On the Spectral Evolution of Large Networks

Jérôme Kunegis

Institute for Web Science and Technologies
University of Koblenz-Landau
kunegis@uni-koblenz.de

November 2011

Vom Promotionsausschuss des Fachbereichs 4: Informatik der Universität Koblenz-Landau zur Verleihung des akademischen Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation.

PhD thesis at the University of Koblenz-Landau.

Datum der wissenschaftlichen Aussprache:	9. November 2011
Vorsitz des Promotionsausschusses:	Prof. Dr. Karin Harbusch
Berichterstatter:	Prof. Dr. Steffen Staab
Berichterstatter:	Prof. Dr. Christian Bauckhage
Berichterstatter:	Prof. Dr. Klaus Obermayer

Abstract

In this thesis, I study the spectral characteristics of large dynamic networks and formulate the spectral evolution model. The spectral evolution model applies to networks that evolve over time, and describes their spectral decompositions such as the eigenvalue and singular value decomposition. The spectral evolution model states that over time, the eigenvalues of a network change while its eigenvectors stay approximately constant.

I validate the spectral evolution model empirically on over a hundred network datasets, and theoretically by showing that it generalizes a certain number of known link prediction functions, including graph kernels, path counting methods, rank reduction and triangle closing. The collection of datasets I use contains 118 distinct network datasets. One dataset, the signed social network of the Slashdot Zoo, was specifically extracted during work on this thesis. I also show that the spectral evolution model can be understood as a generalization of the preferential attachment model, if we consider growth in latent dimensions of a network individually.

As applications of the spectral evolution model, I introduce two new link prediction algorithms that can be used for recommender systems, search engines, collaborative filtering, rating prediction, link sign prediction and more. The first link prediction algorithm reduces to a one-dimensional curve fitting problem from which a spectral transformation is learned. The second method uses extrapolation of eigenvalues to predict future eigenvalues.

As special cases, I show that the spectral evolution model applies to directed, undirected, weighted, unweighted, signed and bipartite networks. For signed graphs, I introduce new applications of the Laplacian matrix for graph drawing, spectral clustering, and describe new Laplacian graph kernels. I also define the algebraic conflict, a measure of the conflict present in a signed graph based on the signed graph Laplacian. I describe the problem of link sign prediction spectrally, and introduce the signed resistance distance. For bipartite and directed graphs, I introduce the hyperbolic sine and odd Neumann kernels, which generalize the exponential and Neumann kernels for undirected unipartite graphs. I show that the problem of directed and bipartite link prediction are related by the fact that both can be solved by considering spectral evolution in the singular value decomposition.

Zusammenfassung

In dieser Doktorarbeit beschreibe ich das spektrale Verhalten von großen, dynamischen Netzwerken und formuliere das spektrale Evolutionsmodell. Das spektrale Evolutionsmodell beschreibt das Wachstum von Netzwerken, die sich im Laufe der Zeit ändern, und charakterisiert ihre Eigenwert- und Singulärwertzerlegung. Das spektrale Evolutionsmodell sagt aus, dass im Laufe der Zeit die Eigenwerte eines Netzwerks wachsen, und die Eigenvektoren nahezu konstant bleiben.

Ich validiere das spektrale Evolutionsmodell empirisch mit Hilfe von über einhundert Netzwerkdatensätzen, und theoretisch indem ich zeige, dass es eine gewisse Anzahl von bekannten Algorithmen zur Kantenvorhersage verallgemeinert, darunter Graph-Kernel, Pfad-Zähl-Methoden, Rangreduktion und Triangle-Closing. Die Sammlung von Datensätzen, die ich verwende enthält 118 distinkte Datensätze. Ein Datensatz, das soziale Netzwerk mit negativen Kanten des Slashdot-Zoo, wurde speziell während des Verfassens dieser Arbeit extrahiert. Ich zeige auch, dass das spektrale Evolutionsmodell als Generalisierung des Preferential-Attachment-Modells verstanden werden kann, wenn Wachstum in latenten Dimensionen einzeln betrachtet wird.

Als Anwendungen des spektralen Evolutionsmodells führe ich zwei neue Algorithmen zur Kantenvorhersage ein, die in Empfehlungssystemen, Suchmaschinen, im Collaborative-Filtering, für die Vorhersage von Bewertungen, für die Vorhersage von Kantenvorzeichen und mehr verwendet werden können. Der erste Kantenvorhersagealgorithmus ergibt ein eindimensionales Curve-Fitting-Problem, aus dem eine spektrale Transformation gelernt wird. Die zweite Methode verwendet Extrapolation von Eigenwerten, um zukünftige Eigenwerte vorherzusagen.

Als Spezialfälle zeige ich, dass das spektrale Evolutionsmodell auf gerichtete, ungerichtete, gewichtete, ungewichtete, vorzeichenbehaftete und bipartite Graphen erweitert werden kann. Für vorzeichenbehaftete Graphen führe ich neue Anwendungen der Laplace-Matrix zur Graphzeichnung, zur spektralen Clusteranalyse, und beschreibe neue Laplace-Graph-Kernel, die auf vorzeichenbehaftete Graphen angewendet werden können. Ich definiere dazu den algebraischen Konflikt, ein Maß für den Konflikt, der in einem vorzeichenbehafteten Graphen vorhanden ist, und das auf der vorzeichenbehafteten Laplace-Matrix begründet ist. Ich beschreibe das Problem der Vorhersage von Kantenvorzeichen spektral, und führe die vorzeichenbehaftete Widerstands-Distanz ein. Für bipartite und gerichtete Graphen führe ich den Sinus-Hyperbolicus- und ungeraden Neumann-Kernel ein, welche den Exponential- und den Neumann-Kernel für ungerichtete unipartite Graphen verallgemeinern. Ich zeige zudem, dass das Problem der gerichteten und bipartiten Kantenvorhersage verwandt sind, dadurch dass beide durch die Evolution der Singulärwertzerlegung gelöst werden können.

Acknowledgments

I could not have written this PhD thesis without help and support from many people.

I thank Narcisse Noubissi Noukumo who got me started with collaborative filtering at the DAI Lab. Martin Mehlitz and Dragan Milošević have helped me getting started with writing papers in the first months as a PhD student. Stephan Schmidt was my collaborator on all things related to signed graph theory. Andreas Lommatzsch helped me during the time where I got my first papers published at conferences. I thank you all for getting me interested in science. I wouldn't even have started writing this thesis if it weren't for your encouragement.

Christian Bauckhage helped me choosing the right topics of research early on in my PhD program, and generally encouraged me throughout my work. Robert Wetzker and Nicolas Neubauer were influential in my study of folksonomies and other multipartite networks. Stephan Spiegel has helped me make the most out of my papers. Damien Fay pushed my work in a statistically grounded direction, and was influential in everything related to the graph Laplacian and network evolution.

I thank Stephan Schmidt, Jan Clausen, Antje Schultz, Thomas Gottron, René Pickhardt and Julia Preusse for their mathematical insight into all areas of graph theory, linear algebra and spectral analysis that I encountered during my PhD program. You're the reason that this thesis is based, I hope, on solid mathematical foundations.

Till Plumbaum, Winfried Umbrath, Torsten Schmidt and Barış Karataş helped me get a more application-oriented approach to machine learning problems. Alan Said and Ernesto William De Luca have taught me how to connect with other scientists. Jürgen Lerner helped me understand how visualization of datasets is tightly connected to link prediction. Your perspective has made it possible for me to disseminate my work to a larger audience at conferences and other events. I thank Christian Banik, Stephan Spiegel and Julia Preusse for choosing to write their diploma theses under my supervision, validating my work and applying it to new datasets and applications.

Klaus Obermayer has helped me keeping on with my work and encouraged me to finish fast. Steffen Staab has given me the opportunity, time and position to follow my vision and complete this thesis. I thank Sebastian Stober and Wolfgang Nejdl for allowing me to give talks related to my dissertation. I thank Sergej Sizov for support in the latter stages in my research.

I explicitely thank my coauthors Dragan Milošević, Stephan Schmidt, Martin Mehlitz, Christian Bauckhage, Andreas Lommatzsch, Stephan Spiegel, Alan Said, Ernesto William De Luca, Damien Fay, Jürgen Lerner, Till Plumbaum, Winfried Umbrath, Barış Karataş, Torsten Schmidt, Arifah Che Alhadi, Nasir Naveed, Maciek Janik, Thomas Gottron, Ansgar Scherp and Steffen Staab. Without your cooperation and patience for writing papers, many parts of this thesis would not have been possible.

I thank everyone who has accepted to review or proofread this thesis: Jan Clausen, Damien

Fay, Klaas Dellschaft, Antje Schultz, Renata Dividino, René Pickhardt, Andreas Lommatzsch, Tina Walber, Julia Preusse, Gerd Gröner, Christoph Ringelstein, Alexander Korth and Tobias Walter. Your input was valuable and has contributed directly to the quality of this thesis.

I thank John Shawe-Taylor, Thomas Zaslavsky and Robert West for helpful input on the Neumann kernel, on the algebraic theory of signed graphs and on estimating power-law exponents. I thank Rabeeh Abbasi, Christian Hachenberg, Felix Schwagereit, Thomas Franz, Olaf Görlitz, Fernando Silva Parreiras, Stefan Scheglmann and Christoph Kling for practical help and technical discussions at the WeST Institute.

I thank everyone who has released network datasets to the public, in particular Jure Leskovec, Alan Mislove, Chris Bizer, Munmun De Choudhury, Christian Thurau, Andreas Hotho, Alan Said, Robert Wetzker, Nicolas Neubauer, Pan Hui, Eiko Yoneki and the Wikimedia Foundation. The data mining community needs more people like you.

The research leading to these results has received funding from the European Community's Seventh Frame Programme under grant agreement n^{o} 215453, WeKnowIt and under grant agreement n^{o} 257859, ROBUST, as well as from the German Research Foundation (DFG) under the MULTIPLA project (grant 38457858).

I thank everyone else that I have not mentioned and that have talked to, interacted with and that have listened to me at conferences, workshops and seminars.

Contents

Chapter 1

Introduction

Networks are everywhere. Whenever we look at the interactions between things, a network is formed implicitly. In the areas of data mining, machine learning, information retrieval and others, networks are modeled as *graphs*. Many, if not most problem types apply to graphs, such as clustering, classification, prediction and pattern recognition. Networks arise in almost all areas of research, commerce and daily life: social networks, road networks, communication networks, trust networks, hyperlink networks, chemical interaction networks, neural networks, collaboration networks and lexical networks. The content of books is routinely modeled as book–word networks, taste as person–item networks and interaction as person–object networks. Databases, the data storage method *par excellence*, are based on the relational model, where relations essentially correspond to edges in a network of entities, resulting in arbitrary networks. Accordingly, a growing number of applications in the areas of data mining, machine learning and information retrieval make use of data in the form of networks. Social networks, the Web, scientific citation graphs, user ratings graphs, trust networks and communication networks are all examples of dataset types that can be represented as networks. In all these cases, information is represented as nodes connected by edges, and it is the structure of the network itself that is of interest, as opposed to data attached to each node.

Mathematically, networks are modeled as graphs. A graph is a mathematical structure that consists of nodes called *vertices* and links called *edges*. Networks from many different areas can be modeled as graphs. For instance, vertices can represent persons, websites, scientific publication or words, and edges can represent friendship between persons, links between websites, citations between scientific publications or lexical relationships between words. The mathematical theory studying graphs is called *graph theory*. An important branch of graph theory is spectral graph theory, in which graphs are analysed using methods from linear algebra. For instance, a social network of n persons can be represented as a matrix of size $n \times n$, where an entry is one when two persons are friends of each other and zero otherwise. This matrix is called the adjacency matrix of the social network and contains all structural information about the network, but it does not contain any additional information such as person names. However, this matrix can be used to reveal structural properties of the social network. For instance, the eigenvectors and eigenvalues of this matrix can be used to group people in the network into coherent communities.

Many applications that make use of networks can be described as the prediction of links. For instance, recommending friends in a social network, predicting new hyperlinks on the Web, predicting movie ratings for users and predicting future email traffic can all be described as problems of link prediction. In a broad sense, the link prediction problem consists of predicting the location or weight of future edges in a network, given a known network. Link prediction in this general sense is used in almost all recommender systems, filtering systems and search

engines. Variants of the link prediction problem are the prediction of new links in a network, the prediction of ratings and the prediction of the meaning of edges, for instance to learn whether two persons are going to be friends or enemies.

In this thesis, spectral graph theory is used to predict which nodes of a network will be connected in the future. This problem is called the *link prediction problem*, and is known under many different names depending on the network to which it is applied. For instance, finding new friends in a social network, recommending movies and predicting which scientists will publish a paper together in the future are all instances of the link prediction problem. Several link prediction algorithms based on spectral graph theory are already known in the literature. However there is no general theory that predicts which spectral link prediction algorithms work best, and under what circumstances certain spectral link prediction algorithms work better than others. To solve these problems, this thesis proposes the *spectral evolution model*: A model that describes in detail how networks change over time in terms of their eigenvectors and eigenvalues. By observing certain growth patterns in actual networks, we are able to predict the growth of networks accurately, and thus can implement relevant recommender systems for all types of network datasets.

1.1 Motivation: Recommender Systems

As an example, consider a social network. A social network consists of users and their connections. These connections between users may represent friendship, trust, or simply an interaction such as a sent email. A social network can be represented as a network in which nodes are users and links represent pairwise relations between them. A small sample social network is shown in Figure 1.1. This small network contains six persons and six friendship links between them. Let us now consider one specific user, the one labeled 1 in the network. This user has two friends: 2 and 4. While this user has only two friends in the network, the actual person behind user 1 may in fact know more people in real life. Therefore, a common task in social networks consists recommending new friends to users. In our example, user 1 may in fact know the users 3 and 5, who are already friends of 1's friends. Since we know, in general, only the existence of links between users, a recommender system as described here must base its recommendation on the structural properties of the network, and cannot use other features or properties attached to individual nodes in the graph.

As a result, a good recommender system must predict which links exist between nodes. This problem is called *link prediction*, and will be the main problem studied in this thesis for which new solutions are introduced. In a broader context, the problem of link prediction can be used to model many different applications:

- Social recommender systems can be implemented as the prediction of links between a single user and other items.
- In the case of collaborative filtering, the task is to predict the weight of links between users and items.
- Finding related documents can be seen as predicting links from one document to another, given a set of links between documents and their topics or contained words.
- Finding related work can be modeled as link prediction between scientific publications connected by citations.
- Similar authors in a bibliographic network can be found by predicting links between authors.

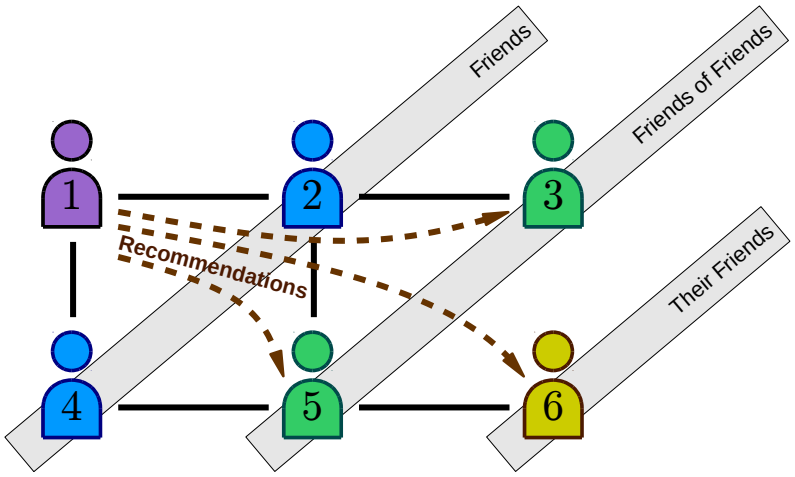


Figure 1.1: A social network of six persons $(1, 2, 3, 4, 5, 6)$ and their relationships (solid lines). The persons 2, 3, 4, 5 and 6 can be partitioned into friends of 1, friends-of-friends of 1, and further friends. A recommendation can be represented as a new edge between 1 and other persons that are not direct friends (dashed lines). The prediction of these kinds of edges is solved by recommender systems, and is one of the applications to which the methods presented in this thesis can be applied.

- Biological interactions between molecule types can be predicted using the known metabolic network.

These applications suggest the study of link prediction algorithms that apply to any type of network. As we will see, an unweighted network can be interpreted as a network with edge weights zero or one, and the link prediction problem then becomes the problem of predicting this value of zero or one for a given node pair.

1.2 Background: Eigenvalue Decomposition

How can networks be analysed? On one hand, the global structure of networks has to be taken into account to characterize the network itself and to compare it with other networks. On the other hand, local structure is needed for learning at the node and link level, such as when defining node centralities or link prediction functions. As shown in this thesis, both levels can be studied separately by considering the *eigenvalue decomposition*. The eigenvalue decomposition is a mathematical construction that can be uniquely computed. When the eigenvalue decomposition is applied to a network, the result consists of eigenvectors and eigenvalues. Decomposing a network into eigenvalues and eigenvectors, both levels can be analysed separately: The eigenvalues, which are also called the spectrum of the network, contain global information about the network, and the eigenvectors contain local information. This separation can then be exploited to predict network growth. As we will show, this is possible because in practice, only the eigenvalues of a network change over time, and the eigenvectors stay constant. By extrapolating the change of eigenvalues into the future, the new eigenvalues can be recombined with the known eigenvectors to give the predicted future network, resulting in new links.

These two observations are the basis of the rest of this thesis: to analyse networks spectrally. The result of this work can therefore be used in all applications that need to make *predictions* in *networks*.

1.3 Thesis: Spectral Evolution

The main result presented in this thesis is the *spectral evolution model* of network growth. The spectral evolution model describes the change in a network in terms of the eigenvalue decomposition of its adjacency matrix. In order to talk about change in a network, we need to know the state of the network at different times. Since in this work we are only interested in the network structure, we only need to know the set of links in function of time. In all network datasets we encountered, this can be derived by knowing the creation time of each link, although it would be enough to know the creation order of edges. The spectral evolution model described in this thesis can be stated as follows:

- During the process of network evolution, the eigenvalues increase, while the eigenvectors stay approximately constant. This observation is possible in networks where edge creation times are known.

These observations can be made on many real-world networks. In these networks, knowing that the eigenvectors stay constant allows us to derive link prediction algorithms that compute the eigenvalue decomposition of a given network, and learn a new set of eigenvalues. By reusing the known eigenvectors, the new eigenvalues give link prediction scores for all missing edges in the network, and can be used to implement recommender systems in these networks. As we will show, these link prediction methods also work when it is not known at what time an edge is added, resulting in link prediction algorithms that apply to any kind of network.

1.4 Application: Link Prediction Algorithms

The spectral evolution model can be exploited to derive new link prediction algorithms, which in turn can be used for recommender systems. In this thesis, we will derive two new link prediction algorithms, both of which give only accurate results when network growth is spectral:

- A curve fitting algorithm, which works by supervised learning of a spectral transformation based on observed network growth. This algorithm is suitable for networks in which growth is spectral and regular.
- An extrapolation algorithm, in which the growth of individual eigenvalues is projected into the future, resulting in predictions for new edges. This algorithm is suitable for networks whose growth is spectral but irregular.

While the two algorithms both assume the correctness of the spectral evolution model, they are otherwise very different: This first algorithm assumes smooth spectral transformations, whereas the second algorithm assumes irregular spectral transformations. Both algorithms are applied to a collection of over one hundred network datasets in experiments. In addition to these two new algorithms, we also show evaluation results for common link prediction algorithms as a baseline.

1.5 Outline

Chapter 2 introduces spectral graph theory, which is the basis for the eigenvalue and singular values decompositions. Then, we present the large collection of networks datasets used in this work, including the Slashdot Zoo, which was extracted in the course of writing this thesis from the technology news website `slashdot.org`. This chapter makes the following contributions:

- A survey of one hundred and eighteen network datasets (2.2)
- A new dataset, the Slashdot Zoo (2.2.3)

Chapter 3 introduces the main observation of this thesis: The spectral evolution model. This chapter presents several tests on the network collection to validate the model directly, showing that eigenvectors of networks stay approximately constant over time, while eigenvalues grow. This chapter makes the following contributions:

- A new model of graph growth, the spectral evolution model (3.1)
- The latent preferential attachment model (3.3.5)

Chapter 4 describes methods for learning link prediction functions based on spectral transformations, generalizing a certain number of standard link prediction algorithms. Two new algorithms are presented: one based on curve fitting, the other on spectral extrapolation. This chapter makes the following contributions:

- A new link prediction algorithm based on curve fitting (4.2)
- A new link prediction algorithm based on spectral extrapolation (4.3)

Chapter 5 studies the special case of networks with weighted edges. A particular case are edges with negative weight, which leads to an extension of the Laplacian matrix. A special focus is given on the implicit multiplication rule resulting from negative weights. This chapter makes the following contributions:

- A new spectral graph drawing algorithm for signed networks (5.2)
- A formalization of the algebraic conflict (5.4.4)
- The introduction of signed spectral clustering (5.5)
- The application of graph kernels to link sign prediction (5.6)
- The signed resistance distance (5.8.1)

Chapter 6 presents the special case of networks with asymmetry. These are either bipartite, resulting in a rectangular biadjacency matrix, or directed, giving an asymmetric square adjacency matrix. In both cases, the eigenvalue decomposition is replaced with the singular value decomposition. This chapter makes the following contributions:

- The hyperbolic sine and odd Neumann pseudokernels for bipartite link prediction (6.1.2)

Each chapter will give short experimental results relevant to one area. Appendix A gives the list of all datasets used in the dissertation, along with basic statistics about them. The complete evaluation results are listed in Appendix B. Finally, Appendix C gives a short guide to computing matrix decompositions using GNU Octave.

1.6 Publications and Own Contributions

This thesis contains material published previously in conference papers, as listed in Table 1.1. Material included from these papers was written by myself. Papers by me which did not contribute to this thesis are omitted.

The analysis of the Slashdot Zoo in Section 2.2.3 is based on a paper published at the International World Wide Web Conference 2009 [KLB09]. The Slashdot Zoo was extracted and analysed by me. Parts of the analysis of the spectral evolution model in Chapter 3 and of the spectral extrapolation algorithm in Section 4.3 were published at the International Conference on Information and Knowledge Management 2010 [KFB10]. The spectral evolution and extrapolation methods were developed by myself. The curve fitting algorithm for link prediction described in Section 4.2 was originally published at the International Conference on Machine Learning 2009 [KL09], and was developed by me. Chapter 5 on signed networks is based on work published at the Industrial Conference on Data Mining 2007 [KS07], at the European Conference on Artificial Intelligence 2008 [KSB+08] and at the SIAM International Conference on Data Mining 2010 [KSLL10]. In these papers, I contributed the parts relevant to this thesis: the derivation of the signed Laplacian, the implementation of the graph kernels, and the proofs. The work on bipartite networks in Section 6.1 is based on a paper published at the International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems 2010 [KDLA10]. This paper was written by me, with coauthors having a supportive role. The paper [KSN+10] was not published during my PhD studies. All datasets presented in that paper were prepared by me. The paper about link prediction using temporal tensor decomposition presented at the Behavioral Informatics Workshop 2011 [SCAK11] was mainly written by Stephan Spiegel and Jan Clausen; my contribution lies in showing the equivalence between the tensor decomposition-based method and joint diagonalization, as explained in Section 4.5.

