPherson, L. Smith-Lovin, and J.M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27:415–444, 2001.

[85] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6912 of *Lecture Notes in Computer Science*, pages 437–452. Springer Berlin Heidelberg, 2011.

[86] Lankeshwara Munasinghe and Ryutaro Ichise. Time score: A new feature for link prediction in social networks. *IEICE Trans.*, 95-D(3):821–828, 2012.

[87] Julia Neidhardt, Nataliia Pobiedina, and Hannes Werthner. What can we learn from review data? In *ENTER 2015: Volume 6*. e-Review of Tourism Research, 2015.

[88] Mark E.J. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E 64*, 2:025102, 2001.

[89] Mark E.J. Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[90] Gergely Palla, Albert-Laszlo Barabasi, and Tamas Vicsek. Quantifying social group evolution. *Nature*, 446:664–667, 2007.

[91] Nataliia Pobiedina. Modeling the flow and change of information on the web. In *Proceedings of the 21st World Wide Web Conference (Companion Volume)*, WWW (Companion Volume), pages 173–178, 2012.

[92] Nataliia Pobiedina and Ryutaro Ichise. Predicting citation counts for academic literature using graph pattern mining. In *Proceedings of the 27th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*, IEA/AIE, pages 109–119, 2014.

[93] Nataliia Pobiedina and Ryutaro Ichise. Citation count prediction as a link prediction problem. *Journal of Applied Intelligence*, 42(3):1–17, 2015.

[94] Nataliia Pobiedina, Julia Neidhardt, Maria del Carmen Calatrava Moreno, Laszlo Grad-Gyenge, and Hannes Werthner. On successful team formation: Statistical analysis of a multiplayer online game. In *Proceedings of the IEEE 15th Conference on Business Informatics*, CBI, pages 55–62, 2013.

[95] Nataliia Pobiedina, Julia Neidhardt, Maria del Carmen Calatrava Moreno, and Hannes Werthner. Ranking factors of team success. In *Proceedings of the 22nd International World Wide Web Conference (Companion Volume)*, WWW (Companion Volume), pages 1185–1194, 2013.

[96] Nataliia Pobiedina, Stefan Rümmele, Sebastian Skritek, and Hannes Werthner. Benchmarking database systems for graph pattern matching. In *Proceedings of the 25th International Conference on Database and Expert Systems Applications*, DEXA, pages 226–241, 2014.

[97] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. `http://www.r-project.org/`.

[98] Nataliia Rümmele, Ryutaro Ichise, and Hannes Werthner. Exploring supervised methods for temporal link prediction in heterogeneous social networks. In *Proceedings of the International World Wide Web Conference (Companion Volume)*, WWW (Companion Volume), 2015.

[99] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[100] Rossano Schifanella, Alain Barrat, Ciro Cattuto, Benjamin Markines, and Filippo Menczer. Folks in folksonomies: social link prediction from shared metadata. In *Proceedings of the 3rd International Conference on Web Search and Web Data Mining*, WSDM, pages 271–280, 2010.

[101] Xiaolin Shi, Jure Leskovec, and Daniel A. McFarland. Citing for high impact. In *Proceedings of the 10th annual joint conference on Digital libraries*, JCDL, pages 49–58, 2010.

[102] Neil J. Smelser. *Problematics of Sociology*. The Georg Simmel Lectures, 1995.

[103] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427 – 437, 2009.

[104] Tommi Syrjänen and Ilkka Niemelä. The Smodels system. In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning*, LPNMR, pages 434–438, 2001.

[105] Jie Tang, Tiancheng Lou, and Jon Kleinberg. Inferring social ties across heterogenous networks. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM, pages 743–752, 2012.

[106] Ben Taskar, Ming fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *in Neural Information Processing Systems*, 2003.

[107] Bogdan George Tudorica and Cristian Bucur. A comparison between several NoSQL databases with comments and notes. In *Proceedings of the Roedunet International Conference*, pages 1–5, 2011.