Table 1.1: List of papers written for this PhD thesis.

Datasets
The Slashdot Zoo [KLB09] (WWW 2009)
A survey of network datasets [KSN+10] (Unpublished)
Link Prediction
Assessing unrated items [KLMA07] (ICDIM 2007)
Adapting ratings [KA07] (IRI 2007)
Tensor decomposition [SCAK11] (BI@PAKDD 2011)
Graph Kernels
Similarity kernels [KLB08] (ICPR 2008)
Kernel scalability [KLBA08] (RecSys@ECAI 2008)
Learning Spectral Transformations
Learning by curve fitting [KL09] (ICML 2009)
Spectral extrapolation [KFB10] (CIKM 2010)
Signed Networks
Electrical resistance models with negative edges [KS07] (IndCDM 2007)
Signed resistance distance [KSB+08] (ECAI 2008)
Signed Laplacian kernels [KSLL10] (SDM 2010)
Bipartite Networks
Bipartite link prediction [KDLA10] (IPMU 2010)

Chapter 2

Networks

The central data structure used in this thesis is the network. An example of a network is given by the World Wide Web, which consists of websites connected by hyperlinks. In this example, websites represent nodes of the network and hyperlinks represent links between nodes. This structure of nodes connected by links can also be found in many other domains: persons connected by friendship, words connected by lexical relationships, users and emails sent between them, etc. In this thesis, we will look at networks in an abstract way: We are only interested in the network itself, not in the identity of nodes. Thus, we will not know the names of users in a social network, or the content of websites on the Web, or the content of emails. Instead, only the structure of the network consisting of nodes and links will be important.

Networks are used in applications such as recommender systems, rating prediction or similarity search. As we will see, these problems can be understood as cases of *link prediction*, in which a set of links is known, and new links must be predicted. For instance, being able to predict new friendship links in a social network can be used to recommend new friends to people. Likewise, predicting new hyperlinks in the World Wide Web network can be used to find websites similar to a given one. These different types of applications can all be seen as special cases of the link prediction problem: Given a network that changes over time, predict which links will appear in the future. Thus, a major test of any network model is its applicability for link prediction. In order to evaluate the network model introduced in this thesis, network datasets are needed. For this purpose, we collected one hundred and eighteen large network datasets. These datasets were collected over the course of the PhD program, and are mostly available on the World Wide Web. One dataset was acquired during the writing of this thesis: The Slashdot Zoo, the social network from the website `slashdot.org`. The Slashdot Zoo is special in that it contains not only friendship links, but also enemy links.

This chapter has two parts. First, in Section 2.1, we define the concept of network, review different types of networks encountered in practice, and give the necessary mathematical definitions from graph theory. Then, in Section 2.2, we present the collection of network datasets used in this thesis. This dataset collection contains one hundred and eighteen datasets of different types.

2.1 Definitions

Networks can be observed in many areas of data mining, machine learning, recommendations and related areas. Many datasets published both online and offline are in fact networks.

In the social sciences, graph theory has for a long time be used to study social networks. As an early example from psychology, the term *small world* was made popular by Stanley Milgram in the 1960s to describe the surprisingly low number of links needed to reach any node from

any other node in the person–person graph [Mil67]. More recently, the World Wide Web, or simply the *Web*, has emerged as the giant network of pages connected by hyperlinks. In the last years, the Linked Data initiative has proceeded to turn the Web into a semantic network of interconnected entities, giving rise to the Semantic Web.

Other areas of computer science too have moved to considering network data rather than numerical data. As an example, consider machine learning, where the most recent subtopics are concerned with network-like data structures. The topic of relational learning is by definition concerned with relations in the mathematical sense, which are typically modeled as networks. In probabilistic analysis, graphical models such as Bayesian networks and Markov random fields are used to model sets of random variables that are connected by dependencies. A major class of newer machine learning problems can be summarized as *link prediction*, in which links must be predicted given known links between nodes. This formalism is for instance used in recommender systems, where new links between users and products are predicted.

Outside computer science, networks are routinely used to model all kinds of things, from atoms to the global economy. In chemistry, molecules are modeled as networks of connected atoms. In quantum physics, networks consisting of particles and their interactions are aggregated into Feynman diagrams as a way to compute probabilities of reactions between them. In biology, the relations between genes and the interactions between proteins and other metabolites are modeled as networks in which a path of incident edges is called a biochemical pathway. Also in biology, a series of predator-prey relationships is called a food chain, and multiple food chains form a food web. In neuroscience, a population of interconnected neurons is modeled as a neural network. In epidemiology, the percolation of diseases from one organism to the other is modeled as a network. In sociology, various types of relationships between persons, groups and even countries are studied using network analysis. In linguistics, the relations between words make up a *lexical network*. In economics, the relations between actors are modeled as networks. Transportation and other infrastructural networks are used in operations research. For instance, the worldwide network of airports and flights connecting them forms a network that combines geographic and geopolitical aspects. In technology, a sensor network consists of individual autonomous sensors connected wirelessly to other nearby sensors.

2.1.1 Network Properties

A basic network consists only of nodes and undirected links connecting them. Many real networks however have more structure: Links may be directed and weighted, there may be two fundamentally different nodes types such as users and items, and links may be annotated with link creation times. These additional network properties can be taken into account without much change to the underlying methods we use in this thesis. We consider three basic properties:

- Edge structure: Links may be undirected, directed, or bipartite.
- Edge types: Edges may be simple, multiple, signed or weighted.
- Metadata: Edges may or may not be annotated with timestamps.

Edge Structure The simplest networks are undirected, meaning that an edge between nodes i and j does not have a direction. In other words, we can consider the edge $i \to j$ to be equivalent to the edge $j \to i$. An example of an undirected network is a social network such as Facebook[1], in which a friendship between two users does not have an inherent orientation. If edges have

[1] facebook.com

an orientation, then the network is directed. As an example, the fact that user i follows user j on a social networking site such as Twitter[2] does not imply that user j follows user i. The Twitter network is thus directed. In an authorship network such as that from DBLP[3], scientific publications are connected to their authors. The DBLP network has two classes of nodes (authors and publications), with all links connecting nodes of different types; the DBLP authorship network is thus bipartite. In folksonomies such as Delicious[4], tags are assigned to an item by a user. Thus, tag assignments connect three classes of nodes; they form a tripartite hypergraph.

Edge Types The simplest kind of edge is a simple edge, such as in the Facebook friendship network. Since there can be at most one friendship link between any two users, the Facebook social network is a simple graph. In networks with simple edges, links in the network are persistent. In the case of Facebook, this means that a friendship link stays active forever once it has been created[5]. On the other hand, in networks such as email networks, a link represents a message written by a user to another user. Since users can write any number of emails at different points in time, an email network may have multiple links between a given user pair. This type of graph is called a multigraph. In networks with multiple edges, individual links usually represent individual events between two nodes, and therefore the graph represents the sum of past interactions. On the technology news site Slashdot[6], users can not only mark other users as friends, but also as foes. In the Slashdot Zoo, a *friend* link can be seen as positive and a *foe* link as negative. The Slashdot Zoo thus has positive and negative links; it is a signed graph. In the Netflix movie recommender[7], users rate movies on a five star rating scale. Links are thus weighted by a rating, and the network is weighted.

Metadata Arguably, all networks are created by a growth process, in which nodes and links are added over time. To capture this information, links may be annotated with link creation times. Nodes too may be annotated with creation times, but we will ignore this information since the addition of an unconnected node does not change a network structurally—it is only the addition of its first link that changes the structure of the network.

The resulting network properties are summarized in Table 2.1. Examples of networks of all types are given in Figure 2.1. Directed and undirected networks as defined here have only one node type, as opposed to bipartite networks, which have two node types. Therefore, we will call directed and undirected networks *unipartite*.

2.1.2 Network Categories

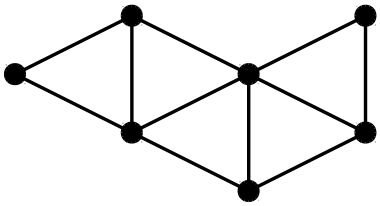
Being so diverse, the networks we study fall into several common categories. In each of these categories, nodes and links are of one particular type, such as person or document for nodes and friendship or citation for links.

Authorship networks are unweighted bipartite networks consisting of links between authors and their works. In some authorship networks such as that of scientific literature, works have typically a few authors, whereas works in other authorship networks may have many authors, as in Wikipedia articles. In an authorship network, vertices corresponding to

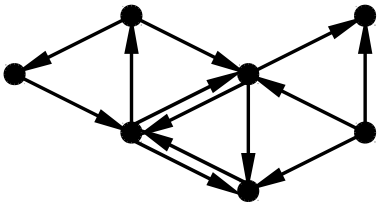
[2] twitter.com
[3] www.informatik.uni-trier.de/∼ley/db
[4] delicious.com
[5] We do not consider link removal, since no datasets in our collection contains such information.
[6] slashdot.org
[7] netflix.com

Table 2.1: Network properties.

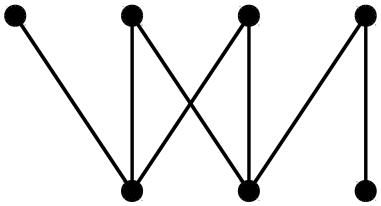
Edge structure		
U	**Undirected**	Edges are symmetric
D	**Directed**	Edges are asymmetric
B	**Bipartite**	Edges connect two types of nodes
T	**Tripartite**	Hyperedges connect three types of nodes
Edge types		
–	**Simple**	Simple edges
=	**Multiple**	Multiple edges are possible
±	**Signed**	Positive and negative edges
⋆	**Weighted**	Edges are ratings
Metadata		
🕒	**Time**	Edges have timestamps



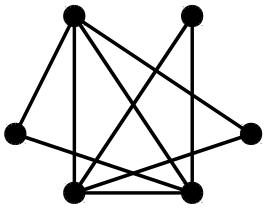
(a) Undirected simple network, e.g. friendship between persons



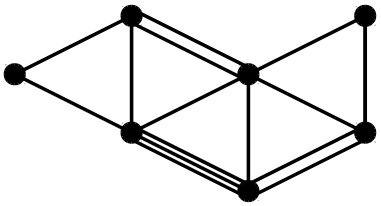
(b) Directed network, e.g. hyperlinks between websites



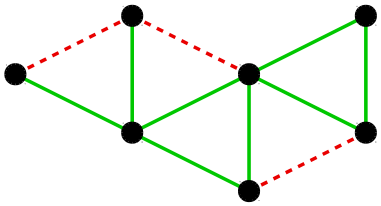
(c) Bipartite network, e.g. membership of persons in groups



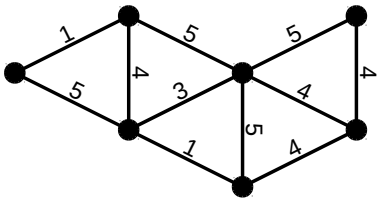
(d) Tripartite network, e.g. tag assignments between users, items and tags



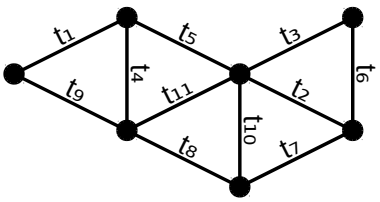
(e) Multiple edges, e.g. email messages



(f) Signed edges, e.g. friendship and enmity between persons



(g) Ratings, e.g. people rating each other



(h) Edges creation times, e.g. scientists writing a publication together at a given date

Figure 2.1: Examples of the network properties we consider in this thesis. The case (a) represents the basic network without any special properties. Properties (b–d) are exclusive and optional, and represent the different kinds of edge structure. Properties (e–g) are exclusive and optional, and represent possible edge types. Property (h) is optional and represents additional edge metadata.

works are added one by one with all their incident edges, and the link prediction problem is not meaningful. Therefore, we also consider author-author networks of co-authorships. In these networks, each edge represents a joint authorship, with the timestamp representing the publication date. Example of authorship networks are the Wikipedia user–article edit network, and the DBLP network of scientists and their publications.

Communication networks contain edges that represent single messages between persons. Communication networks are directed and may have multiple edges. Timestamps denote the date of a message. Examples of communication networks include emails, Facebook wall posts and Twitter posts addressed using the "@name" notation.

Co-occurrence networks represent the simultaneous appearance of items. Co-occurrence networks are unipartite, undirected and unweighted. Examples are the co-purchase network from Amazon, and the *is-similar* relationship of DBpedia.

Feature networks are bipartite, and denote any kind of feature assigned to entities. This includes user memberships in online groups, as well as player memberships in sports clubs. Feature networks are unweighted and have edges that are not annotated with edge creation times. Examples are the genre of songs and the clubs in which a football player has played.

Folksonomies consist of tag assignments connecting a user, an item and a tag. For folksonomies, we follow the 3-bipartite approach and consider the three possible bipartite networks, i.e. the user–item, user–tag and item–tag networks. This allows us to apply methods for bipartite graphs to hypergraphs, which is not possible otherwise. Examples of folksonomies are Delicious for bookmarks, CiteULike for scientific publications and MovieLens for movies.

Interaction networks represent collections of single events between entities. Most interaction networks have edges with timestamps. Examples are the interaction of proteins and other molecules in biological organisms and the Last.fm user–song listening network.

Physical networks represent physically existing network structures. This includes physical computer networks and road networks. These networks result from an underlying two-dimensional geographical layout. Examples are road networks and the autonomous systems of the Internet.

Rating networks consist of assessments given to entities by users, weighted by a rating value. Most rating networks are bipartite, when users rate items. A few rating networks are unipartite and directed, when users rate other users. Most rating networks are weighted; others are signed, when there is an explicit neutral rating value of zero. Examples are the Netflix movie ratings and Jester joke ratings.

Reference networks consist of citations or hyperlinks between publications, patents or web pages. Reference networks are directed. In most reference networks, edges are created along with nodes. Therefore, the link prediction problem cannot be applied to them. An exception are hyperlink networks, where links can be added at any time. Examples are hyperlinks between pages on the World Wide Web and citations between scientific publications.

Social networks represent relations of friendship between persons. Some social networks have negative edges, which denote enmity. All social networks have simple or signed edges, since it is not possible to add another user multiple times to one's friend (or foe) list. In

social networks, timestamps denote when a link was established. Examples are Facebook friendships, the Twitter follower relationship, and friends and foes on Slashdot.

Trust networks connect persons or entities by links of trust. Trust networks are necessarily directed. Links may also denote distrust, in which case negative edge weights are used. Examples are given by the sites Epinions and Advogato.

To study all these kinds of networks, we need a common mathematical framework that allows us to generalize from an actual network to an abstract mathematical structure. At the same time, our mathematical framework must be flexible enough to represent each of these network types. As we will show, this mathematical framework is given by *graph theory*.

2.1.3 Graph Theory

A network is represented mathematically by a graph, in which nodes are called vertices and links are called edges. The simplest type of graph is undirected and unweighted. More complex structure is given by directed, bipartite, weighted and signed graphs. As a general rule in this thesis, we will mean a simple, undirected, unweighted graph whenever we speak of a *graph*, and use the attributes *directed*, *bipartite*, *weighted*, *multiple* and *signed* when needed.

We denote a simple, undirected and unweighted graph $G = (V, E)$, where V is the set of vertices (nodes), and $E \subseteq \{\{i, j\} \mid i, j \in V\}$ is the set of edges. If $i, j \in V$ are two vertices, $\{i, j\}$ denotes the edge connecting them. Two vertices are called adjacent if they are connected by an edge. When two vertices i and j are adjacent, we write $i \sim j$. These types of graphs are *simple* because only one edge may connect a given vertex pair, *undirected* because edges do not have an orientation, and *unweighted* because edges do not have weights. We do not exclude loops, i.e. an edge between a node and itself. An example of a loop is an email sent from one person to themselves in a communication network. Other types of networks are free of loops, such as friendship networks.

A directed graph $D = (V, A)$ consists of a set of vertices V and a set of arcs $A \in V \times V$. An arc $a = (i, j) \in A$ is directed, and is said to point from i to j. Directed graphs are also called *digraphs*.

A graph is bipartite if its vertices can be partitioned into two sets such that all edges connect vertices from different partitions. In other words, a graph $G = (V, E)$ is bipartite if there is a partition $V = V_1 \dot\cup V_2$ such that $E \subseteq \{\{i, j\} \mid i \in V_1, j \in V_2\}$. Bipartite networks arise for instance when modeling the interaction between users and items. Another example of common bipartite networks are collections of text documents: Documents and words can be modeled as the two vertex sets, with an edge connecting a document with a word if the word is contained in the document.

A graph may allow multiple edges between any vertex pair. We will call the number of edges between i and j the multiplicity of $\{i, j\}$, and denote it $m(i, j)$. The corresponding graph is $G = (V, E, m)$. We define a weighted graph as $G = (V, E, w)$, where $w(i, j)$ is defined as the weight of the edge $\{i, j\}$. We do not consider the case of weighted multigraphs, since we did not encounter any network dataset of this form. Analogously, a directed graph with multiple edges is denoted $D = (V, A, m)$ and a weighted directed graph is denoted $D = (V, A, w)$.

A signed graph is a weighted graph in which two kind of edges are distinguished: positive and negative edges [Zas82]. In other words, the weights $w(i, j)$ of a signed graph can be positive or negative. If $w(i, j)$ is always positive for all edges $\{i, j\}$, the graph is unsigned. In a strict sense, a signed graph only has the possible edge weights $+1$ and -1. Indeed, many of the signed graphs we study only contain the weights $\{\pm 1\}$. Other signed graphs however may have other possible weight values. We will not differentiate between the two cases.

A hypergraph is a generalization of a graph in which edges are replaced by hyperedges that can connect any number of vertices. A hypergraph can be written as $H = (V, Y)$, where V is the vertex set and $Y \subseteq \mathcal{P}(V)$ is the hyperedge set. Here, $\mathcal{P}(V)$ is the power set of V, i.e. the set of all subsets of V. In our collection of network datasets, only folksonomies such as Delicious are hypergraphs, and they have a special structure: The vertex set can be partitioned into users, items and tags as $V = V_\mathrm{u} \,\dot\cup\, V_\mathrm{i} \,\dot\cup\, V_\mathrm{t}$, and all hyperedges are of the form $\{i, j, k\}$, with $i \in V_\mathrm{u}$, $j \in V_\mathrm{i}$ and $k \in V_\mathrm{t}$. In other words, hyperedges of folksonomies have cardinality three and connect a user, an item and a tag. Therefore, folksonomies are called tripartite or 3-regular hypergraphs. In order to apply matrix decompositions to them, we break down hypergraphs into three bipartite components, each of which is generated by *forgetting* one of the three vertex types. Thus, the user–item bipartite component of a folksonomy is the bipartite graph with multiple edges $G_\mathrm{ui} = (V, E, m)$ consisting of the vertices $V = V_\mathrm{u} \,\dot\cup\, V_\mathrm{i}$ and the edge set $E = \{\{i, j\} \mid \exists k : \{i, j, k\} \in Y\}$ with $m(i, j) = |\{k \mid \{i, j, k\} \in Y\}|$. This construction results in the three bipartite graphs G_ui, G_ut and G_it.

Example The small synthetic graph G_s in Figure 2.2 will serve as a running example throughout the text. This graph is unweighted, undirected and has nine vertices and eleven edges.

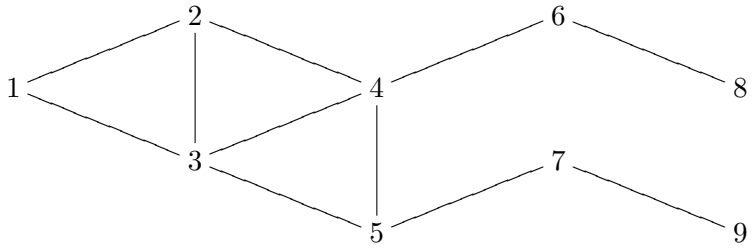


Figure 2.2: A small synthetic graph G_s with nine vertices (1–9) and eleven edges.

2.1.4 Linear Algebra

Linear algebra is the branch of mathematics that is concerned with matrices and vectors. Matrices and vectors can be used to represent many different kinds of objects and structures. In this thesis, we will use matrices to represent networks. Therefore, we introduce basic notions of linear algebra here. Note that we only give the necessary definitions needed in this thesis. Most definitions can be stated in more general ways. This section is not a general introduction to linear algebra. Instead, it is meant as a reminder of the basic notions of linear algebra needed in this thesis for readers that already have a basic grasp of linear algebra.

A vector $\mathbf{x} \in \mathbb{R}^n$ of size n consists of n real numbers. We will write $\mathbf{x}_i$ to denote the i^{th} component of $\mathbf{x}$ for $i \in \{1, \ldots, n\}$. Vectors will have bold lowercase names such as $\mathbf{u}$, $\mathbf{v}$, etc. The norm of a vector $\mathbf{x} \in \mathbb{R}^n$ is defined as

$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{n} \mathbf{x}_i^2}. \tag{2.1}$$

If $\|\mathbf{x}\| = 1$, then $\mathbf{x}$ is said to be a unit vector. The dot product of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ is defined as

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} \mathbf{x}_i \mathbf{y}_i. \tag{2.2}$$

If $\mathbf{x} \cdot \mathbf{y} = 0$, then $\mathbf{x}$ and $\mathbf{y}$ are said to be orthogonal.

A matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ of size $m \times n$ consists of mn real numbers arranged using indexes $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$. The component in the i^{th} row and j^{th} column of $\mathbf{X}$ is written $\mathbf{X}_{ij}$. Matrices will have bold uppercase names such as $\mathbf{A}$, $\mathbf{B}$, etc. A matrix is square if $m = n$. The Frobenius norm of a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is defined by

$$\|\mathbf{X}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{X}_{ij}^2}. \tag{2.3}$$

The transpose $\mathbf{X}^{\mathrm{T}}$ of a square matrix $\mathbf{X}$ is defined by

$$(\mathbf{X}^{\mathrm{T}})_{ij} = \mathbf{X}_{ji}. \tag{2.4}$$

A matrix $\mathbf{X}$ is symmetric if $\mathbf{X} = \mathbf{X}^{\mathrm{T}}$. Symmetric matrices are necessarily square. A matrix $\mathbf{X}$ is diagonal if $\mathbf{X}_{ij} = 0$ whenever $i \neq j$. The i^{th} row of $\mathbf{X} \in \mathbb{R}^{m \times n}$ is a vector denoted $\mathbf{X}_{i\bullet}$. The j^{th} column of $\mathbf{X}$ is a vector denoted $\mathbf{X}_{\bullet j}$. The product of two matrices $\mathbf{X} \in \mathbb{R}^{m \times r}$ and $\mathbf{Y} \in \mathbb{R}^{r \times n}$ is defined as

$$(\mathbf{XY})_{ij} = \mathbf{X}_{i\bullet} \cdot \mathbf{Y}_{\bullet j}. \tag{2.5}$$

Some simple matrices have special notations. The unit matrix $\mathbf{I}_{n \times n}$ is the diagonal matrix of size $n \times n$ defined by $(\mathbf{I}_{n \times n})_{ii} = 1$ for all i. The matrices $\mathbf{0}_{m \times n}$ and $\mathbf{1}_{m \times n}$ are the matrices of all zeroes and of all ones of size $m \times n$. We omit the indexes if they are clear from the context.

Examples

$$\begin{bmatrix} -0.6 \\ 0.8 \end{bmatrix} \in \mathbb{R}^2 \text{ is a unit vector}$$

$$\begin{bmatrix} 1 & -1 & 2 \\ -1 & 3 & 0 \\ 2 & 0 & -2 \end{bmatrix} \in \mathbb{R}^{3 \times 3} \text{ is a symmetric matrix}$$

2.1.5 Algebraic Graph Theory

In order to analyse graphs, algebraic graph theory is a common approach. In algebraic graph theory, a graph with n vertices is represented by an $n \times n$ matrix called the adjacency matrix, from which other matrices can be derived.

Adjacency Matrix The edge set of a graph $G = (V, E)$ can be represented by a matrix whose characteristics follow those of the graph. An unweighted undirected graph on n vertices can be represented by an $n \times n$ 0/1 matrix $\mathbf{A}$ defined by:

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } \{i, j\} \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

The matrix $\mathbf{A}$ is called the adjacency matrix of G. For a directed graph D, $\mathbf{A}$ is in the general case not symmetric:

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } (i, j) \in A \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

If $G = (V, E, m)$ is a graph with multiple edges, the adjacency matrix is defined as

$$\mathbf{A}_{ij} = \begin{cases} m(i,j) & \text{if } \{i,j\} \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

For a weighted graph, we define

$$\mathbf{A}_{ij} = \begin{cases} w(i,j) & \text{if } \{i,j\} \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

The adjacency matrix of a directed graph with multiple edges or edge weights is defined analogously.

Biadjacency Matrix In a bipartite graph $G = (V_1 \dot\cup V_2, E)$, the adjacency matrix can be written as

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^{\mathrm{T}} & \mathbf{0} \end{bmatrix}, \tag{2.10}$$

where the $|V_1| \times |V_2|$ matrix $\mathbf{B}$ is called the biadjacency matrix. As in this study all bipartite graphs are undirected, $\mathbf{A}$ is symmetric, and the analysis of $\mathbf{A}$ can be reduced to the analysis of $\mathbf{B}$. As a general rule, we will only use the biadjacency matrix $\mathbf{B}$, and not mention its adjacency matrix $\mathbf{A}$.

Degree Matrix Degree matrices are diagonal matrices with the degree of each vertex as diagonal values. In an unweighted undirected network with n nodes, the diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ is defined such that $\mathbf{D}_{ii}$ equals the number of vertices adjacent to the vertex i. It is defined as

$$\mathbf{D}_{ii} = \sum_j |\mathbf{A}_{ij}|. \tag{2.11}$$

The absolute value in this definition is necessary when dealing with negatively weighted edges, as explained in Chapter 5. Equivalently, the degree matrix may be defined as $\mathbf{D} = \mathbf{1}_{1\times n}|\mathbf{A}|$, where $\mathbf{1}_{1\times n}$ is the matrix of size $1 \times n$ filled with ones, and $|\mathbf{A}|$ is the absolute value of $\mathbf{A}$, i.e. the matrix defined by $|\mathbf{A}|_{ij} = |\mathbf{A}_{ij}|$. In some works, the matrix $\mathbf{D}$ as defined here for weighted graphs is called the *strength matrix*, and the word *degree matrix* is reserved for the corresponding unweighted network [BBPSV04]. Here, we always use the weighted variant of $\mathbf{D}$ if weights are available. Note that in networks with multiple edges, both concepts overlap, since k parallel edges of weight one can be interpreted as a single edge of weight k. When talking about the degree of a single vertex i, we will also use the notation $d(i) = \mathbf{D}_{ii}$.

If $\mathbf{A}$ is not symmetric, we define the diagonal outdegree matrix $\mathbf{D}_{\text{out}} \in \mathbb{R}^{|V|\times|V|}$ and the diagonal indegree matrix $\mathbf{D}_{\text{in}} \in \mathbb{R}^{|V|\times|V|}$ separately:

$$(\mathbf{D}_{\text{out}})_{ii} = \sum_j |\mathbf{A}_{ij}| \tag{2.12}$$

$$(\mathbf{D}_{\text{in}})_{ii} = \sum_j |\mathbf{A}_{ji}| \tag{2.13}$$

Similarly, for bipartite graphs, we define two diagonal degree matrices $\mathbf{D}_1$ and $\mathbf{D}_2$, corresponding to the vertex sets V_1 and V_2:

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 \end{bmatrix} \tag{2.14}$$

$$(\mathbf{D}_1)_{ii} = \sum_j |\mathbf{B}_{ij}| \tag{2.15}$$

$$(\mathbf{D}_2)_{ii} = \sum_j |\mathbf{B}_{ji}| \tag{2.16}$$

Normalized Adjacency Matrices The adjacency matrix $\mathbf{A}$ and the biadjacency matrix $\mathbf{B}$ can be normalized in the following way:

$$\mathbf{N} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \quad \text{for undirected networks} \tag{2.17}$$

$$\mathbf{N} = \mathbf{D}_{\text{out}}^{-1/2}\mathbf{A}\mathbf{D}_{\text{in}}^{-1/2} \quad \text{for directed networks} \tag{2.18}$$

$$\mathbf{M} = \mathbf{D}_1^{-1/2}\mathbf{B}\mathbf{D}_2^{-1/2} \quad \text{for bipartite networks} \tag{2.19}$$

We call $\mathbf{N}$ the normalized adjacency matrix and $\mathbf{M}$ the normalized biadjacency matrix. In these definitions, we use the power $\mathbf{D}^{-1/2}$ of a diagonal matrix $\mathbf{D}$, which is again a diagonal matrix and can be computed entrywise using the expression $(\mathbf{D}^{-1/2})_{ii} = (\mathbf{D}_{ii})^{-1/2}$. This type of normalization effectively divides the weight of each edge $\{i, j\}$ be the geometric mean of the degrees of i and j. If $G = (V, E)$ is an unweighted undirected graph, then its normalized adjacency matrix is the non-normalized adjacency matrix of the weighted graph $G' = (V, E, w)$ with edge weights $w(i, j) = \sqrt{d(i)d(j)}$, where $d(i)$ is the degree of vertex i. Therefore, the normalized adjacency matrix $\mathbf{N}$ has been used when the degree distribution of the network is very skewed for various learning tasks [GMZ03].

Laplacian Matrix The Laplacian matrix $\mathbf{L}$ is a characteristic matrix of a graph of the same size as the adjacency matrix. For undirected, unipartite graphs the Laplacian matrix is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{2.20}$$

The matrix $\mathbf{L}$ is symmetric. $\mathbf{L}$ is also called the combinatorial Laplacian or the Kirchhoff matrix, due to the relation to electrical networks, as described in Section 4.2.3.

Laplacian matrices can also be defined for directed graphs. These Laplacians are either symmetric as in [Chu05], or asymmetric as in [BRZ11]. These constructions are not used in this thesis. Instead, we ignore edge directions when applying the Laplacian matrix to a directed graph with asymmetric adjacency matrix $\mathbf{A}$, giving

$$\mathbf{L} = \mathbf{D}_{\text{out}} + \mathbf{D}_{\text{in}} - \mathbf{A} - \mathbf{A}^{\mathrm{T}} \tag{2.21}$$

In bipartite graphs, the Laplacian matrix is given by

$$\mathbf{L} = \mathbf{D} - \mathbf{A} = \begin{bmatrix} \mathbf{D}_1 & -\mathbf{B} \\ -\mathbf{B}^{\mathrm{T}} & \mathbf{D}_2 \end{bmatrix} \tag{2.22}$$

This bipartite Laplacian is square and symmetric. We additionally define the normalized Laplacian as

$$\mathbf{Z} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} \quad \text{for unipartite graphs} \tag{2.23}$$

$$\mathbf{Z} = (\mathbf{D}_{\text{out}} + \mathbf{D}_{\text{in}})^{-1/2}\mathbf{L}(\mathbf{D}_{\text{out}} + \mathbf{D}_{\text{in}})^{-1/2} \quad \text{for directed graphs} \tag{2.24}$$

$$\mathbf{Z} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} \quad \text{for bipartite graphs} \tag{2.25}$$

This normalized Laplacian matrix is related to the normalized adjacency matrix $\mathbf{N}$ and to the normalized biadjacency matrix $\mathbf{M}$ in the following way:

$$\begin{aligned}
\mathbf{Z} &= \mathbf{I} - \mathbf{N} && \text{for unipartite graphs} \\
\mathbf{Z} &= \mathbf{I} - \mathbf{N} - \mathbf{N}^{\mathrm{T}} - \mathbf{D}_{\mathrm{in}}^{-1/2}(\mathbf{A} + \mathbf{A}^{\mathrm{T}})\mathbf{D}_{\mathrm{out}}^{-1/2} && \text{for directed graphs} \\
\mathbf{Z} &= \begin{bmatrix} \mathbf{I} & \mathbf{M} \\ \mathbf{M}^{\mathrm{T}} & \mathbf{I} \end{bmatrix} && \text{for bipartite graphs}
\end{aligned}$$

Example Here are various matrices associated to the graph G_{s} introduced in Figure 2.2:

$$\mathbf{A} = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}$$

$$\mathbf{N} = \begin{bmatrix}
0 & 1/\sqrt{6} & 1/\sqrt{8} & 0 & 0 & 0 & 0 & 0 & 0 \\
1/\sqrt{6} & 0 & 1/\sqrt{12} & 1/\sqrt{12} & 0 & 0 & 0 & 0 & 0 \\
1/\sqrt{8} & 1/\sqrt{12} & 0 & 1/4 & 1/\sqrt{12} & 0 & 0 & 0 & 0 \\
0 & 1/\sqrt{12} & 1/4 & 0 & 1/\sqrt{12} & 1/\sqrt{8} & 0 & 0 & 0 \\
0 & 0 & 1/\sqrt{12} & 1/\sqrt{12} & 0 & 0 & 1/\sqrt{6} & 0 & 0 \\
0 & 0 & 0 & 1/\sqrt{8} & 0 & 0 & 0 & 1/\sqrt{2} & 0 \\
0 & 0 & 0 & 0 & 1/\sqrt{6} & 0 & 0 & 0 & 1/\sqrt{2} \\
0 & 0 & 0 & 0 & 0 & 1/\sqrt{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1/\sqrt{2} & 0 & 0
\end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix}
2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & -1 & -1 & 3 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 2 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 2 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1
\end{bmatrix}$$

2.1.6 Spectral Graph Theory

Spectral graph theory is a branch of algebraic graph theory that applies matrix decompositions to characteristic graph matrices in order to study a graph's properties [Chu97, CRS97]. The word *spectral* refers to the spectrum of networks, which is given by the eigenvalue decomposition of a graph's adjacency or Laplacian matrix, as described in the following. Spectral graph theory can be used to study graph properties such as connectivity, centrality, balance and clustering.

Let $\mathbf{X}$ be any $m \times n$ matrix, not necessarily symmetric or square. Then, the equation

$$\mathbf{X}\mathbf{u} = \lambda\mathbf{u} \tag{2.26}$$

is called the eigenvalue equation. If the pair $(\lambda, \mathbf{u})$ is a solution to this equation, the vector $\mathbf{u}$ is called an *eigenvector* of $\mathbf{X}$ and λ is called an *eigenvalue* of $\mathbf{X}$. The pair $(\lambda, \mathbf{u})$ is called an *eigenpair* of $\mathbf{X}$. The set of all eigenvalues λ is called the *spectrum* of $\mathbf{X}$. If $\mathbf{u}$ is an eigenvector of $\mathbf{X}$, then any multiple of $\mathbf{u}$ is also an eigenvector of $\mathbf{X}$. Therefore, the set of vectors corresponding to an eigenvalue λ is a linear space. Its dimension is called the multiplicity of the eigenvalue λ.

Eigenvalue Decomposition A square symmetric matrix $\mathbf{A}$ can be written in the following way:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}} \tag{2.27}$$

where $\mathbf{U}$ is an $n \times n$ orthogonal matrix, and $\mathbf{\Lambda}$ is an $n \times n$ diagonal matrix. A matrix $\mathbf{U}$ is orthogonal when $\mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{I}$, or equivalently when $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$. Another characterization of an orthogonal matrix $\mathbf{U}$ is that its columns are pairwise orthogonal vectors and each has unit norm. The values $\mathbf{\Lambda}_{kk}$ are the eigenvalues of $\mathbf{A}$, and the columns of $\mathbf{U}$ are its eigenvectors. We will designate the eigenvalues by $\lambda_k = \mathbf{\Lambda}_{kk}$ and the eigenvectors by $\mathbf{u}_k = \mathbf{U}_{\bullet k}$ for $1 \leq k \leq n$.

The eigenvector and eigenvalue with index k form an eigenpair $(\lambda_k, \mathbf{u}_k)$. These eigenpairs can be used to write $\mathbf{A}$ as a sum of rank-one matrices:

$$\mathbf{A} = \sum_{k=1}^{n} \lambda_k \mathbf{u}_k \mathbf{u}_k^{\mathrm{T}} \tag{2.28}$$

Here, each outer product of eigenvectors $\mathbf{u}_k \mathbf{u}_k^{\mathrm{T}}$ results in an $n \times n$ matrix of rank one. The corresponding eigenvalue is λ_k. The set of all eigenvalues $\{\lambda_k\}$ of a matrix is called its *spectrum*. The same eigenvalue λ may appear multiple times in $\mathbf{\Lambda}$. Therefore, we will understand the spectrum as a multiset. For instance, when defining a sum over all eigenvalues, we will implicitly understand that multiple eigenvalues are summed a corresponding number of times.

The eigenvalue decomposition can be used to find the best approximation in a least-squares sense to $\mathbf{A}$. For a given r, the following optimization problem consists of finding a matrix $\tilde{\mathbf{A}}$ of rank at most r that approximates the matrix $\mathbf{A}$:

$$\min_{\tilde{\mathbf{A}}} \left\| \mathbf{A} - \tilde{\mathbf{A}} \right\|_{\mathrm{F}} \tag{2.29}$$

$$\text{s.t.} \quad \operatorname{rank}(\tilde{\mathbf{A}}) \leq r \tag{2.30}$$

A solution $\tilde{\mathbf{A}}$ to this problem can be obtained with the eigenvalue decomposition of $\mathbf{A}$. Assuming that the values in $\mathbf{\Lambda}$ are sorted such that $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$, this solution can be written as:

$$\tilde{\mathbf{A}} = \mathbf{U}_{\bullet\,(1\ldots r)} \mathbf{\Lambda}_{(1\ldots r)\,(1\ldots r)} \mathbf{U}^{\mathrm{T}}_{\bullet\,(1\ldots r)} \tag{2.31}$$

In practice, only rank-reduced eigenvalue decompositions can be computed, with r much smaller than the size n of the matrix. Typical values for r in recommender systems are not larger than 100. Values for all datasets are given in Table A.2 in Appendix A.

Singular Value Decomposition In the general case, an asymmetric square or rectangular matrix does not have an eigenvalue decomposition. Instead, the singular value decomposition can be computed. Let $\mathbf{B}$ be a $m \times n$ rectangular matrix. Then $\mathbf{B}$ can be written as

$$\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} \tag{2.32}$$

where $\mathbf{U}$ is an $m \times \min(m,n)$ orthogonal matrix, $\mathbf{V}$ an $n \times \min(m,n)$ orthogonal matrix, and $\mathbf{\Sigma}$ is a $\min(m,n) \times \min(m,n)$ diagonal matrix. This is the singular value decomposition of $\mathbf{B}$; $\mathbf{\Sigma}$ contains the singular values of $\mathbf{B}$, and the columns of $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vectors of $\mathbf{B}$, respectively. As before, the triples $(\sigma_k, \mathbf{u}_k, \mathbf{v}_k)$ can be used to express $\mathbf{B}$ as a sum of rank-one matrices:

$$\mathbf{B} = \sum_{k=1}^{\min(m,n)} \sigma_k \mathbf{u}_k \mathbf{v}_k^{\mathrm{T}} \tag{2.33}$$

By multiplying the column $\mathbf{U}_{\bullet k}$ and the entry $\mathbf{\Sigma}_{kk}$ with -1, we arrive at an equally valid decomposition. Therefore, we can require without loss of generality that all singular values are nonnegative. Also without loss of generality, we require that singular values are ordered as $\sigma_1 \geq \ldots \geq \sigma_{\min(m,n)} \geq 0$.

For the same reasons as for the eigenvalue decomposition, it is only possible to compute a rank-reduced singular value decomposition in practice, with $r \ll \min(m,n)$. One usually chooses an r with a typical value not larger than 100.

Decomposition of Characteristic Graph Matrices The characteristic graph matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{N}$, $\mathbf{M}$ and $\mathbf{L}$ can all be decomposed using the eigenvalue decomposition when they are symmetric or using the singular value decomposition when not. The resulting eigenvalues and singular values are then called the spectrum of the underlying graph. Certain properties of the decompositions of characteristic graph matrices can be derived from their definitions.

The adjacency matrices $\mathbf{A}$ and $\mathbf{N}$ of undirected graphs are symmetric. The largest eigenvalue of $\mathbf{A}$ is positive, and its smallest eigenvalue has the same absolute value if and only if the graph is bipartite. The largest eigenvalue of $\mathbf{N}$ is one. The smallest eigenvalue of $\mathbf{N}$ is -1 when the graph is bipartite. Otherwise it is larger than -1. The sum of the eigenvalues of $\mathbf{A}$ is zero, i.e. $\mathbf{A}$ has trace zero. The trace of $\mathbf{N}$ is also zero.

All eigenvalues of $\mathbf{L}$ are nonnegative, i.e. $\mathbf{L}$ is positive-semidefinite [Bol98]. In a graph G without negatively weighted edges, the multiplicity of the eigenvalue zero of the Laplacian $\mathbf{L}$ equals the number of connected components in G. If G is connected, the eigenvalue zero has multiplicity one, and the second smallest eigenvalue is known as the algebraic connectivity of the graph [Chu97]. The eigenvector corresponding to that second smallest eigenvalue is called the *Fiedler vector*, and has been used successfully for clustering the nodes of G [DGK04]. The special role of the Laplacian matrix in signed graphs is described in Chapter 5.

In an unweighted graph, the normalized Laplacian matrix $\mathbf{Z} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ can be interpreted as the ordinary Laplacian of a network with normalized edges, in which each edge $\{i,j\}$ has the weight $1/\sqrt{d(i)d(j)}$, where $d(\cdot)$ gives the degree of each node. Therefore, $\mathbf{Z}$ has the same properties as $\mathbf{L}$, i.e. it is positive-semidefinite, and the multiplicity of the eigenvalue zero equals the number of connected components of the graph. The relation $\mathbf{Z} = \mathbf{I} - \mathbf{N}$ can then be used to deduce properties of $\mathbf{N}$: The largest eigenvalue of $\mathbf{N}$ is 1, and its multiplicity equals the number of connected components in the graph.

Table 2.2 gives all combinations of matrix decompositions we will use. This thesis will study the behavior of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{\Sigma}$ and $\mathbf{\Lambda}$ of networks as these grow, to find a growth pattern that can be used to solve the link prediction problem in the corresponding networks.

Applications A certain number of interesting graph properties can be described spectrally, such as connectivity [Moh91], centrality [BP98], conflict and balance [KSLL10], and clustering [Lux07]. As an example, Figure 2.3 shows the small synthetic graph G_{s} from Figure 2.2 (page 13) annotated with the dominant eigenvector of the matrices $\mathbf{A}$, $\mathbf{N}$ and $\mathbf{L}$.

Table 2.2: Graph characteristic matrices and their decompositions.

	Non-normalized	Normalized	Laplacian
Undirected	$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$	$\mathbf{N} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$	$\mathbf{L} = \mathbf{D} - \mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$
Directed	$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$	$\mathbf{N} = \mathbf{D}_{\mathrm{out}}^{-1/2}\mathbf{A}\mathbf{D}_{\mathrm{in}}^{-1/2} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$	$\mathbf{L} = \mathbf{D}_{\mathrm{out}} + \mathbf{D}_{\mathrm{in}} - \mathbf{A} - \mathbf{A}^{\mathrm{T}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$
Bipartite	$\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$	$\mathbf{M} = \mathbf{D}_1^{-1/2}\mathbf{B}\mathbf{D}_2^{-1/2} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$	$\mathbf{L} = \begin{bmatrix} \mathbf{D}_1 & -\mathbf{B} \\ -\mathbf{B}^{\mathrm{T}} & \mathbf{D}_2 \end{bmatrix} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$

For $\mathbf{A}$ and $\mathbf{N}$, we show the eigenvector with highest absolute eigenvalue. These eigenvectors are nonnegative when the graph does not have edges with negative weights, as explained by the Perron–Frobenius theorem, which states that the dominant eigenvector of any matrix with nonnegative entries is nonnegative [Per07]. The dominant eigenvectors of $\mathbf{A}$ and $\mathbf{N}$ can be interpreted as a measure of centrality that is related but not equivalent to PageRank [Bon87], and is called eigenvector centrality. In signed networks, the dominant eigenvector of $\mathbf{A}$ can be used to cluster nodes into groups consistent with edge signs [BL04].

For $\mathbf{L}$, we show the eigenvector with smallest nonzero eigenvalue, which is also known as the Fiedler vector [Fie73]. The Fiedler vector gives a simple clustering of the nodes of the network. By separating nodes with positive and negative values in the Fiedler vector, the network is split into two parts connected by few edges and each containing many internal edges [Lux07]. The eigenvalue corresponding to the Fiedler vector is called the algebraic connectivity of the graph. This value is high when the graph is well-connected. If the graph is unconnected, this value is zero. In graphs with negatively weighted edges, the smallest eigenvalue of $\mathbf{L}$ and $\mathbf{Z} = \mathbf{I} - \mathbf{N}$ are not always zero, and characterize the amount of *conflict* present in the graph. This is described in detail in Chapter 5.

After this introduction of the basic definition from graph theory, the next section introduces the collection of large network datasets used in this thesis.

2.2 The Network Collection

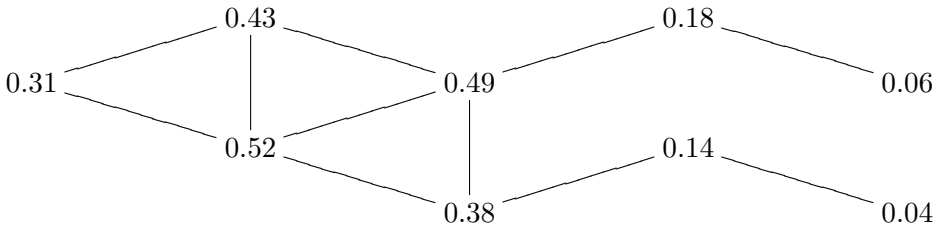
Experiments in this thesis are done on a collection of one hundred and eighteen large network datasets. These datasets are either unipartite (such as social networks) or bipartite (such as user–item rating networks). Some networks have edge weights, e.g. ratings or trust/distrust links, in which case the link prediction problem also consists of predicting the weight of edges. Edge weights admit negative values in some datasets. The networks include social networks, citation graphs, hyperlink graphs, trust networks, rating graphs (i.e. user–item graphs), communication graphs and others. Many of these graphs are used in previous literature in various machine learning and data mining subfields. Edges in some datasets are annotated with timestamps. Figure 2.4 gives an overview of the networks by their size. The full list of datasets is given in Appendix A.