[108] Jorge Valverde-Rebaza and Alneu de Andrade Lopes. Exploiting behaviors of communities of twitter users for link prediction. *Social Network Analysis and Mining*, 3(4):1063–1074, 2013.

[109] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th Annual Southeast Regional Conference*, page 42, 2010.

[110] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. In *Proceedings of the 7th IEEE International Conference on Data Mining*, ICDM, pages 322–331, 2007.

[111] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.

[112] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *SIGKDD Explorations*, 5(1):59–68, 2003.

[113] D. Watts and S. Stogatz. Small world. *Nature*, 393:440–442, 1998.

[114] Evan W. Xiang. A Survey on Link Prediction Models for Social Network Data. Technical report, The Hong Kong University of Science and Technology, 2008.

[115] Jennifer Xu and Hsinchun Chen. Criminal network analysis and visualization. *Commun. ACM*, 48(6):100–107, 2005.

[116] Rui Yan, Congrui Huang, Jie Tang, Yan Zhang, and Xiaoming Li. To better stand on the shoulder of giants. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, JCDL, pages 51–60, 2012.

[117] Rui Yan, Jie Tang, Xiaobing Liu, Dongdong Shan, and Xiaoming Li. Citation count prediction: learning to estimate future citations for literature. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM, pages 1247–1252, 2011.

[118] Yang Yang, Nitesh V. Chawla, Yizhou Sun, and Jiawei Han. Predicting links in multi-relational and heterogenous networks. In *Proceedings of the 12th IEEE International Conference on Data Mining*, ICDM, 2012.

[119] Yang Yang, Ryan N. Lichtenwalter, and Nitesh V. Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, pages 1–32, 2014.

[120] W.W. Zachary. An information flow model for conflict and fission in small groups. *J. Anthropol. Res.*, 33(4):452–473, 1977.

[121] Jianhan Zhu, Jun Hong, and John G. Hughes. Using markov models for web site link prediction. In *Proceedings of the 13th ACM Conference on Hypertext and Hypermedia*, HYPERTEXT, pages 169–170, 2002.

[122] Lei Zou, Lei Chen, M. Tamer Özsu, and Dongyan Zhao. Answering pattern match queries in large graph databases via graph embedding. *VLDB J.*, 21(1):97–120, 2012.

Acronyms

AA Adamic-Adar score. 31, 37, 108, 109

AUPR Area Under Precision-Recall curve. 22, 74, 109

AUROC Area Under Receiver Operating Characteristic curve. 21, 74, 109

CART Classification and Regression Tree. 51

CIT Conditional Inference Trees. 51, 74

CN Common Neighbors score. 30, 31, 37, 108, 109

GER Graph Evolution Rules. 32

GERM Graph Evolution Rule Miner. 32

HPLP High Performance Link Prediction model. 33, 108

HT Hitting Time. 31

JC Jaccard's coefficient. 30, 31, 37, 108, 109

LR Logistic Regression. 74

mLR Multinomial Logistic Regression. 50

MPIA Maximum Past Influence for Authors. 52, 106

MPIV Maximum Past Influence for Venue. 52

MRF Markov Random Field. 34

MRIP Multi-Relational Influence Propagation model. 37, 108

MRLP Multi-Relational Link Prediction model. 33

mSVM Multi-class Support Vector Machines. 51

PA Preferential Attachment score. 30, 37, 108

PF PropFlow. 32, 109

PPI Protein-Protein Interaction. 38

PR-curve Precision-Recall curve. 22

PRM Probabilistic Relational Model. 35

RF Random Forests. 74

RMSE Root of Mean Squared Error. 57

ROC curve Receiver Operating Characteristic curve. 21

RPR Rooted PageRank. 32

SVR Support Vector Regression. 51

TNR True Negative Rate. 21

TPIA Total Past Influence for Authors. 52, 106

TPIV Total Past Influence for Venue. 52

TPR True Positive Rate. 21

TS Time Score. 36

VCP Vertex Collocation Profile. 33, 37, 108

WWW World Wide Web. 15