The largest dataset is the Twitter social network with 42 million nodes and 1.5 billion edges, and the dataset with the least number of edges is the metabolic network of *Caenorhabditis elegans*, with 453 nodes and 4,596 edges. From a statistical point of view, larger networks not only contain more data to analyse but are also more regular and thus better suited for statistical analyses. The networks in our collection fall into several common categories, as summarized in Table 2.3. The network categories given in that table are not absolute. Other classifications are possible, and some networks may fit into several categories.

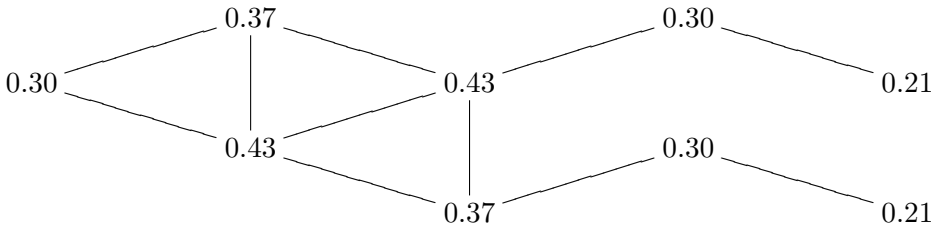
Several types of datasets are specifically not covered here. We exclude networks such as person-birthplace networks, since each person has only one birthplace essentially resulting in a

Table 2.3: Network categories with their properties and typical application domains. For the properties, see Table 2.1.

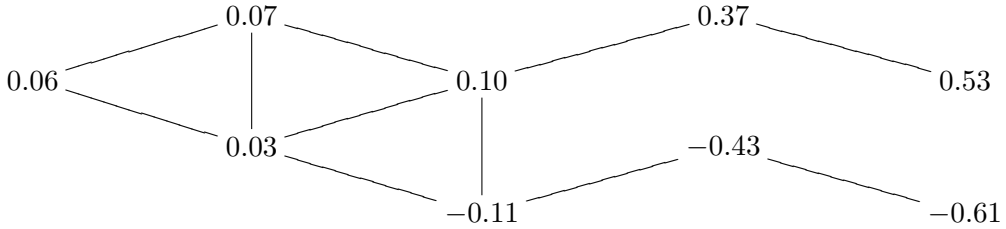
	Links	Nodes	Properties		Applications	Count
Authorship	(Co-)authorship	Authors, works	U B	−=	Topic analysis	43
Communication	Message	Persons, items	UDB	−=	Link prediction	9
Co-occurrence	Co-occurrence	Items	D	−	Classification, clustering	2
Features	Has-feature	Objects, features	B	−=	Classification	15
Folksonomy	Tag assignment	Users, tags, items	T	−=	Trend analysis	6
Interaction	Event	Entities	UDB	=	Link prediction	5
Physical	Connection	Nodes	UD	−=	Routing	6
Ratings	Rating	Users, items	DB	±⋆	Recommendation	9
Reference	Citation, hyperlink	Documents	D	−	Ranking	11
Social	Friendship	Persons	UD	− ±	Communities	9
Trust	Trust	Persons	D	−=±	Security	3
Total						118



(a) Eigenvector centrality (adjacency matrix **A**)



(b) Eigenvector centrality (normalized adjacency matrix **N**)



(c) Fiedler vector (Laplacian matrix **L**)

Figure 2.3: The dominant eigenvectors of characteristic graph matrices. In these graphs, each node i is labeled with the component $\mathbf{u}_i$ of the dominant eigenvector of one characteristic graph matrix of the network. In the cases (a) and (b), the eigenvectors are nonnegative due to the Perron–Frobenius theorem, as explained in the text, and represent the eigenvector centrality of nodes. The vector shown in (c) is also known as the Fiedler vector.

feature vector, making link prediction, degree distributions and other applications impossible or trivial. Also, dense complete networks are excluded for similar reasons, and because they are usually very small. Similarly, we exclude trees and forests[8], since their simply-connected structure makes all path-based statistics trivial. All networks we consider are connected or almost connected, meaning that most nodes can be reached from any node by following a path along edges. Disconnected networks are usually uninteresting, since the properties of each connected component can be studied separately, and many of the typical applications such as link prediction are not sensible for them.

2.2.1 Overview

Because of the large number of datasets used, we give each dataset a short two- or three-letter code, such as LJ for the LiveJournal social network or NX for the Netflix rating network. The complete list of datasets and their codes is given in Table A.1 in Appendix A. Due to the special structure of folksonomies, we use a different naming scheme for them. Each folksonomy such as Delicious or BibSonomy has a one-letter code such as D for Delicious and B for BibSonomy. A folksonomy consists of users, tags and items. Each edge in a folksonomy is a hyperedge connecting one user, one tag and one item. For hypergraphs, we only consider the three underlying bipartite subgraphs consisting of users and tags, users and items or tags and items. These subgraphs are denoted with the indexes ut, ui and ti. For instance, the user–item subgraph of Delicious is denoted D_{ui}. Mathematically, we will call the set of all networks in the collection $\mathcal{G}$, with $|\mathcal{G}| = 118$.

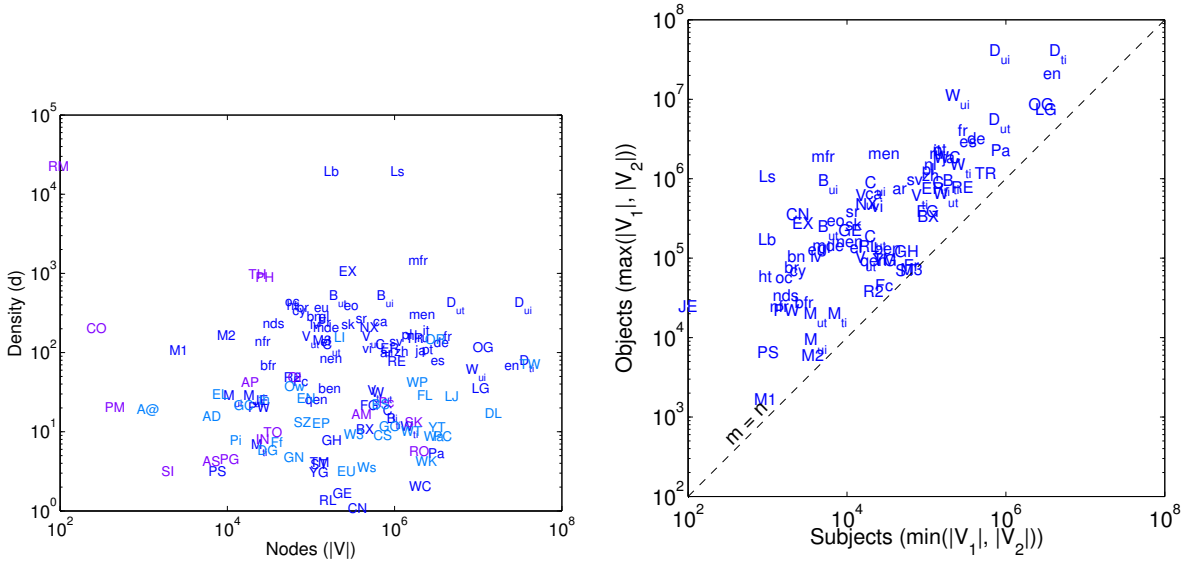
Figure 2.4(a) shows all 118 datasets arranged by number of nodes and edges. As in all subsequent scatter plots, each network is represented by its two-letter code given in Table A.1. This plot shows that in general, the number of edges grows linearly with the number of nodes. In other words, the mean number of neighbors is *not* related to network size. The mean number of neighbors in a network is called the density, and as we will see later, can be observed to grow over time in most networks. The independence of the density to the network size shows that this is a strictly local phenomenon, i.e. only valid for a single network at different times. This also implies that the proportion of existing vertices to all possible vertices is inversely proportional to the network size. Therefore, larger networks are more and more sparse in the sense that the amount of present edges is much less than the number of possible edges, which grows quadratically with the node count.

Figure 2.4(b) shows all bipartite datasets arranged by the count of both vertex types. The most skewed bipartite networks are Jester (JE) and the French Wiktionary (mfr) with few items for many users, and last.fm (Ls, Lb), Delicious (D), Twitter (W) and BibSonomy (B), with many items for few users.

2.2.2 Network Models

In this section, we present a brief survey of common network models applied to our network collection. Properties of networks can be summarized by *characteristic numbers*, each one describing a specific aspect of the network. In the literature, characteristic numbers have often been used to validate network models, i.e. patterns of network growth that lead to specific values for these characteristic numbers. Using characteristic numbers, networks can be identified as scale-free networks, small-world networks and evolving networks. We review these three models in turn.

[8] Forests are graphs without cycles. Trees are connected forests.



(a) All networks by number of nodes and density

(b) All bipartite networks by nodes

Figure 2.4: An overview of the size of all datasets. (a) All networks by the number of nodes and the density, defined as the mean number of neighbors for each node. (b) All bipartite networks and bipartite subgraphs of folksonomies by the two node counts.

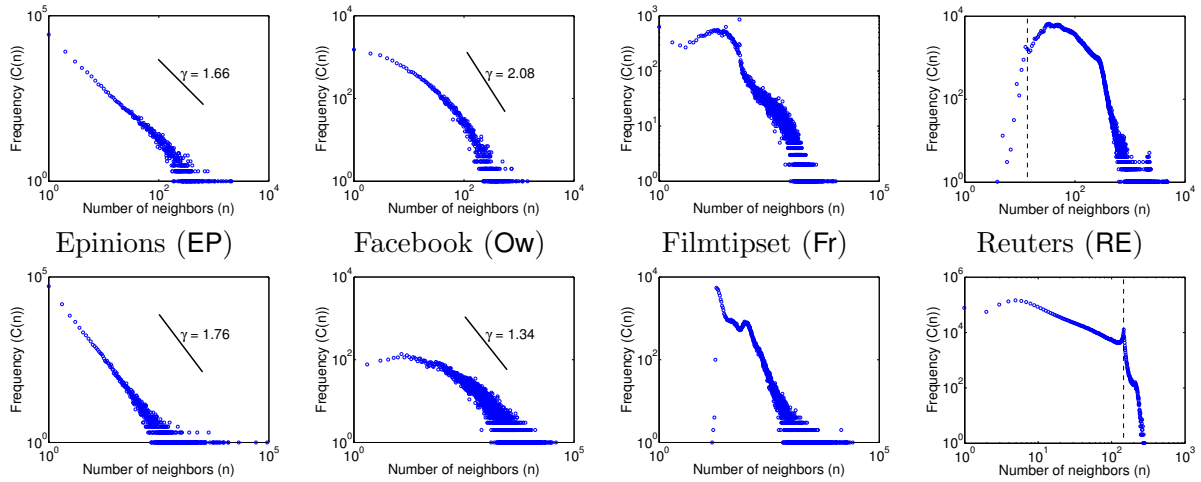
Scale-free Networks

Real-world networks are far from being random. A random graph as defined by Erdős and Rényi [ER59] is a graph in which it is assumed that each edge is present with a given probability independently of the presence of other edges. In random graphs, the number of neighbors of a vertex follows a binomial distribution. However, the degree distributions of actual networks have been observed to *not* follow a binomial distribution, making the Erdős–Rényi model unsuitable for the modeling of real networks. Instead, many networks were observed by Barabási and Albert to follow power-law degree distributions [BA99]. Such networks are called scale-free networks. In this section, we verify whether the networks under study can be qualified as scale-free.

The random graph and scale-free network models both arise from a network growth process: Random graphs can be generated by postulating that every edge is present with probability p, and scale-free networks by assuming that the probability of an edge being formed is proportional to the degree of the two connected nodes, given a degree distribution.

Two distributions are typically studied: the distribution of vertex degrees, and the distribution of edge multiplicities. The degree distribution in a scale-free network can be described by a power law, stating that the number of nodes with n neighbors is proportional to $n^{-\gamma}$. The constant γ is usually considered to characterize the preferential attachment observed in a network, and represents an important graph characteristic. Figure 2.5 shows degree distributions for many networks, customarily drawn on a double logarithmic scale, such that an exact power law is represented by a line. We observe a simple power law in many cases, but other cases are more complex. In certain networks, a power law is only observed for a certain range of the distribution.

Some degree distributions seem to follow the discrete Gaussian exponential (DGX) model, which corresponds to a discretized lognormal distribution, and is represented by an apparent squared dependency on degree distribution plots [BFK01]. In some networks, the degree distribution seems to follow more complex patterns, which we conjecture could be modeled as



(a) Power law (b) Lognormal (c) Mixtures (d) Limits

Figure 2.5: Diversity of degree distributions observed in various networks. The number of nodes $C(n)$ having exactly n neighbors is plotted against n, on a double logarithmic scale. The parameter γ was estimated to fit $C(n) \sim n^{-\gamma}$ using the method from [New06b]. (a) Power-law degree distributions. (b) Lognormal degree distributions. (c) Mixed degree distributions. (d) Degree distributions with lower and upper limits.

mixtures of DGX distributions. We also observe that *passive* degree distributions are more likely to follow power laws than *active* degree distributions. In a user–item rating network for instance, the user degree distribution is active in the sense that the user himself is active in giving ratings, and the item degree distribution is passive. We attribute this to the fact that active distributions are more dependent on the behavior of the specific case. A unipartite example for this are citation networks: In the DBLP citation dataset (Pi) for instance, the distribution of the number of references peaks at 10, whereas the number of citations received by publications is scale-free. Some degree distributions also contain artifacts of sampling, pruning methods or limits in the number of allowed neighbors.

If a degree distribution follows a power law, its exponent γ can be easily estimated [CSN09]. In Figure 2.6, we plot the estimated power law exponent γ against the number of nodes for all unipartite networks. We observe that the exponent γ does not correlate with network size. This makes the parameter γ a scale-free network characteristic. Additionally, we observe that most values of γ lie between 1.0 and 2.0. This is surprising since a simple preferential attachment model is characterized by a value of $\gamma = 3.0$, and most extended preferential attachment models predict a value in the interval $[2.0, 3.0]$. Most published values for γ are also in this range, see for instance Table I in [New06b]. On the other end of the scale, the listings of Prosper.com (PW) have numbers of members that watch them following a power-law with exponent $\gamma = 10.7$. In other words, only very few listing are watched by many members and the very large majority of listings is watched by only a few members.

Figure 2.7 shows the distribution of edge multiplicities. For unweighted networks with multiple edges between node pairs, we observe edge multiplicity distributions to follow power laws closely, although some variations are apparent. In datasets with weighted edges (of which none has multiple edges), the distribution of edge weights is much less regular. Although weighted edges can be related to multiple edges by interpreting the number of parallel edges as

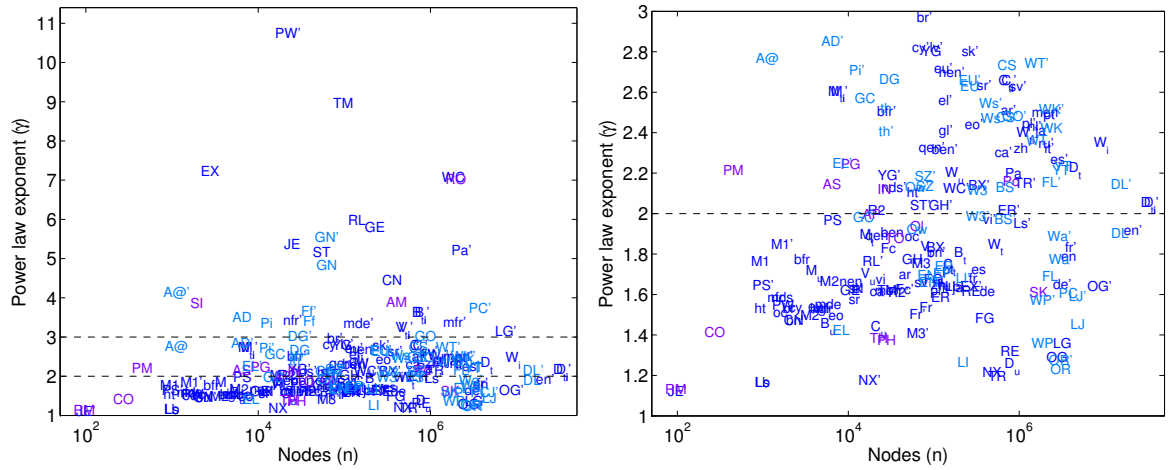
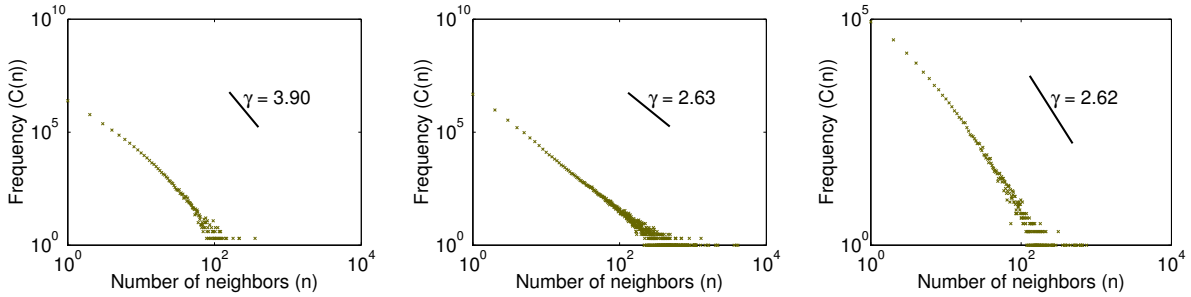


Figure 2.6: The datasets arranged by power law exponent γ. Each bipartite dataset is represented twice, once for each node type. For object nodes in the second node set of bipartite networks, an apostrophe is appended to the name. For folksonomies, the three exponents are marked with an index u, t or i. The image on the right shows the plot restricted to $1.0 \leq \gamma \leq 3.0$.



DBLP (Pc) Portuguese Wikipedia (pt) Facebook (Ow)

Figure 2.7: Examples of edge multiplicity distributions. In these plots, the number of edges having multiplicity $C(n)$ is plotted against n, on a double logarithmic scale. The parameter γ was estimated to fit $C(n) \sim n^{-\gamma}$ using the method from [New06b].

an edge weight, networks with natively weighted edges apparently do not follow any consistent weight distribution.

Small-world Networks

In the previous section, we measured the degree distribution in order to characterize a network. The degree distribution is however only a *local* metric, capturing single nodes and their neighborhood. Most processes in networks are however *global*, such as routing, link prediction, flow and diffusion. To study these phenomena, we must look at global properties of networks.

Routing, flow and diffusion can be modeled as information being passed from one node to the next along edges in the network. To study their behavior on a network-wide scale thus requires us to look at paths through the network. We can thus consider two properties of paths: Their length when they span the whole network, and the probability of them being present. The typical length of paths across a network is made precise by the *network diameter*, which is defined as the maximum length of all shortest paths connecting any two nodes. The probability of an edge appearing between two nodes with a common neighbor is called the

clustering coefficient.

In 1998, Watts and Strogatz [WS98] inspected the characteristic path length and clustering coefficient of several networks and made a surprising observation: The characteristic path length is much lower than predicted, whereas the clustering coefficient is much higher than predicted. This contradicted several previous random graph models, in particular those where edges are randomly distributed and those that are completely regular, forming a lattice. By starting with a regular lattice and randomly rewiring edges to connect random vertices, they were able to reproduce networks with low characteristic path lengths (due to rewiring) and high clustering coefficients (due to the initial lattice). This model became known as the small-world network model. The small-world phenomenon is most famously illustrated by the *six degrees of separation*, made famous in 1967 by social psychologist Stanley Milgram [Mil67, TM69]. Since then, six degrees of separation have been consistently reported in newer studies for even the largest social networks [LH08].

We can now estimate to what extent our datasets are small-world networks by comparing their characteristic path lengths and clustering coefficients. Definitions of both measures vary slightly in the literature. Here, we use the following definitions:

Effective Diameter To indicate the characteristic path length, we use the 90-percentile effective diameter $\delta_{0.9}$, which denotes in how many steps one can reach, on average, 90% of all other nodes, and interpolating between adjacent path length values. The effective diameter can be considered a robust extension of the actual diameter, which suffers from being skewed by long chains [LKF07]. As an alternative, the mean shortest path length between any two nodes is sometimes used.

Clustering Coefficient The clustering coefficient of a network is the probability that two adjacent edges in that network are completed by a third edge to form a triangle. Let $G = (V, E)$ be an unweighted, undirected graph. Then the clustering coefficient is defined as:

$$c = \frac{|\{(i,j,k) \mid \{i,j\} \in E, \{j,k\} \in E, \{i,k\} \in E\}|}{|\{(i,j,k) \mid \{i,j\} \in E, \{j,k\} \in E\}|} \tag{2.34}$$

Instead of the clustering coefficient c, we use the relative clustering coefficient c_{rel}, equal to the clustering coefficient divided by the fill of the network. The fill of the network is the proportion of possible edges that are present, and corresponds to the expected value of the clustering coefficient when edges are distributed randomly as in the Erdős–Rényi model.

$$c_{\text{rel}} = \frac{1}{|E|/|V|^2} c \tag{2.35}$$

As a result, c_{rel} is larger than 1.0 when the clustering coefficient is larger than expected in a random graph. Due to the division by the fill, the relative clustering coefficients of different networks can be compared even if the networks have different sizes.

The resulting scatter plot is shown in Figure 2.8, where the more *small-world* networks are at the top-left. The plot shows that almost all datasets are small-world networks, but some more than others. This is the case for instance for the Wikipedia talk network (WK). On the other hand, the Gnutella (GN) and Pretty Good Privacy (PG) networks are much less small-world networks. One network though is not a small-world network: the California road network (RO). Since a road network is physically a mesh, its diameter is very high (492), and the network is thus not a small world. The second largest effective diameter is found in the DBpedia similarity network (SI), and is about 16.0. Then, the Berkeley/Stanford hyperlink dataset (BS) has diameter around 10.5. Generally, we find hyperlink networks to have high

diameters, in accordance with Albert et al. [AJB99], where a value of 11.2 is reported for a network of 800 million web pages. The collaboration networks of arXiv (PH, TH) have a small diameter, but also a small clustering coefficient. In accordance with the well-known *six degrees of separation* observation, the diameter of the social networks varies in the range $[5.0, 7.0]$ except for Advogato (AD) and the English Wikipedia (EL), which have lower diameter. We can also read from the plot that Líbímseti.cz (LI) and the metabolic network (PM) have small-world characteristics near to a random graph, and that the two hyperlink networks (BS, WT) are more lattice-like.

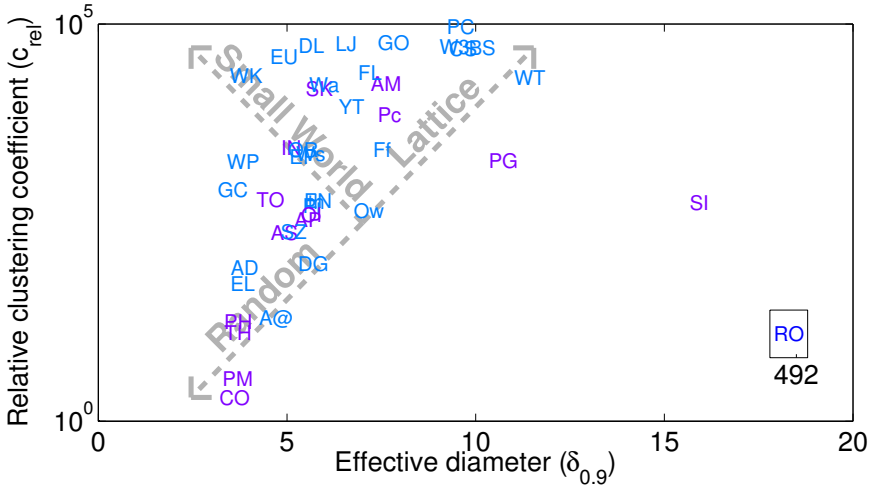


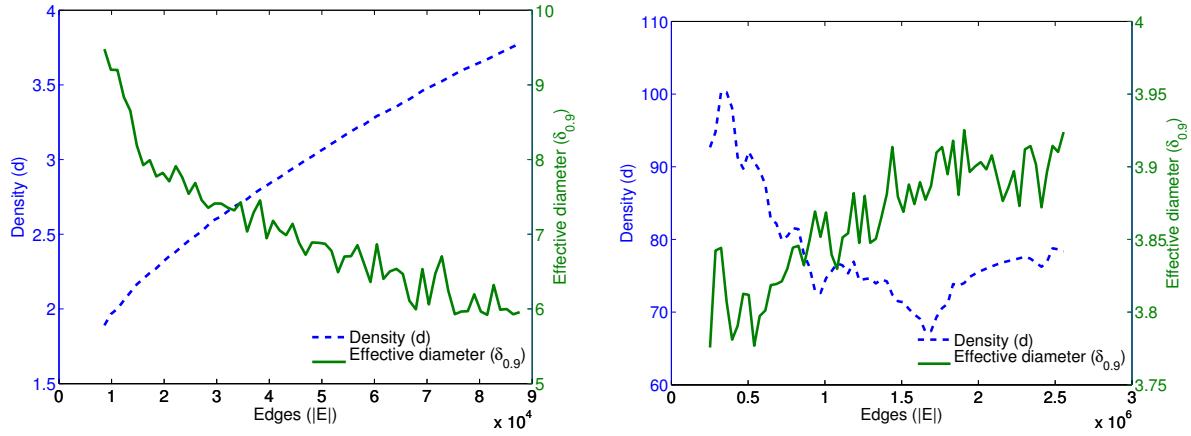
Figure 2.8: Verifying the small-world hypothesis: The datasets arranged by diameter and relative clustering coefficient. The California road network (RO) is outside of the main plot; its effective diameter $\delta_{0.9}$ is about 492.

Evolving Networks

Most networks are the result of an evolution process. It is therefore interesting to study the change of network characteristics over time. In addition to the rate of edge creation, several non-trivial patterns have been observed in several types of networks, in particular that the network density increases and the diameter decreases [LKF07]. Only for a subset of all datasets the edge creation times are known; Table A.1 gives the complete list.

Network dynamics is an important aspect in network analysis because it directly leads to link prediction algorithms in the following way: If a certain pattern is observed in the temporal dynamic of the networks, extrapolating this pattern into the future directly gives a prediction for the future appearance of edges. Being able to predict links in a network then allows one to implement virtually all types of recommender systems.

In [LKF07], Leskovec et al. studied the evolution of network characteristics over time and made the following observations: The diameter of graphs seems to *shrink* and the density to *grow*. This result is surprising, since a scale-free model would indicate at least the diameter to increase with network size. We verify this behavior for our datasets in Figures 2.9 and 2.10. The behavior of the density and the diameter in our datasets is generally as predicted in [LKF07], where it is derived from a graph growth model called *forest fire*. We observe the Netflix (NX),



Digg (DG) BibSonomy (B_{ut})

Figure 2.9: Evolution of network characteristics over time. For two networks, we show, in function of the number of edges $|E|$ in the network, the network's density d and the effective diameter $\delta_{0.9}$.

Digg (DG), YouTube (YT), Internet topology (TO), Facebook (OI) and Last.fm (Ls, Lb) networks to clearly show shrinking diameters in conjunction with increasing density. On the other hand, Enron (EN) and BibSonomy (B) display the opposite behavior. We therefore observe that the forest fire model is less universal than the scale-free and small-world models. We also note that there is no discernible correlation between the temporal evolution of the diameter and density. On another level, we confirm that shrinking diameters and increasing density are also observed for bipartite networks.

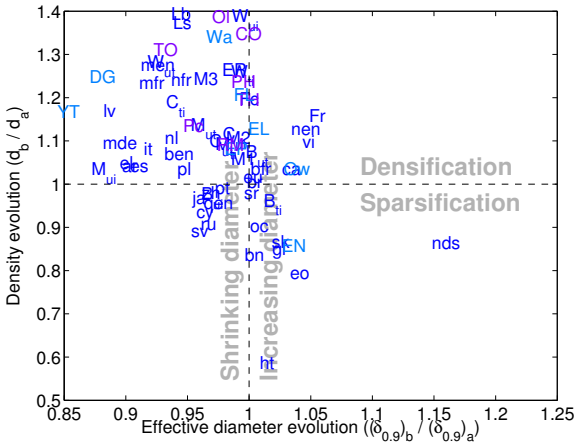


Figure 2.10: The evolution of the diameter $\delta_{0.9}$ and density d over time. This plot compares the ratio of the diameter and density of the complete networks ($(\delta_{0.9})_\mathrm{b}$ and d_b) and of the network containing half of all edges ($(\delta_{0.9})_\mathrm{a}$ and d_a). Only networks with timestamps are shown.

2.2.3 The Slashdot Zoo

In addition to the numerous publically available network datasets listed in Appendix A, one network dataset was extracted during the course of writing this thesis: the Slashdot Zoo (SZ).

Slashdot[9] is a technology news website founded in 1997. It publishes stories written by editors or submitted by users and allows users to comment on them. In 2002, the site added the *Zoo* feature, which lets users tag other users as *friends* and *foes*. In contrast to most popular social networking services, Slashdot is one of the few sites that also allows users to rate other users negatively. The Slashdot Zoo dataset is available online[10].

The Slashdot Zoo network we extracted contains 77,985 users and 510,157 links. Each link consists of a user who gave an endorsement and a user who received the endorsement. Endorsements can be either positive (*friend*) or negative (*foe*). Apart from this distinction, no other information is available; in particular, the creation date of endorsements is not known. In addition to the terms *friend* and *foe*, Slashdot also uses the terms *fan* and *freak*: A user is always the fan of his friends and the freak of his foes. Figure 2.11 summarizes these relationships.

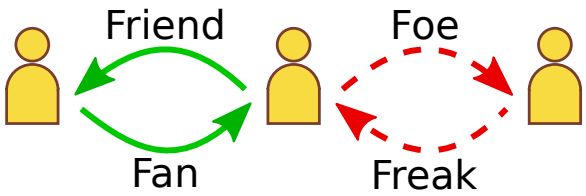


Figure 2.11: The two types of links allowed in the Slashdot Zoo (friend and foe) give rise to four kinds of relationships: friends, fans, foes and freaks. A user is the fan of his friends and the freak of his foes.

Figure 2.12 is a graphical representation of the Slashdot Zoo network. The sign of an edge is represented by its color, with green representing the *friend* relationship and red representing the *foe* relationship. The graph is centered at user *CmdrTaco*, founder of Slashdot and active editor. While most social network modeling approaches allow for weighted edges, the weights are usually restricted to positive values. However, some relationships such as distrust and dislike are inherently negative. In such cases, the social network contains *negative edge weights*. This is the case for the Slashdot Zoo, and will be described in Chapter 5.

The dataset was crawled from slashdot.org between May and October 2008, and the dataset does not represent a true snapshot of the network, and may exhibit anomalies. For instance, because the Slashdot website does not indicate when a friend or foe was added, some users in our dataset may have more than 400 friends or foes, although Slashdot generally limits the number of friends and foes to 200 users and to 400 users for subscribers.

Slashdot is known for both having very popular and prominent users on the one hand, and rather unpopular users on the other hand. Prominent and popular users of Slashdot include CmdrTaco (Rob Malda, the founder of Slashdot and a popular editor), John Carmack (prominent computer game programmer), Bruce Perens (prominent computer programmer and open source advocate) and CleverNickName (Wil Wheaton, Star Trek actor). In addition, Slashdot is well known for having a tradition of *trolling*, i.e. the posting of disruptive, false or offensive information to fool and provoke readers. The high number of such trolls may explain why the *foe* feature is useful on Slashdot. It allows for tagging known trolls and reducing their visibility.

Table 2.4 displays basic statistics of the dataset. All distances were calculated without taking into account the edge directions and signs. The measured average distance is less than the average distance in a random graph, confirming that the Slashdot Zoo is a small-world network. Figure 2.13 shows the degree distributions in the Slashdot Zoo. As expected, the degree distribution in the Slashdot Zoo follows a power law.

[9] slashdot.org
[10] dai-labor.de/IRML/datasets

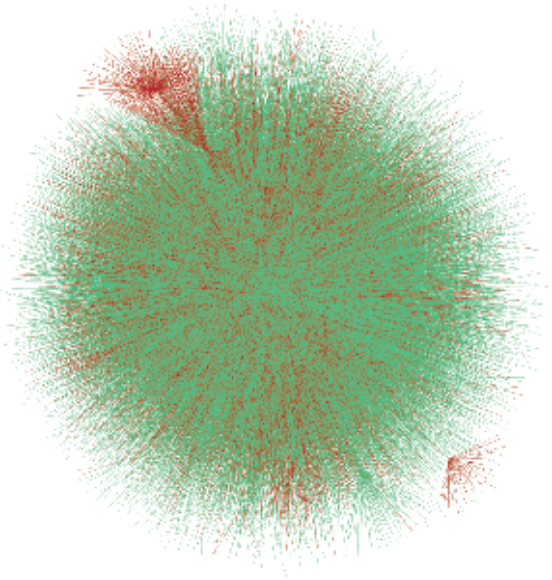
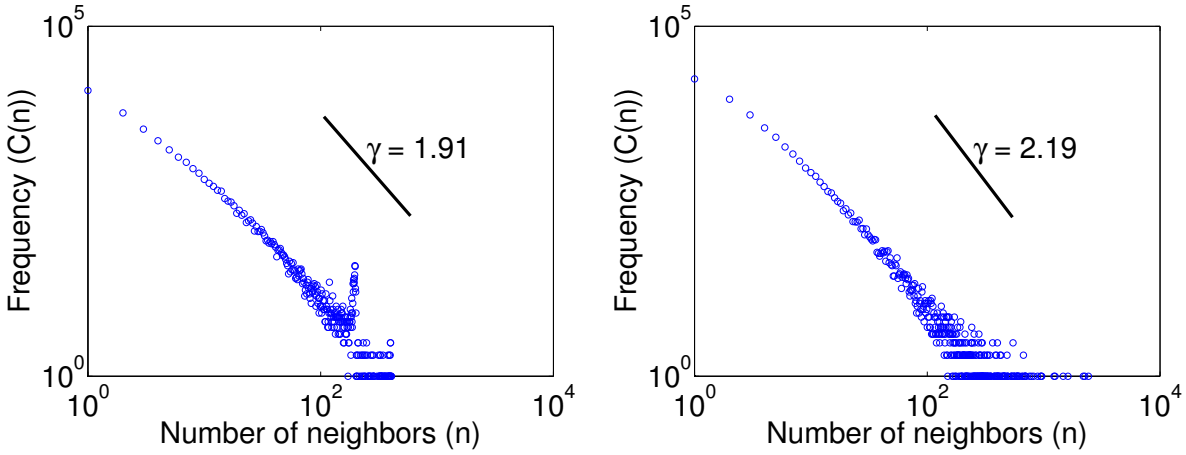


Figure 2.12: The Slashdot Zoo network represented as a graph, where nodes represent users and edges indicate relationships. The network contains 79,120 nodes and 515,581 edges. *friend* relationships are shown in green edges and *foe* relationships in red; the orientation of edges is not shown. The graph is centered at user *CmdrTaco*.

Table 2.4: Statistics about the Slashdot Zoo. The mean friend count and mean fan count are necessarily equal, as do the mean foe and freak counts. The sign and direction of edges was ignored in the calculation of the diameter and radius. In parentheses, we show the average distance in a random graph, as defined by Watts and Strogatz [WS98].

Users	79,120
Links	515,581
Friend links	392,326
Foe links	123,255
Fill	0.000083884
Diameter	6
Radius	3

Mean link count	6.542
Mean friend/fan count	4.978
Mean foe/freak count	1.564
Median links	3
Median friend count	1
Median foe count	0
Median fans count	1
Median freaks count	1



(a) Outdegree (friend and foes)

(b) Indegree (fans and freaks)

Figure 2.13: Double logarithmic plot of the degree distributions in the Slashdot Zoo, showing that they follow a power law. The limit of 200 friends and foes is visible for the outdegree plot (a).

2.3 Summary

Data from many diverse application areas can be modeled as networks. Whether they are unipartite, bipartite, weighted, signed, directed or undirected, graphs play a fundamental role in many disciplines because they can model many real-world interactions. We use as a testing ground in this thesis a collection of one hundred and eighteen network datasets. One of these networks, the Slashdot Zoo, was specifically extracted for this thesis. The Slashdot Zoo is a social network with positive and negative links, and will be the basis for the analysis of signed networks in Chapter 5.

Chapter 3

The Spectral Evolution Model

This chapter describes the *spectral evolution model*, which constitutes the main statement in this thesis. The spectral evolution model characterizes the change of a network in terms of the eigenvalue decomposition of its adjacency matrix. It states that over time, eigenvalues change, while eigenvectors stay constant. The spectral evolution model is stated and verified empirically in this chapter for unweighted undirected unipartite networks. The spectral evolution model serves as the basis for the two new link prediction algorithms described in the next chapter. The case of weighted networks is described in Chapter 5 and that of bipartite and directed networks in Chapter 6. In the examples of this chapter, we will also use directed networks, but always ignore edge orientations for them.

We begin this chapter in Section 3.1 by deriving the spectral evolution model from a simple link prediction model, the matrix exponential kernel. Then, in Section 3.2, we verify the spectral evolution model empirically using our collection of network datasets. After establishing that the spectral evolution model can be observed in many networks, we show that it can be derived from multiple models of graph growth in Section 3.3, in particular from many known graph kernels and from a generalization of the preferential model attachment. These models will give intuitive and mathematical explanations for the spectral evolution model. In Section 3.4, we finally present two control tests to make sure that the spectral evolution model is specifically not implied by random graph models, making the spectral evolution model a non-trivial property of real networks.

3.1 Overview

Many machine learning problems on networks can be described as link prediction: finding movies a user might like, recommending friends, or finding related topics in a collaboration graph. In all these cases, a given network is assumed to *grow* over time and the task is to predict where new edges will appear and, in some cases, to also predict their weight. On the other hand, any given link prediction algorithm can be interpreted as a model of graph growth by assuming that networks actually grow as the algorithm predicts. In this work, we are concerned with those link prediction algorithms that have an algebraic description. We show that they imply a network growth model that we call *spectral evolution*. Our hypothesis states that in terms of the eigenvalue decomposition of a network's adjacency matrix $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$, growth can be described as a transformation of the spectrum $\mathbf{\Lambda}$, without significant change in the eigenvectors $\mathbf{U}$.

Each algebraic link prediction algorithm, including most graph kernels, represents a specific assumption about network growth, leading to a different spectral transformation function. Each of these link prediction functions can be applied in different situations, depending on the assumptions made about the network. Despite this, all algebraic link prediction algo-

rithms are of the same form: a change of the eigenvalues following a specific function, without change in the eigenvectors. Therefore, they are all generalized by the spectral evolution model. Starting with a network's adjacency matrix $\mathbf{A}$, we first compute its eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$. It can then be shown that many common link prediction algorithms can be expressed as $\mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^{\mathrm{T}}$, where F is a function that applies a real function $f(\lambda)$ to each diagonal element of $\mathbf{\Lambda}$.

Consider the following example: The matrix exponential of the adjacency matrix $\mathbf{A}$ is sometimes used as a link prediction function [KL02]. The matrix exponential is defined as $\exp(\mathbf{A}) = \sum_{k=0}^{\infty} \mathbf{A}^k/k!$, and can be interpreted in the following way: $\exp(\mathbf{A})_{ij}$ equals a weighted sum over all paths from the node i to the node j in the network, where paths are weighted by the inverse of the factorial of their length. This description shows why the exponential of the adjacency matrix is a suitable link prediction function:

- The link prediction score is higher when there are many paths between i and j, because the powers of $\mathbf{A}$ count the number of paths.
- The link prediction score is higher when these paths are short, because the weights $1/k!$ are decreasing.

The matrix exponential can also be written in terms of the eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ as

$$\exp(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}) = \mathbf{U}\exp(\mathbf{\Lambda})\mathbf{U}^{\mathrm{T}},$$

where $\exp(\mathbf{\Lambda})$ can be computed by applying the real exponential function to $\mathbf{\Lambda}$'s diagonal elements. This shows that the matrix exponential is a spectral transformation of the adjacency matrix $\mathbf{A}$. We will see later that several other link prediction functions can be expressed as spectral transformations in an analogous way. First however, we state the spectral evolution model.

Definition 1 (Spectral evolution model)**.** *A network that changes over time is said to follow the spectral evolution model when its spectrum evolves while its eigenvectors stay approximately constant.*

This definition is not mathematically strict. Whether a network conforms to the spectral evolution model depends on the interpretation of when an eigenvectors stays *approximately constant*. This fuzziness will be made precise in Section 3.4. For now, we will interpret the definition loosely, taking eigenvectors as approximately constant when their change is small. The spectral evolution model is visualized schematically in Figure 3.1: In this simplified picture, a small network to which edges are added consecutively has three different spectra at three different times, but the same set of eigenvectors at all times.

To examine the validity of the spectral evolution model, we take three steps. First, we verify it empirically in our collection of datasets. Then, we review existing link prediction functions that imply the spectral evolution model and finally, we present control tests to verify that the spectral evolution model does not arise from other processes, such as random graph growth.

3.2 Empirical Verification

Which network model does accurately describe the growth of real networks? Given a network, which link prediction function does best model its growth? To answer these questions, we examine the spectra of large, real-world networks and observe their temporal evolution. In the

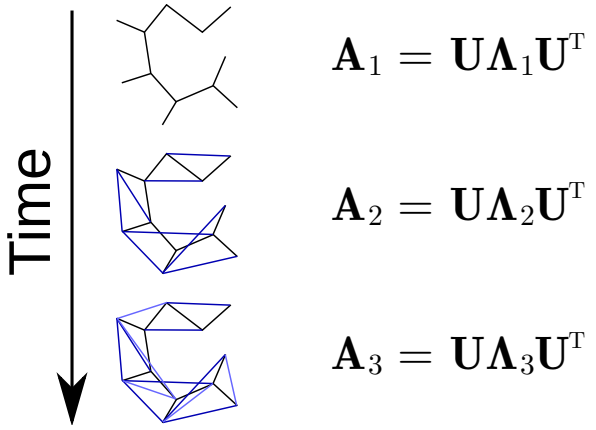


Figure 3.1: Illustration of the spectral evolution model. As edges appear in a network, the eigenvalues of the network's adjacency matrix change, while the eigenvectors stay constant.

following sections, we look at the evolution of the eigenvalue decomposition of the networks given in Table A.1 in Appendix A.

For each network, we split the set of edges into $T = 75$ bins by edge creation time. For each bin, we take the network as it was after the arrival of that bin's edges, and compute the first r eigenvalues of the resulting adjacency matrix. In the networks we study in this chapter, the creation times of edges are known. r is chosen in function of network size to give reasonable runtimes. We denote by $\mathbf{A}_t$ the adjacency matrix of all edges present after time t. Thus $\mathbf{A}_T = \mathbf{A}$ is the full adjacency matrix.

Two representative datasets are used throughout this section:

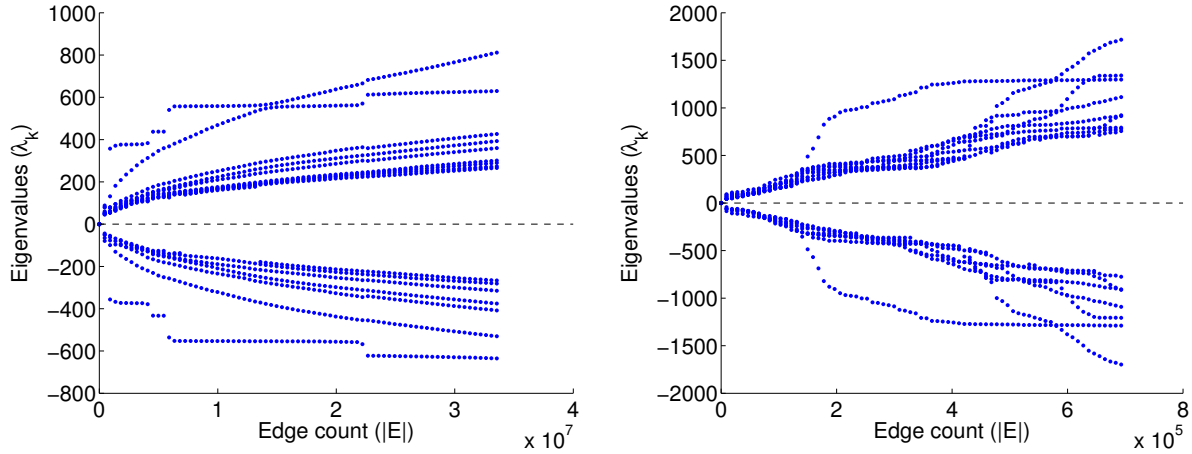
- The English Wikipedia hyperlink network, containing links between articles (WP).
- The Facebook user–user network of wall posts (Ow).

These two networks are unipartite, unweighted, undirected, and link creation times are known for them. The Facebook wall post network has parallel edges, and the English Wikipedia hyperlink network does not. In other words, multiple edges may connect the same vertex pair only in the Facebook graph.

3.2.1 Spectral Evolution

Figure 3.2 shows the spectra of the two networks as functions of time. For each network, the plot shows the top-r eigenvalues λ_k by absolute value in function of time. The value r was chosen in function of the network size to give acceptable runtimes. The list of values by dataset is given in Appendix A.

A first inspection of these plots suggests that the eigenvalues and singular values grow over time. The observed spectral growth of these networks is sometimes regular, as in the case of Wikipedia, or irregular, as in the case of Facebook. By regular growth, we mean that all eigenvalues grow roughly with the same speed. If growth is irregular as for the Facebook dataset, eigenvalues may overtake each other. In many datasets, the observed irregularity is only partial: Growth of eigenvalues as a whole does not seem to follow a specific pattern, although the growth of each individual eigenvalue is smooth. This is a first indication of the irregularity of network growth, and will be a justification later on for extrapolating the eigenvalue growth into the future.



Wikipedia hyperlinks (WP) Facebook wall posts (Ow)

Figure 3.2: The spectral evolution of large real-world networks in function of time. At each time, the graphs shows the r dominant eigenvalues of the network.

We cannot however know at this point that two consecutive eigenvalues with similar value are related to the same eigenvector. In fact, any continuous or small change to a matrix can lead to a small change in its spectrum. For the spectral evolution model to be valid, spectra must not only grow, but eigenvectors corresponding to individual eigenvalues must be stable. This is inspected in the next test.

3.2.2 Eigenvector Evolution

To verify whether the spectral evolution model describes the growth of large, real networks, we compare for each time t the eigenvectors of $\mathbf{A}_t = \mathbf{U}_t \, \mathbf{\Lambda}_t \, \mathbf{U}_t^{\mathrm{T}}$ with the eigenvectors of $\mathbf{A}_T = \mathbf{U}_T \mathbf{\Lambda}_T \mathbf{U}_T^{\mathrm{T}}$ in Figure 3.3. In these plots, we rely on the eigenvectors to be ordered by their corresponding eigenvalue. The plots show one curve for each latent dimension k, showing the number of edges $|E|$ on the X axis and the absolute dot product of the k^{th} eigenvectors at times t and T on the Y axis. The similarity between eigenvectors is computed as the cosine similarity in the following way:

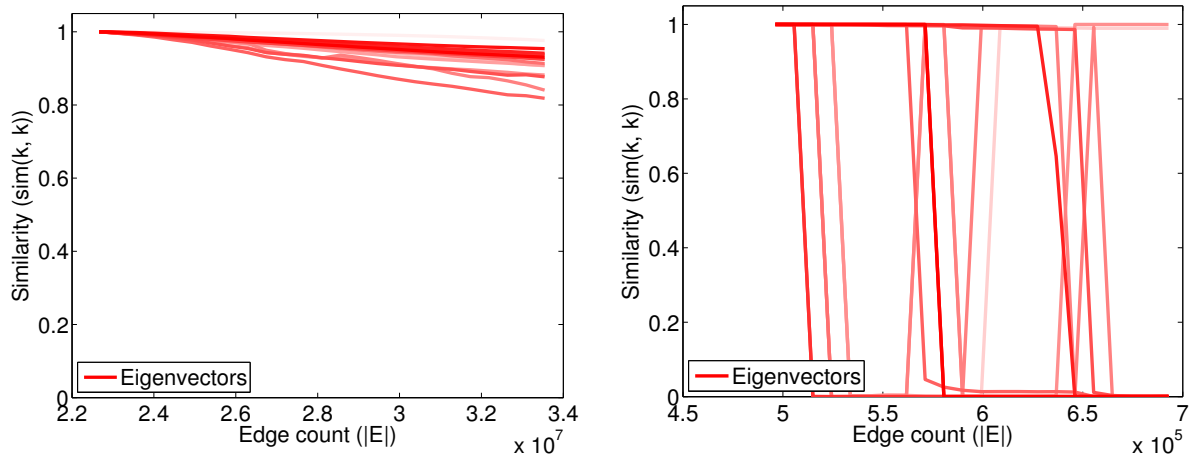
$$\mathrm{sim}(k, l) = |(\mathbf{U}_T)_{\bullet k} \cdot (\mathbf{U}_t)_{\bullet l}| \tag{3.1}$$

The figure shows, for each k, the curve of $\mathrm{sim}(k, k)$ in function of t. Eigenvectors corresponding to large eigenvalues are shown in bright colors and those corresponding to smaller eigenvalues in light colors.

The eigenvector evolution plots suggest the following interpretation. The general trend leaves the eigenvector similarity as measured by the dot product near one for the lifetime of the network, with similarity being higher for those eigenvectors with larger eigenvalues. In the Facebook network, the similarity of eigenvectors suddenly drops to zero at specific points in time. As we will see next, this is due to eigenvectors changing places in the decomposition, or in other words, the eigenvalues changing order. The next test will inspect these permutations.

3.2.3 Eigenvector Stability

How stable are eigenvectors over time? To answer this question we compare the eigenvectors at an intermediary time t with the eigenvectors at the last known time T. In this test, we choose



Wikipedia hyperlinks (WP) Facebook wall posts (Ow)

Figure 3.3: The evolution of eigenvectors: The similarity between the eigenvectors of the full graph and the eigenvectors of partial graphs, by increasing number of missing edges. Each eigenpair is represented by one curve, with brighter colors for dominant latent dimensions.

t such that three quarters of all edges are present at time t. This 75% split is consistent with the training/test split we perform in the next chapter.

We will consider $\mathbf{A}_t$, the adjacency matrix containing all edges present up to time t and $\mathbf{A}_T$, the adjacency matrix of all edges. We consider the following eigenvalue decompositions:

$$\mathbf{A}_t = \mathbf{U}_t \mathbf{\Lambda}_t \mathbf{U}_t^{\mathrm{T}}$$
$$\mathbf{A}_T = \mathbf{U}_T \mathbf{\Lambda}_T \mathbf{U}_T^{\mathrm{T}}$$

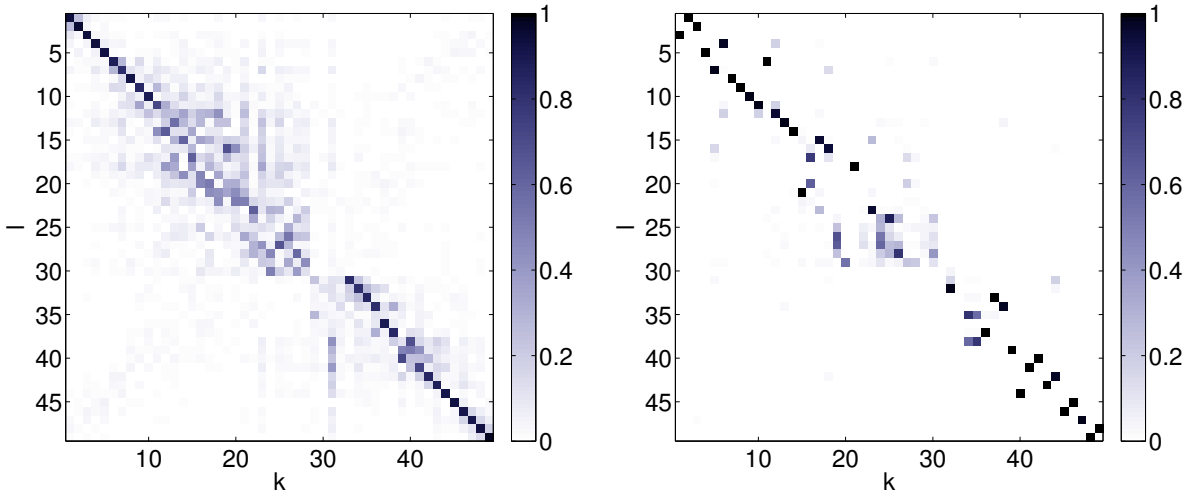
We then compute $\mathrm{sim}(k, l)$ for all pairs of eigenvector indexes (k, l). We show the resulting matrices using white for zero and black for one, and continuous shades in between in Figure 3.4. These plots give an indication as to what extent eigenvectors are preserved over time. If all eigenvalues are distinct and network evolution is purely spectral, the matrices $\mathrm{sim}(k, l)$ are permutation matrices. In addition, they are diagonal when the eigenvalues do not overtake each other. The derivation from a diagonal matrix gives an indication to the monotony of the underlying spectral transformation.

Testing the eigenvector stability in this way has one drawback. If two eigenvalues are equal, an exchange between their eigenvectors does not change the matrix. This case can be recognized on the eigenvector permutation plots of Figure 3.4 as sub-squares containing intermediate values between zero and one. These sub-squares are in fact arbitrary orthogonal matrices, since any orthogonal basis of the multidimensional eigenspace corresponding to a multiple eigenvalue can be returned by the eigenvalue decomposition. To avoid this, the next test is designed to be robust against such multiple eigenvalues.

3.2.4 Spectral Diagonality Test

We now present a test of the amount of change in the eigenvectors which we call the spectral diagonality test. If

$$\mathbf{A}_t = \mathbf{U}_t \mathbf{\Lambda}_t \mathbf{U}_t^{\mathrm{T}}$$



Wikipedia hyperlinks (WP) Facebook wall posts (Ow)

Figure 3.4: The absolute dot product $\mathrm{sim}(k,l) = |\mathbf{u}_k \cdot \mathbf{u}_l|$ of all eigenvector pairs $(\mathbf{u}_k, \mathbf{u}_l)$ at two times t and T in network evolution. These plots show permutation matrices (with zero in white and one in black) when the network evolution is purely spectral and eigenvalues are simple. The derivation from a diagonal matrix gives an indication to the monotony of the underlying spectral transformation.

is the eigenvalue decomposition of a network's adjacency matrix $\mathbf{A}_t$ at an intermediate time t, then at the final time T it is assumed to become

$$\mathbf{A}_T = \mathbf{U}_t(\mathbf{\Lambda}_t + \mathbf{\Delta})\mathbf{U}_t^{\mathrm{T}},$$

where $\mathbf{\Delta}$ should be diagonal. Because $\mathbf{U}_t$ has orthogonal columns, we can compute the best fit of $\mathbf{\Delta}$ in a least-squares sense by

$$\mathbf{\Delta} = \mathbf{U}_t^{\mathrm{T}}(\mathbf{A}_T - \mathbf{A}_t)\mathbf{U}_t. \tag{3.2}$$

The resulting matrix $\mathbf{\Delta}$ is intended to give an indication to what extent growth is spectral. Note that in this expression, $\mathbf{A}_T - \mathbf{A}_t$ is the adjacency matrix containing all edges that have appeared between times t and T. It is shown for the two example networks in Figure 3.5. We can make several observations: First, the two matrices are nearly diagonal. In fact, we make this observation for all datasets in our collection. However, the deviation from a diagonal matrix is not equal for all datasets. For the Facebook wall posts network (Ow), the matrix is very near to a diagonal one, and thus the growth is very nearly spectral. For the Wikipedia hyperlink network (WP), the deviation is larger. This is a pattern that we observed in almost our entire collection of datasets: Datasets where multiple edges are allowed mostly have better spectral growth than those where only simple edges exist. A simple explanation can be given for communication networks such as email networks: In these networks, an edge will appear between already connected nodes with a certain probability, making growth more spectral. In fact, if edges appeared exactly with a frequency proportional to the number of previously existing edges, the adjacency matrix would simply be multiplied by a constant, and growth would thus be spectral.

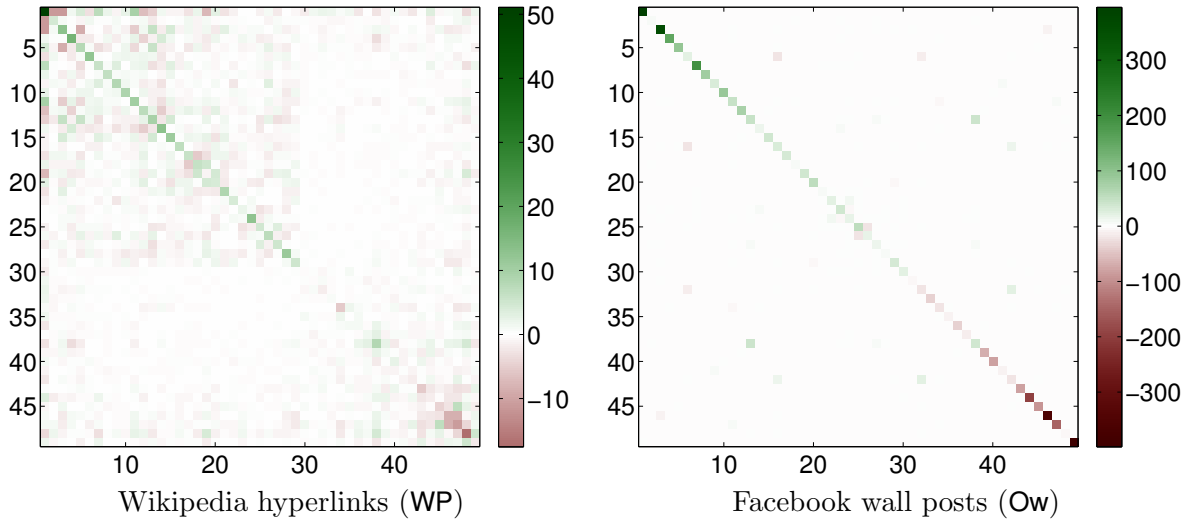


Figure 3.5: The spectral diagonality test of spectral network evolution. If network evolution is perfectly spectral, the plots show a clean diagonal. These plots show the matrix $\mathbf{\Delta}$ as defined in Equation 3.2 for the two example datasets.

3.3 Generalizing Other Models

In the previous section, we have observed that the spectral evolution model holds in many large, real network datasets. In this section, we will derive the spectral evolution model from other, more specific graph growth models. As we will see, many common graph growth models implicitly rely on spectral growth. In other words, the spectral evolution model can be understood as a generalization of the models presented in this section. We will look at these models and study how they can be expressed as a change of a network's eigenvalue decomposition.

3.3.1 Triangle Closing

Arguably, the simplest graph growth model is the triangle closing model. This model predicts that in a graph, new edges will appear between nodes that have common neighbors, thus forming (*closing*) triangles. In social networks, the triangle closing model is known as the *friend of a friend* model. The triangle closing model is one of the easiest to compute nontrivial link prediction methods, and is used in practice on very large datasets [LBKT08].

Algebraically, this model can be expressed as the square $\mathbf{A}^2$ of the adjacency matrix. The matrix $\mathbf{A}^2$ contains, for each vertex pair (i, j), the number of common neighbors of i and j:

$$(\mathbf{A}^2)_{ij} = \sum_k \mathbf{A}_{ik}\mathbf{A}_{jk} \tag{3.3}$$

The triangle closing model will thus predict those links that complete the highest number of triangles. The square $\mathbf{A}^2$ can be expressed using the eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ as

$$\begin{aligned}\mathbf{A}^2 &= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}} \\ &= \mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^{\mathrm{T}}.\end{aligned}$$

Here, we use the fact that $\mathbf{U}$ is orthogonal and thus $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$. As we see, the eigenvalues $\mathbf{\Lambda}$ are replaced by their squares $\mathbf{\Lambda}^2$. The triangle closing model is thus a spectral transformation. Note that this is also true when using the reduced eigenvalue decomposition of $\mathbf{A}$. Since $\mathbf{\Lambda}$ is

diagonal, its square can be computed by squaring each of its diagonal element. This equation shows that the triangle closing model $\mathbf{A}^2$ is a spectral transformation that corresponds to the real function $f(\lambda) = \lambda^2$.

3.3.2 Path Counting

The triangle closing model states that edges will appear between nodes separated by a path of length two. In real networks however, we also may expect edges to connect two vertices that are more than two edges apart. We will thus suppose that new edges complete cycles of any length, but give lower weight to longer cycles [LJZ09]. The number of paths of a given length k between any two vertices of a graph can be expressed by the k^{th} power of its adjacency matrix $\mathbf{A}$. By the same argument as for $\mathbf{A}^2$, we can derive that $\mathbf{A}^k$ is a spectral transformation of $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$:

$$\mathbf{A}^k = \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^{\mathrm{T}}$$

Thus, the k^{th} power of the adjacency matrix corresponds to the spectral transformation $f(\lambda) = \lambda^k$. Because matrix multiplication is distributive, a linear combination of powers of $\mathbf{A}$ is also a spectral transformation. In general, every power series $p(\mathbf{A})$ is a spectral transformation:

$$\begin{aligned} p(\mathbf{A}) &= \mathbf{U}p(\mathbf{\Lambda})\mathbf{U}^{\mathrm{T}} \\ \alpha\mathbf{A}^2 + \beta\mathbf{A}^3 + \gamma\mathbf{A}^4 + \ldots &= \mathbf{U}(\alpha\mathbf{\Lambda}^2 + \beta\mathbf{\Lambda}^3 + \gamma\mathbf{\Lambda}^4 + \ldots)\mathbf{U}^{\mathrm{T}} \end{aligned} \tag{3.4}$$

In particular, a power series $p(\mathbf{A})$ can be used as a graph growth model whenever its coefficients are nonnegative and decreasing, i.e. if $\alpha > \beta > \gamma > \ldots \geq 0$. Power series of this form have two desirable properties of link prediction functions:

- Vertices connected by more paths are more likely to connect in the future (positivity of the coefficients).
- Vertices connected by shorter paths are more likely to connect in the future (ordering of the coefficients).

Note that we do not consider the zeroth and first powers of $\mathbf{A}$: They only contribute to an additive constant for all node pairs.

3.3.3 Graph Kernels

A kernel is a symmetric and positive-semidefinite function of two variables that denotes a similarity or proximity between objects. A graph kernel is defined for a given graph, and denotes the similarity between any two vertices of that graph. Many graph kernels exist, most being derived from specific graph models and typically applied to different applications [ISKM05, FYPS06, SK03, KL02]. Since they denote a form of similarity between two vertices, graph kernels can be used for link prediction.

Note that the null space of a matrix $\mathbf{A}$, i.e. the set of vectors $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{0}$, is also called the *kernel* of $\mathbf{A}$. This meaning of the word *kernel* is not used in this text. The phrase *graph kernel* may also refer to a measure of similarity between individual graphs. This meaning too is not used here.

Many graph kernels are spectral transformations. Each of these graph kernels can be described by a symmetric positive-definite matrix $K(\mathbf{A})$ of the same size as the graph's adjacency matrix $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$, and that can be written as $K(\mathbf{A}) = \mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^{\mathrm{T}}$ for various functions $F(\mathbf{\Lambda})$

which apply a real function $f(\lambda)$ to each eigenvalue in $\mathbf{\Lambda}$. In addition to being spectral transformations, these graph kernels' functions $f(\lambda)$ are convex, which implies that large eigenvalues grow quicker than smaller eigenvalues. By construction, these graph kernels can be computed from the eigenvalue decomposition of $\mathbf{A}$. Because the graph kernels we describe here are based on the adjacency matrix $\mathbf{A}$, we will call them *adjacency kernels*. Graph kernels based on other matrices are introduced later in the next chapter.

Exponential Kernel The matrix exponential of the adjacency matrix is a graph kernel that can be used as a link prediction function [KL02]. A scale parameter α is typically inserted, to balance the relative weight of short and long paths.

$$K_{\mathrm{EXP}}(\mathbf{A}) = \exp(\alpha\mathbf{A}) = \sum_{k=0}^{\infty} \frac{\alpha^k}{k!}\mathbf{A}^k \tag{3.5}$$

It corresponds to the spectral transformation $f(\lambda) = e^{\alpha\lambda}$. When written as a power series, we see that paths are weighted by the inverse factorial of their length. The matrix exponential has nonnegative decreasing coefficients, and is therefore suited to link prediction. The matrix exponential is also called the exponential diffusion kernel.

Neumann Kernel The Neumann kernel is given by

$$K_{\mathrm{NEU}}(\mathbf{A}) = (\mathbf{I} - \alpha\mathbf{A})^{-1} = \sum_{k=0}^{\infty} \alpha^k\mathbf{A}^k, \tag{3.6}$$

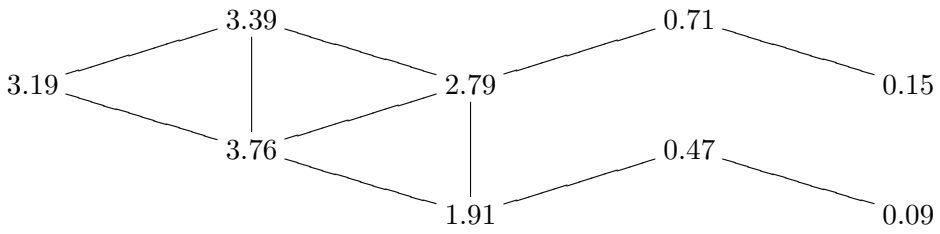
where α is chosen such that $\alpha^{-1} > |\lambda_1|$, $|\lambda_1|$ being $\mathbf{A}$'s largest absolute eigenvalue, or equivalently the graph's spectral norm [KSTC02]. The resulting spectral transformation is $f(\lambda) = 1/(1 - \alpha\lambda)$. As the matrix exponential, the Neumann kernel can be written as a power series with nonnegative decreasing coefficients, and is therefore a suitable link prediction function. The Neumann kernel takes its name from the Neumann series $\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \ldots = (\mathbf{I} - \mathbf{A})^{-1}$ [Mey00], itself named after the German mathematician Carl Gottfried Neumann. In many scientific publications, the Neumann kernel is called the *von Neumann kernel*. We are not aware of how the particle *von* was introduced into this name, and use the name without *von* in this work. The Neumann kernel is sometimes called the Neumann diffusion kernel [FYPS06]. The link prediction function resulting from the Neumann kernel is also called the Katz index [Kat53].

Example Using the small synthetic graph from Figure 2.2, the following Figure 3.6 shows the exponential and Neumann graph kernels. For each kernel $\mathrm{X} \in \{\mathrm{EXP}, \mathrm{NEU}\}$, each vertex i is annotated with the value $K_{\mathrm{X}}(1, i)$, i.e. with the kernel's similarity values between each vertex and vertex 1, the leftmost vertex on the graph's representation.

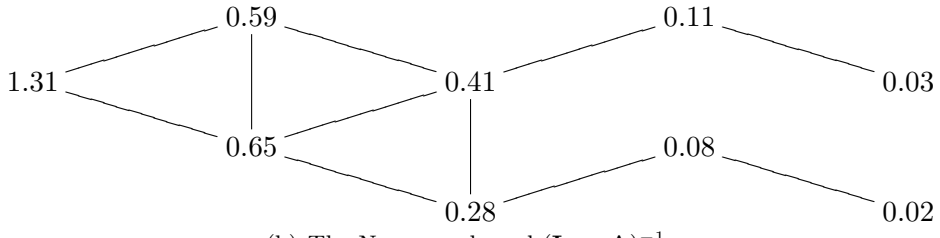
Other Graph Kernels Some graph kernels such as the commute-time kernel are the spectral transformation of other characteristic graph matrices, such as the graph Laplacian $\mathbf{L}$ or the normalized adjacency matrix $\mathbf{N}$. These kernels are however better suited for spectral clustering and less for link prediction [RLH09]. We will come back to Laplacian graph kernels in the next chapter, when applying them to signed graphs.

3.3.4 Rank Reduction

For large graphs, the eigenvalue and singular value decompositions can only be computed up to a small rank r, in practice not larger than about 100. The result is an approximation to the



(a) The exponential kernel $\exp(\alpha\mathbf{A})$



(b) The Neumann kernel $(\mathbf{I} - \alpha\mathbf{A})^{-1}$

Figure 3.6: The exponential and Neumann kernels applied to the small synthetic example network. The values are the similarity scores of each node to the leftmost node (node number 1). The parameters are $\alpha = 1$ for the exponential kernel and $\alpha = 0.25$ for the Neumann kernel.

adjacency matrix $\mathbf{A}$ that uses only the r dominant eigenpairs.

$$R_r(\mathbf{A}) = \mathbf{U}_{\bullet\,(1\ldots r)}\mathbf{\Lambda}_{(1\ldots r)\,(1\ldots r)}\mathbf{U}^{\mathrm{T}}_{\bullet\,(1\ldots r)} \tag{3.7}$$

where $\mathbf{U}_{\bullet\,(1\ldots r)}$ and $\mathbf{\Lambda}_{(1\ldots r)\,(1\ldots r)}$ are the corresponding eigenvector and eigenvalue matrices reduced to the largest r eigenvectors by absolute value, assuming that the diagonal elements $\lambda_k = \mathbf{\Lambda}_{kk}$ are in decreasing order by absolute value, i.e. that $|\lambda_1| \geq \ldots \geq |\lambda_n|$. This is the best rank-$r$ approximation of $\mathbf{A}$ in a least-squares sense.

Reduction to a rank r can be interpreted as the spectral transformation

$$f(\lambda) = \begin{cases} \lambda & \text{if } |\lambda| \geq |\lambda_r| \\ 0 & \text{otherwise} \end{cases}$$

A theoretical ambiguity arises when several eigenvalues have absolute value $|\lambda_r|$, which is usually ignored by ordering corresponding eigenvectors arbitrarily. In practice, the size of networks and the small value of r make that case very unlikely. The values r we used in experiments for the various datasets are given in Table A.2 in Appendix A. This r is chosen to yield a similar, reasonable runtime for each dataset. As a result, r is larger for smaller datasets.

Although rank reduction appears to be a necessary but unwanted approximation when doing link prediction using spectral methods, some studies have used rank reduction itself as the link prediction method, showing that a lower rank may result in more accurate link prediction [SKKR00, HZC07]. To see how this is possible, consider the entry (i, j) in the adjacency matrix $\mathbf{A}$, which is zero if i and j are not connected. In the matrix $R_r(\mathbf{A})$ however, this entry is not zero, and can be used as a link prediction score for the edge $\{i, j\}$. In methods such as latent Dirichlet allocation [BNJL03] and nonnegative matrix factorization [LS00] where a latent model is considered, rank reduction is also implicit. In Chapter 6, we will apply rank reduction to bipartite networks and show that it is equivalent to latent semantic analysis (LSA) for text datasets.

3.3.5 Latent Preferential Attachment

Preferential attachment is a simple link prediction model based on the idea that the probability of a new link being formed is proportional to the degrees of the nodes it connects. This idea can be extended to the decomposition of a graph's adjacency matrix, resulting in the latent preferential attachment model, which we describe here. In this section, we show that the latent preferential attachment model is equivalent to the spectral evolution model.

In a graph with adjacency matrix $\mathbf{A}$, the number of neighbors of a node i is called the degree of i and can be written as $d(i) = \sum_j \mathbf{A}_{ij}$. In the preferential attachment model, the probability of an edge appearing between the vertices i and j is proportional to both $d(i)$ and $d(j)$. In other words, links are formed with preference to nodes that already have a high number of neighbors. The preferential attachment model leads to networks were few nodes have many neighbors and many nodes have few neighbors. The distributions of node degrees in such networks can be described by *power laws*, meaning that the probability that a node has n neighbors is proportional to $n^{-\gamma}$ for a constant $\gamma > 1$. Typical values for γ are in the range $[2.0, 3.0]$, although many other values larger than 1.0 are observed in practice. Such networks are called scale-free networks [BA99] and, as we established in Section 2.2.2, most real-world networks are of this form.

Now let us consider the eigenvalue decomposition of the adjacency matrix $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$. This decomposition can be used to write $\mathbf{A}$ as a sum of rank-one matrices:

$$\mathbf{A} \approx \sum_{k=1}^{r} \lambda_k \mathbf{u}_k \mathbf{u}_k^{\mathrm{T}}$$

where r is the rank of the decomposition, $\lambda_k = \mathbf{\Lambda}_{kk}$ are the eigenvalues and $\mathbf{u}_k = \mathbf{U}_{\bullet k}$ are the eigenvectors of $\mathbf{A}$. The usual interpretation of a matrix factorization is that each latent dimension k represents a *topic* in the network. Then $\mathbf{U}_{ik}$ represents the importance of vertex i in topic k, and λ_k represents the overall importance of topic k. Each rank-one matrix $\mathbf{A}^{(k)} = \lambda_k \mathbf{u}_k \mathbf{u}_k^{\mathrm{T}}$ can now be interpreted as the adjacency matrix of a weighted graph. This graph G_k will be the complete graph on n vertices, since $\mathbf{u}_k$ is typically not sparse when the network is connected. Therefore, all information about this graph is contained in the weight of the edges. Now, assume that preferential attachment is happening in the network, but restricted to the subgraph G_k. Then the probability of the edge $\{i, j\}$ appearing will be proportional to $d_k(i)d_k(j)$, where $d_k(i)$ is the degree of node i in the graph G_k. This degree can be written as the sum over edge weights in G_k:

$$d_k(i) = \sum_l \mathbf{A}_{il}^{(k)} = \sum_l \lambda_k \mathbf{U}_{ik}\mathbf{U}_{lk} = \mathbf{U}_{ik}\lambda_k \sum_l \mathbf{U}_{lk} \sim \mathbf{U}_{ik}$$

In other words, $d_k(i)$ is proportional to $\mathbf{U}_{ik}$ only, since $\lambda_k \sum_l \mathbf{U}_{lk}$ is independent of i. Therefore, the preferential attachment value is proportional to the corresponding entry in $\mathbf{A}^{(k)}$:

$$d_k(i)d_k(j) \sim \mathbf{U}_{ik}\mathbf{U}_{jk}$$

These values can be aggregated into a matrix $\mathbf{P}^{(k)}$ giving the preferential attachment values for all pairs (i, j):

$$\mathbf{P}^{(k)} \sim \mathbf{u}_k \mathbf{u}_k^{\mathrm{T}}$$

If we now assume that preferential attachment is happening for each subgraph G_k separately, with a weight ε_k depending on the topic k, then the overall preferential attachment prediction

can be written as:

$$\mathbf{P} = \sum_k \varepsilon_k \mathbf{u}_k \mathbf{u}_k^{\mathrm{T}}$$

Here, we replace proportionality by equality since the proportionally constants are absorbed by the constants ε_k. The matrix $\mathbf{P}$ can then be written in the following form, giving its eigenvalue decomposition:

$$\mathbf{P} = \mathbf{U}\mathbf{E}\mathbf{U}^{\mathrm{T}}$$

Where $\mathbf{E}$ is the diagonal matrix containing the individual topic weights $\mathbf{E}_{kk} = \varepsilon_k$. This prediction matrix is a spectral transformation of the adjacency matrix $\mathbf{A}$. Under this model, network growth can be interpreted as the replacement of the eigenvalues $\mathbf{\Lambda}$ by $\mathbf{\Lambda} + \mathbf{E}$:

$$\mathbf{A} + \mathbf{P} = \mathbf{U}(\mathbf{\Lambda} + \mathbf{E})\mathbf{U}^{\mathrm{T}}$$

Since the values $\mathbf{E}$ are not modeled by the latent preferential attachment model, every spectral transformation can be interpreted as latent preferential attachment, and thus the latent preferential attachment model is equivalent to the spectral evolution model.

3.3.6 Irregular Growth

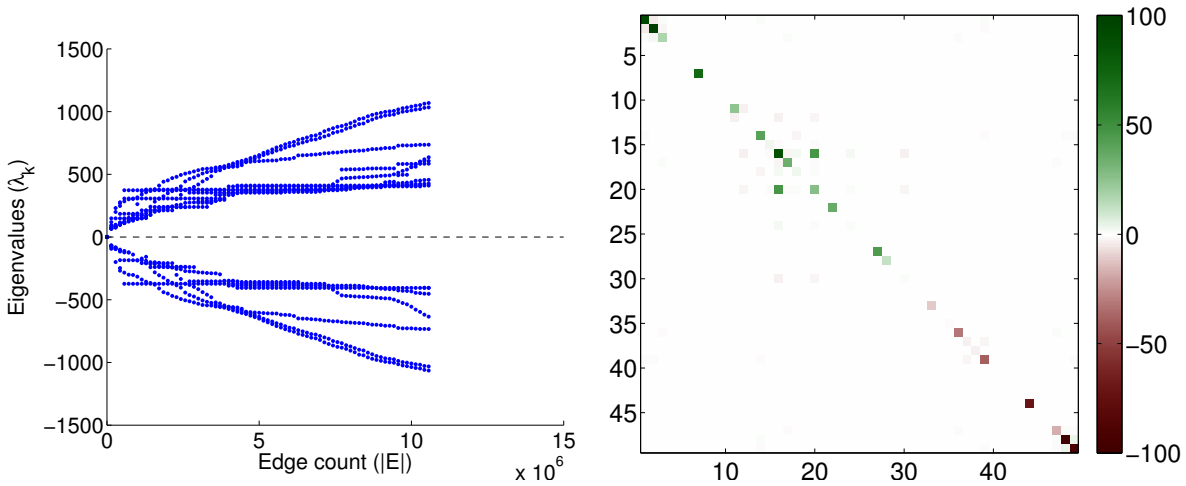
We have seen that the spectral evolution model is a generalization of several graph growth models such as graph kernels. If these existing graph growth models already describe the growth of graphs, why do we need to generalize them? The answer to that question is that most graph kernels and other growth models are only valid to a certain extent. By inspecting the growth of actual networks, we can see why these graph kernels cannot be universally applied for link prediction. Several cases of observed irregular spectral evolution are shown in Figure 3.7.

In these plots, we see that every eigenvalue grows at its own speed. As a result, the spectral transformation function $f(\lambda)$ that gives new eigenvalues as a function of old ones is not regular. This kind of spectral growth cannot be well described by a function f that is monotonous like a graph kernel. Instead, each eigenvalue is observed to grow at a specific speed. Despite this, the evolution of these networks is still spectral. It is only the spectral transformation that is irregular.

From this and other examples, we observe different possible behaviors of eigenvalue evolution. Some eigenvalues grow linearly, and others do not grow. In the same network, different eigenvalues may grow at different speeds. However, we did not encounter any case of an eigenvalue shrinking over time. This behavior may have many explanations, which are usually hard to guess. In most network datasets, nodes are not labeled, and so we cannot inspect the eigenvectors in question to see which vertices have the highest value in them. We may however conjecture that stagnating latent dimensions correspond to communities that die out, or other aspects of the network that do not grow anymore, for whatever reason. Although the reason for each eigenvalue's growth is unknown, it can still be learned, as we will do later in this chapter. In fact, these properties make link prediction algorithms based on spectral transformations more universal, since they do not rely on domain-specific knowledge of the network.

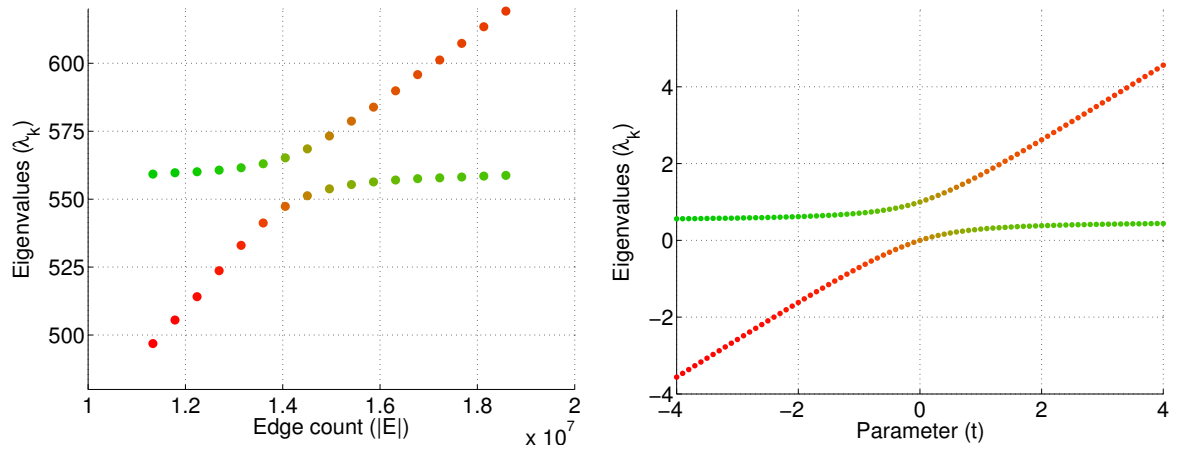
3.3.7 Avoided Crossings

As observed in several examples, an eigenvalue can overtake another. In these cases, our model predicts *crossings* in the spectral plot. Looking at actual spectra, we however observe something slightly different: The smaller, growing eigenvalue slows down growth before it reaches the larger



(a) Eigenvalue evolution (b) Spectral diagonality test

Figure 3.7: Example of irregular but spectral network evolution in the Twitter user–user "@name" mention network (Ws). (a) The evolution of the network's eigenvalues over time, (b) The spectral diagonality test applied to a 75%/25% split of the network. Although the spectral diagonality tests show that growth is spectral in this case, this spectral growth is irregular. The growth of this network can thus not be well described by any of the common graph kernels.



(a) Wikipedia links (WP) (b) Rank-one simulation

Figure 3.8: An avoided crossing as observed in the growth of the Wikipedia hyperlink network's (WP) two dominant eigenvalues and eigenvectors in (a), and in a rank-one model in (b). The red and green color components of the points denote the cosine distance of eigenvectors to initial eigenvectors.

eigenvalue, and the larger eigenvalue starts growing at about the same rate. We observe this behavior in the Wikipedia link network (WP), as shown in Figure 3.8(a).

This phenomenon is called an *avoided crossing* and can be explained as follows [Lax84, 8.5]. To simplify the example, we will assume that certain matrices have rank one, since we are only interested in individual eigenvalues. However, the example can be trivially extended to larger

ranks. Assume that at a time t_0, the adjacency matrix of a network is $\mathbf{A}$. Now, assume that the growth of the network around time t_0 can be described by a linear function. Then there is a matrix $\mathbf{B}$ such that at time t near t_0, the adjacency matrix of the network at time t is $\mathbf{A}_t = \mathbf{A} + (t - t_0)\mathbf{B}$. In the case that $\mathbf{A}$ and $\mathbf{B}$ have rank one, the eigenvalues of $\mathbf{A}_t$ display an avoided crossing behavior in most cases.

Let $\mathbf{A} = \alpha\mathbf{a}\mathbf{a}^{\mathrm{T}}$ and $\mathbf{B} = \beta\mathbf{b}\mathbf{b}^{\mathrm{T}}$. The matrices $\mathbf{A}$ and $\mathbf{B}$ have as only eigenvectors of nonzero eigenvalue $\mathbf{a}$ and $\mathbf{b}$, respectively. Then, $\mathbf{a}$ and $\mathbf{b}$ are also eigenvectors of $\mathbf{A}_t = \mathbf{A} + (t - t_0)\mathbf{B}$ when $\mathbf{a}$ and $\mathbf{b}$ are collinear or orthogonal. Otherwise, $\mathbf{A}_t$ has other eigenvectors and an avoided crossing is observed as in Figure 3.8(b). Thus, an avoided crossing indicates that a decomposition into outer products of orthogonal vectors is not the natural representation for some networks. If the true decomposition using $\mathbf{a}$ and $\mathbf{b}$ is used, the crossing would be *unavoided*. Alternatively, an avoided crossing may result from a perturbation of orthogonal eigenvectors in the same manner.

3.4 Control Tests

We have seen in the previous sections that spectral growth is observed in actual networks, and that it can be explained by several common graph growth models. To make sure that the validity of the spectral evolution model is in itself a nontrivial property of real networks, we have to verify that it does *not* follow from other, random processes. We will verify this for two fundamental random processes: The random addition of edges, and the random sampling of edges. To show that the spectral evolution model is not trivial, it must not be observed in these trivial models. Note that the property of nontriviality is not observed for all graph characteristics. Some advanced characteristics such as the modularity, a measure of the goodness of a node clustering, can be observed in the simplest of random graph models, the Erdős–Rényi model [GSPA04].

Another control tests concerns the choice of the characteristic graph matrix. Until now, we have analysed a graph's adjacency matrix. However, we could equally compute the eigenvalue decomposition of the Laplacian matrix, and ask whether its eigenvectors are stable. As we will see, this is not the case. The spectral evolution model is only observed for the adjacency matrix. This result does not make the Laplacian matrix useless. As we will see in the next chapter, the Laplacian matrix can be used to predict links, by learning a mapping from it to the adjacency matrix.

3.4.1 Random Graph Growth

The first random growth model we investigate consists of adding edges randomly to a given graph. Let $\mathbf{A}_t = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ be the adjacency matrix of a network, and $\mathbf{A}_{t+1} = \mathbf{A}_t + \mathbf{E}$ a perturbation of $\mathbf{A}_t$ where $\|\mathbf{E}\|_F = \varepsilon$ is small and $\mathbf{E}$ can be thought of as an edge of infinitesimal weight added to the network. The eigenvalue decomposition of $\mathbf{A}_{t+1} = \tilde{\mathbf{U}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{U}}^{\mathrm{T}}$ then has bounds of the following order:

$$\left\|\mathbf{\Lambda} - \tilde{\mathbf{\Lambda}}\right\|_{\mathrm{F}} = O(\varepsilon^2) \tag{3.8}$$

$$\left|\mathbf{U}_{\bullet k} \cdot \tilde{\mathbf{U}}_{\bullet k}\right| = O(\varepsilon) \tag{3.9}$$

These results can be shown by a perturbation argument [Ste90], and ultimately can be derived from theorems by Weyl [Wey12] and Wedin [Wed72]. As a result, eigenvectors are expected to change faster than eigenvalues for random additions to the adjacency matrix.

We verify these theoretical results experimentally using our collection of datasets. We use the following methodology: For each dataset, we consider the network at time t, where 75% of edges are present. Then, we add edges randomly until the network has many edges as the actual network at the final known time T. This gives a new timeline where the graph grows from 75% to 100% of edges. The actual evolution and the new, artificial evolution is then compared. The results are shown in Figure 3.9.

Observations The first two tests compare the actual and artificial evolution of the network's eigenvalue and eigenvectors, as done in Sections 3.2.1 and 3.2.2 on actual networks. We observe that in the artificial timeline, neither eigenvalues nor eigenvectors change significantly (plots 3.9(a) and 3.9(b)). The fact that eigenvalues do not change is explained by the fact that adding edges randomly will, over long times, only change the eigenvalue with eigenvector $\mathbf{1}$, i.e. the vector containing only ones. This means that the dominant eigenvectors are almost orthogonal to the constant vector, which is known to be true in high dimensions [Ham97]. The only exception is for the dominant eigenvector of unsigned networks, which is nonnegative due to the Perron-Frobenius theorem, and therefore has positive dot product with the vector $\mathbf{1}$. Additionally, we perform the spectral diagonality test of Section 3.2.4 in the plots 3.9(c) and 3.9(d). The spectral diagonality tests shows a non-diagonal matrix, implying that artificial growth is not spectral. We conclude that spectral growth of networks is not a consequence of random graph growth, but an inherent property of actual network growth.

3.4.2 Random Sampling

In this test, we generate an Erdős–Rényi random graph, and split its edges randomly into two sets. We then test how well a spectral transformation maps one edge set to the other.

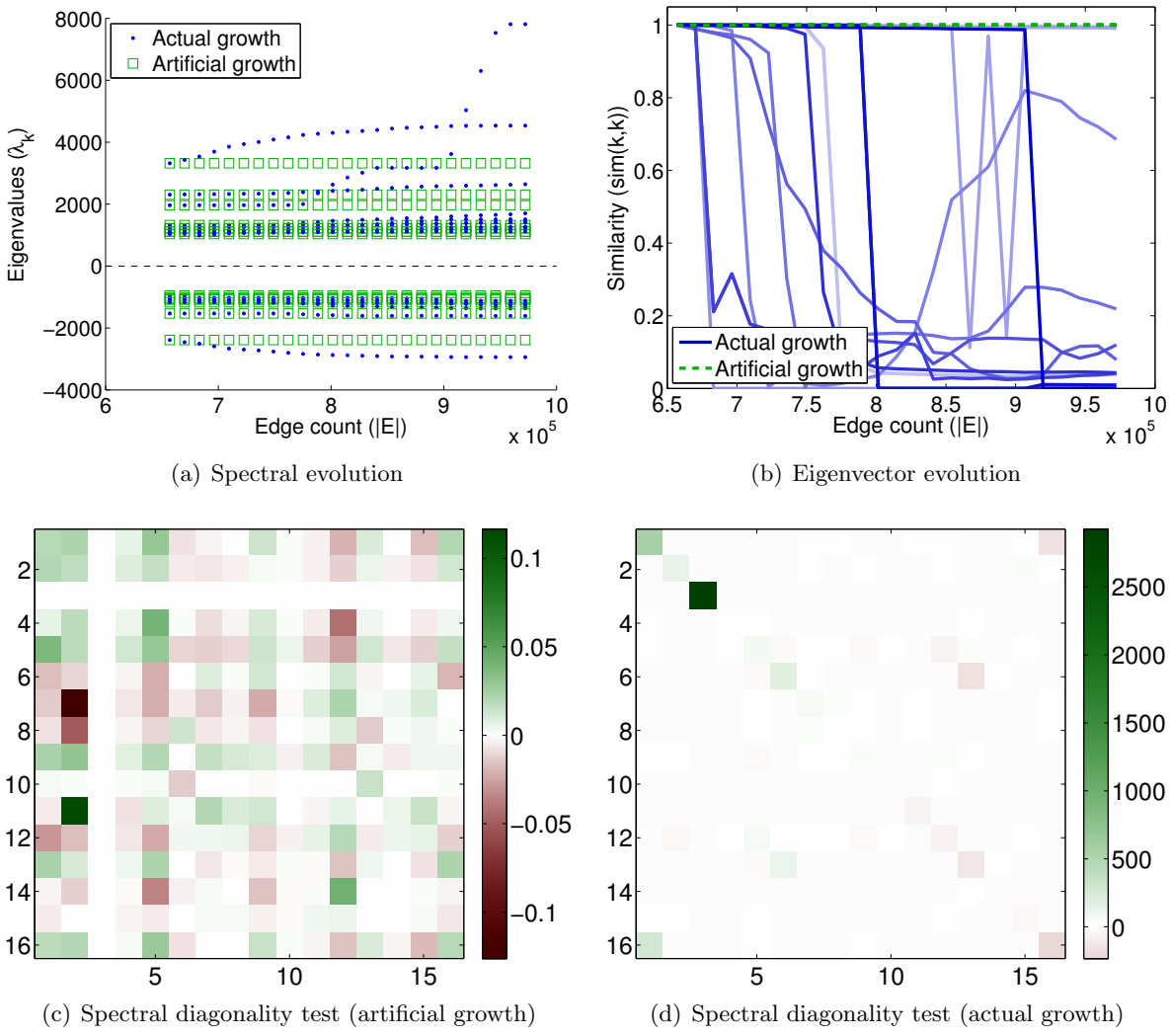
We use the following setup: Let $G = (V, E)$ be a randomly generated graph with $|V| = 1000$, and the probability of each possible existing edge is chosen thus that each vertex has on average three neighbors. In other words, each edge $\{i, j\}$ is present in the graph with a probability of $3/(|V| - 1)$. We make a 75%/25% random split into vertex sets E_a and E_b represented by the adjacency matrices $\mathbf{A}_\mathrm{a}$ and $\mathbf{A}_\mathrm{b}$. These proportions correspond to the training/test split used in the experiments later in this thesis. We then compute the reduced eigenvalue decomposition $\mathbf{A}_\mathrm{a} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathrm{T}$ of rank 100.

Figure 3.10 shows the spectral diagonality test of Section 3.2.4 applied to the random graph G and to the Facebook wall posts network (Ow). The plots show the matrix $\mathbf{U}^\mathrm{T}\mathbf{A}_\mathrm{b}\mathbf{U}$.

Observations In the spectral diagonality test matrix of the random graph (a), there is clearly only one eigenvalue with a significant nonzero value. This single eigenpair can be interpreted as a form of preferential attachment (see Section 3.3.5), and is explained by the fact that choosing edges at random will return edges adjacent to a node i with probability proportional to the degree of i. Apart from that, no structure of the original network is preserved spectrally. By contrast, the spectral diagonality test for the Facebook wall post network (b) is clearly diagonal. See also Figure 3.5 for more examples of spectral diagonality tests applied to actual networks. We conclude that a spectral transformation of rank greater than one is a nontrivial emergent property of actual network growth, and cannot be explained by the Erdős–Rényi random graph model.

3.4.3 Laplacian Spectral Evolution

We observed previously that the eigenvalue decomposition of a network's adjacency matrix $\mathbf{A}$ changes over time in a specific way: the eigenvalues grow, and the eigenvectors stay approxi-



(a) Spectral evolution

(b) Eigenvector evolution

(c) Spectral diagonality test (artificial growth)

(d) Spectral diagonality test (actual growth)

Figure 3.9: Comparing actual and artificial graph growth. These plots compare the actual growth of the Enron email dataset (EN) with a random graph growth model starting at the moment where 75% of edges are present in the dataset.

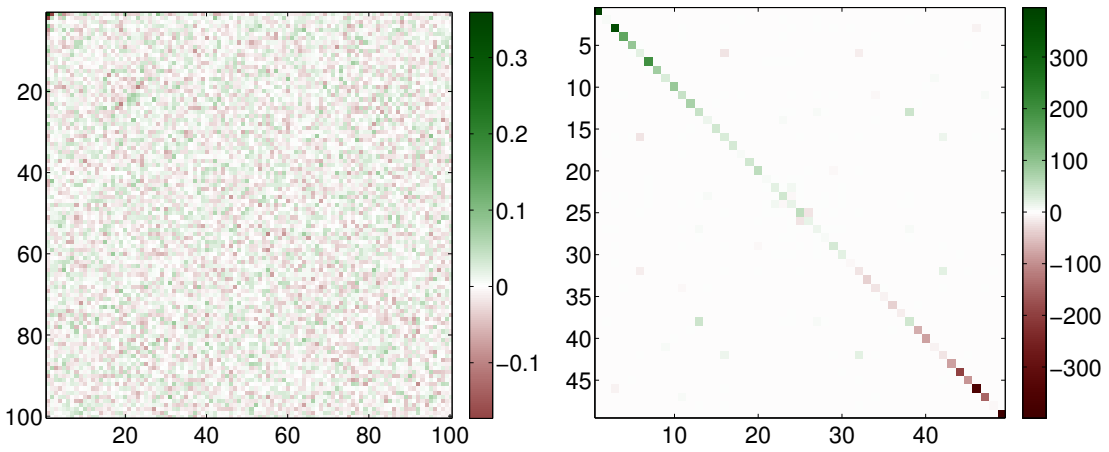
mately constant. Having defined other characteristic graph matrices such as the Laplacian $\mathbf{L}$ and the normalized adjacency matrix $\mathbf{N}$, we can ask the question whether the spectral evolution model can be applied to them.

Given an unweighted undirected graph $G = (V, E)$ with adjacency matrix $\mathbf{A}$, we recall that its normalized adjacency matrix $\mathbf{N}$ and its Laplacian $\mathbf{L}$ are given by

$$\mathbf{N} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$
$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

where $\mathbf{D}$ is the diagonal degree matrix in which $\mathbf{D}_{ii}$ is the degree of vertex i. Due to the relation $\mathbf{I} - \mathbf{N} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, the matrix $\mathbf{N}$ is a spectral transformation of the normalized Laplacian $\mathbf{Z} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. For this reason, what we will say about $\mathbf{N}$ will also be true for the normalized Laplacian $\mathbf{Z}$.

Figure 3.11 shows the spectral and eigenvector evolution of the matrices $\mathbf{N}$ and $\mathbf{L}$. In these plots, we consider the eigenvalue decompositions of the matrices $\mathbf{N}_t$ and $\mathbf{L}_t$ at time t, for all



(a) Random sampling (b) Facebook wall posts (Ow)

Figure 3.10: The spectral diagonality test of Section 3.2.4 applied to (a) a random sampling from an Erdős–Rényi graph and (b) the Facebook wall posts network (Ow). In these tests a diagonal matrix corresponds to a spectral transformation implying that only the Facebook network yields a spectral transformation.

times t:

$$\mathbf{N}_t = \mathbf{U}_t \mathbf{\Lambda}_t \mathbf{U}_t^{\mathrm{T}} \tag{3.10}$$

$$\mathbf{L}_t = \mathbf{U}_t \mathbf{\Lambda}_t \mathbf{U}_t^{\mathrm{T}} \tag{3.11}$$

For $\mathbf{N}$, we compute the top-r eigenvalues and corresponding eigenvectors. For $\mathbf{L}$, we compute the smallest r eigenvalues and corresponding eigenvectors. This is necessary because the matrix $\mathbf{L}$ is usually inverted when used (see e.g. Section 4.2.3). The value r depends on the dataset and is chosen to give reasonable runtimes. Values for r are listed in Appendix A. We show two kinds of plots: the spectral evolution as described in Section 3.2.1 and the eigenvector evolution as described in Section 3.2.2.

Observations We observe that for both $\mathbf{N}$ and $\mathbf{L}$, the eigenvalues are constant most of the time, and, rarely, they change abruptly. The corresponding eigenvectors are not stable, and also change abruptly to completely different values, indicated by a similarity of zero. We conclude that the spectral evolution of $\mathbf{N}$ and $\mathbf{L}$ is not smooth, and that the spectral evolution model does not apply to them. Even though the matrices $\mathbf{N}$ and $\mathbf{L}$ do not change smoothly, they can be used for link prediction, by applying graph kernels to them. This will be explained in Section 4.2.

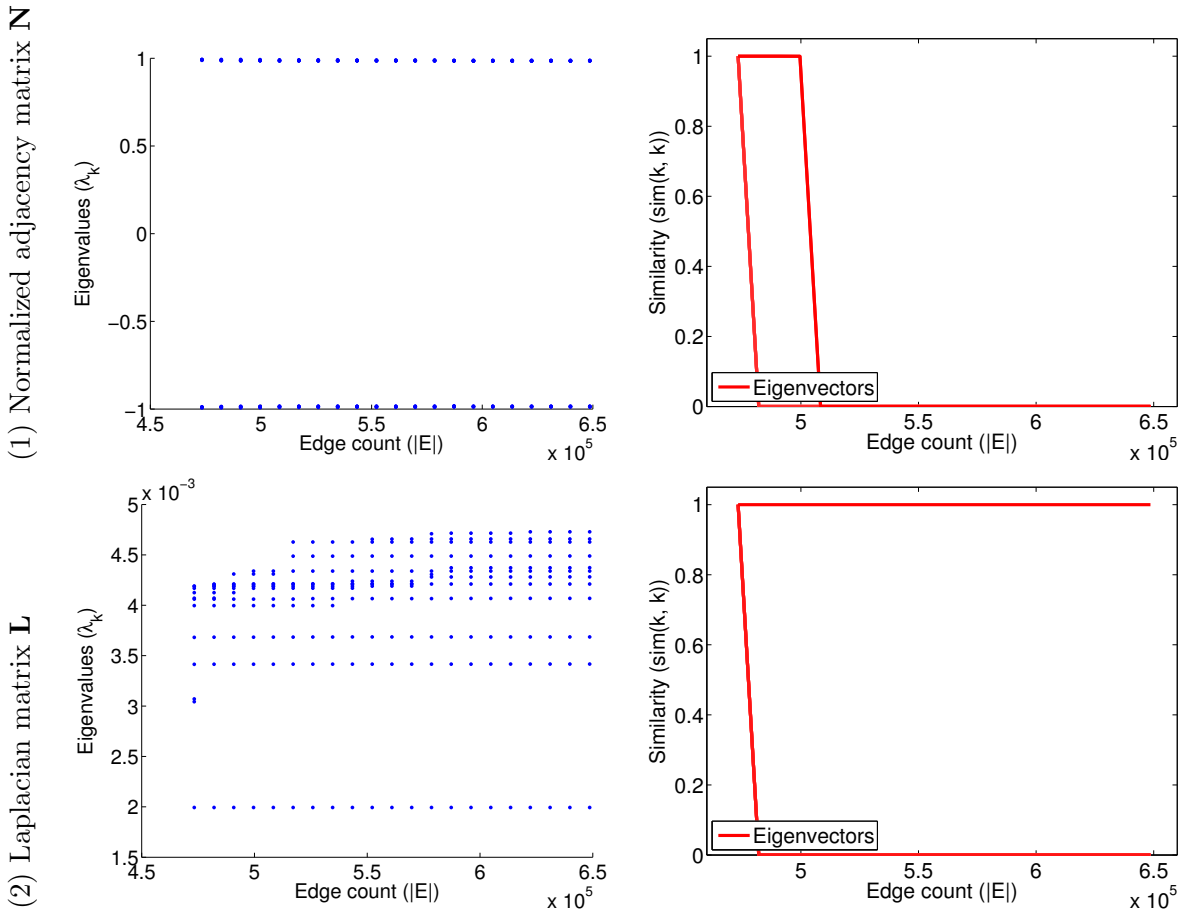
There is also an algebraic reason why the normalized adjacency and Laplacian matrices do not grow spectrally. Consider the Laplacian $\mathbf{L}$ of an unweighted, undirected graph $G = (V, E)$. Now consider the graph $G' = (V, E \dot\cup \{i, j\})$, i.e. the graph G to which an edge $\{i, j\}$ has been added. The Laplacian $\mathbf{L}'$ of G' is given by

$$\mathbf{L}' = \mathbf{L} + \mathbf{E},$$

where $\mathbf{E}$ is the matrix defined by $\mathbf{E}_{ij} = \mathbf{E}_{ji} = -1$, $\mathbf{E}_{ii} = \mathbf{E}_{jj} = 1$ and all other entries of $\mathbf{E}$ are zero. The matrix $\mathbf{E}$ can be written as

$$\mathbf{E} = \mathbf{x}\mathbf{x}^{\mathrm{T}}$$

where the vector $\mathbf{x}$ is defined by $\mathbf{x}_i = +1$, $\mathbf{x}_j = -1$ and $\mathbf{x}_k = 0$ for $k \neq i, j$. Due to a well-known theorem (see e.g. [Wil65, p. 97]), adding $\alpha \mathbf{x}\mathbf{x}^{\mathrm{T}}$ to a symmetric matrix will have the following effect on the spectrum of the matrix: If $\alpha > 0$, then the eigenvalues can only increase; if $\alpha < 0$, the eigenvalues can only decrease. Therefore, going from $\mathbf{L}$ to $\mathbf{L}'$ can only increase the eigenvalues, not decrease them. However, known link prediction methods using the Laplacian as will be described in Section 4.2.3 increase the inverse of the Laplacian eigenvalues and thus would decrease the smallest Laplacian eigenvalue. Therefore, these kernels are not consistent with a spectral evolution of the Laplacian spectrum. The same argument is valid for $\mathbf{Z}$, making a spectral evolution of $\mathbf{N} = \mathbf{I} - \mathbf{Z}$ unrealistic.



(a) Eigenvalue evolution (b) Eigenvector evolution

Figure 3.11: The spectral evolution in the EU institution email network (EU). (1) the normalized adjacency matrix $\mathbf{N}$, (2) the Laplacian matrix $\mathbf{L}$. Two experiments are shown: (a) the spectral evolution as defined in Section 3.2.1, (b) the eigenvector evolution as defined in Section 3.2.2. In both cases, we observe that the largest eigenvalues of $\mathbf{N}$ and smallest eigenvalues of $\mathbf{L}$ stay constant.

3.5 Summary: The Spectral Evolution Model

The spectral evolution model states that in many real-world networks, growth can be described by a change of the spectrum, while the corresponding eigenvectors remain constant. This assumption is true to a large extent in most networks we analysed, based on the observation

of the change in eigenvalues and eigenvectors of networks over time. A more sophisticated test of the spectral evolution model is what we have called the spectral diagonality test. This test measures to what extent a snapshot of the network at one point can be diagonalized by the eigenvectors of the same network at another time. The result is a matrix that is diagonal exactly when growth is perfectly spectral. When growth is not spectral, this matrix is not diagonal. The degree to which this matrix is diagonal can therefore be used as an indicator of the spectrality of a network's growth. This matrix will be used again in the context of link prediction in the next chapter, where we will use it to learn a spectral transformation.

The fact that the spectral evolution model can be observed in many networks is not in contradiction to previous results in the literature. In fact, a certain number of known link prediction methods result in network growth that can be described by spectral transformations. This class of link prediction functions includes the triangle closing model, the weighted path counting model, graph kernels based on adjacency matrix and rank reduction approaches. Another explanation of the spectral evolution model is given by an extension to the preferential attachment model, which we call the latent preferential attachment model. As we showed, the latent preferential attachment model is equivalent to the spectral evolution model, under the assumption that a preferential attachment process happens for each eigenpair separately.

To make sure that spectral evolution is a characteristic of real networks and not an artifact of sampling any network randomly, we performed two control tests. By adding edges randomly to an existing network and by sampling a random network, we could not observe any spectral growth. This indicates that the spectral evolution model is indeed a nontrivial property of real-world networks.

Having confirmed that the spectral evolution model applies to many real networks, we can now exploit it to predict links in these networks. In the next chapter, we will thus present two new link prediction methods based on the spectral evolution model.

Chapter 4

Learning Spectral Transformations

In the last chapter, we have observed network growth to follow the *spectral evolution model*. The spectral evolution model that we introduced explained the growth of networks using the eigenvalue decomposition of their adjacency matrix, stating that only the eigenvalues of a network change over time, while their eigenvectors stay constant. In this chapter, this observation is exploited to solve the problem of link prediction: Since only the eigenvalues of a network change over time and not the eigenvectors, it suffices to predict new values for the eigenvalues. We introduce two different methods for doing this. One uses curve fitting to learn a function mapping old to new eigenvalues; the other uses extrapolation of eigenvalues.

The link prediction problem is fundamental in that it applies to virtually all types of networks, and can be used to implement recommender systems and similar applications. In a social network for instance, the task of recommending new friends can be viewed, from the point of view of the recommender system, as the problem of predicting which edges will appear in the network's future evolution, and recommending them immediately. In the physical network of connections between Internet hosts, the prediction of new connections is crucial for planning and optimization. Similar examples exist for all networks in our collection, and justify the choice of the link prediction problem as the main application of the spectral evolution model in this thesis.

The main idea for link prediction under the spectral evolution model consists in predicting future values of a network's eigenvalues, and retaining its eigenvectors. The first method we present takes the approach that graph kernels are correct descriptions of network growth, and learns the parameters of these graph kernels using curve fitting. The second method assumes that the growth of eigenvalues does not follow any specific pattern, and therefore known link prediction functions cannot give accurate link prediction results. Instead, the second method extrapolates the change of eigenvalues into the future for each eigenvalue separately. This chapter describes both methods for unweighted, undirected, unipartite networks. The case of signed, directed and bipartite networks is described in the two following chapters.

We begin the chapter in Section 4.1 by introducing the link prediction problem formally. Then, the two new spectral link prediction methods based on curve fitting and spectral extrapolation are described in Sections 4.2 and 4.3. Experimental results for both methods are given in Section 4.4, along with a discussion of the relative performance of the different graph kernels and of the different characteristic graph matrices. Finally, Section 4.5 reviews similar but unrelated methods used in other areas of data mining on graphs.

4.1 Link Prediction

An important class of applications using networks are recommender systems. In recommender systems, the task can usually be formulated as a problem of link prediction. In a general sense, *link prediction* denotes the problem of predicting links in a network. In a broad sense, link prediction is a very general type of problem that can be formulated on networks, and is typically hard to solve. As examples, recommender systems in the scope of this work are found in many data mining applications:

- A search engine finds documents matching a query. By modeling documents and the words they contain as a bipartite network, matching documents to a query corresponds to finding highly-scored links between the query words and documents.
- A recommender system can be modeled as a network containing both users and items. Recommendations are then found by predicting links from users to items. The recommender network may connect users and items directly such as with ratings, or indirectly, e.g. through topics, categories, user history, sensors, etc.
- Context-aware recommender systems additionally include a context for each query, which can be modeled as links between the query and the contextual elements. The task is then to find links between the query and entities to recommend.
- Rating prediction is a special case of link prediction, where edges are weighted. An important application is collaborative filtering, with users rating items.
- Finding related items can be achieved by predicting links between items connected to a network, even if there are no direct links between the items.
- To recommend new friends in a social network, some recommender systems must find similar nodes in the network. In this case edges are unweighted, and the links to be predicted describe the similarity between nodes.
- To predict whether a user will like a given movie or not, a recommender system must predict edge weights. In this case the network is a bipartite rating graph, and missing links in the network have to be predicted.
- To predict future interactions, for instance emails or scientific coauthorship, link prediction must be performed in a network with multiple edges.

For all these problems, there is a network optionally weighted by real numbers, and the problem consists of predicting future edges. In particular, we can distinguish the following types of link prediction:

- Given a network, predict which edges will appear (link detection or link completion [GKK03]).
- Given a network and a specific node, predict to which other nodes it will connect (recommendation [HCC04]).
- Given a network and a pair of unconnected nodes, determine whether it will be connected (link prediction proper [LNK03]).
- Given a weighted network and two unconnected nodes, determine the weight of an edge connecting them, knowing that an edge will appear between them (collaborative filtering [BHK98], predicting trust [GKRT04], link sign prediction [KLB09]).

These problems can be solved by considering link prediction functions: functions of node pairs returning a numerical score. The different machine learning problems given above can then be distinguished by the way these numbers are interpreted, and how they are compared to the actual networks. Computed scores for a node pair can be used as prediction values directly, or be used to find a ranking, which is then compared with the actual known ranking.

4.1.1 Local Link Prediction Methods

This section describes several simple link prediction methods that do not make use of spectral graph theory. Due to their simplicity, these methods are very widely used, and will serve as a baseline against which complex link prediction methods are evaluated. These link prediction functions compute a link prediction score for a node pair using only information about the immediate neighborhood of the two nodes. We will therefore call these functions local *link prediction functions*. All these link prediction functions can be found in [LNK03].

Let i and j be two nodes in the graph for which a link prediction score is to be computed. To compute a link prediction score for the edge $\{i, j\}$, local link prediction functions depend only on the neighbors of i and j. In contrast to this, the spectral link prediction methods described in the main part of this work take into account the whole network. In other words, they are global.

Common Neighbors The number of common neighbors can be used in itself as a link prediction function [LNK03]. In the example of social recommendations in a user-user network, this implements the *friend of a friend* principle, and is equivalent to recommending non-friends that have the highest number of common friends:

$$p_{\mathrm{CN}}(i, j) = |\{k \mid k \sim i \wedge k \sim j\}| \tag{4.1}$$

The number of common neighbors is a spectral transformation. It is equivalent to the triangle closing model described in Section 3.3.1.

Preferential Attachment Taking only the degree of i and j into account for link prediction leads to the *preferential attachment* model [BA99], which can be used as a model for more complex methods such as modularity kernels [ZM08, New06a]. If $d(i)$ is the number of neighbors of node i, the preferential attachment model gives a prediction for an edge between i and j of

$$p_{\mathrm{PA}}(i, j) = \frac{1}{2|E|} d(i)d(j). \tag{4.2}$$

The factor $1/(2|E|)$ normalizes the sum of predictions for a vertex to its degree. As we saw in 3.3.5, a generalization of the preferential attachment model is equivalent to the spectral evolution model.

Jaccard The Jaccard coefficient measures the amount of common neighbors divided by the number of neighbors of either vertex [LNK03]:

$$p_{\mathrm{JAC}}(i, j) = \frac{|\{k \mid k \sim i \wedge k \sim j\}|}{|\{k \mid k \sim i \vee k \sim j\}|} \tag{4.3}$$

The Jaccard coefficient too can be considered a weighted variant of the common neighbors model. In networks with multiple edges, the Jaccard coefficient can be extended by using multisets in the definition, containing a node k multiple times when it is connected to i or j multiple times.

Adamic–Adar The measure of Adamic and Adar [AA01] counts the number of neighbors, weighted by the inverse logarithm of each neighbor k's degree $d(k)$:

$$p_{\mathrm{AA}}(i,j) = \sum_{k \sim i \wedge k \sim j} \frac{1}{\log(d(k))} \tag{4.4}$$

This can be interpreted as a weighted variant of the common neighbors model.

4.1.2 Normalization

Both local and global link prediction algorithms usually benefit from normalization. Normalization consists of applying a transformation to the data before applying a link prediction algorithm. A typical example of normalization is performed for rating prediction: The overall mean rating is subtracted from all ratings. After predictions have been computed, the overall mean is then added to all predicted ratings. We will call this type of procedure *additive normalization*.

In Section 2.1.5, we showed that the adjacency matrix $\mathbf{A}$ can be replaced by $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ to alleviate the effects of skewed degree distributions. This amounts to replacing each entry $\mathbf{A}_{ij} = 1$ by $1/\sqrt{d(i)d(j)}$. We will call this procedure *multiplicative normalization*. Both types of normalization can be applied to almost all link prediction algorithms. In fact, they can even be applied to the trivial link prediction algorithm that always predicts 1 or to the trivial rating prediction algorithm that always predicts 0:

- Always predicting 1 in combination with multiplicative normalization leads to the preferential attachment model: Predict $d(i)d(j)$ for the edge $\{i,j\}$.
- Always predicting 0 in combination with additive normalization leads to the global mean prediction: Always predict the global mean rating.

In this work, additive normalization is always used implicitly for all weighted networks. Multiplicative normalization is explicitly used by using the normalized matrices $\mathbf{N}$ and $\mathbf{M}$ instead of $\mathbf{A}$ and $\mathbf{B}$.

4.1.3 Evaluating Link Prediction Methods

To evaluate a link prediction algorithm, we must know which links actually appeared in a network and compare them to the links that were predicted. In networks where we know the time at which each edge has appeared, we can split the set of edges by creation time. We can then apply a given link prediction algorithm on the training set, use it to predict links, and compare the predicted links with the test set.

Each of the link prediction algorithms that we introduce in this work is based on a spectral transformation. This spectral transformation is specific to each dataset, and therefore it has to be learned separately for each dataset. Thus, we split the training set again by edge creation time, and use this split to learn a spectral transformation, which is then applied to the whole test set. Finally, the predicted edges are compared with the edges in the test set. Figure 4.1 gives a summary of the method we use.

The split is performed in the following way. First, the training/test split is made, using 75% of all edges in the training set and 25% of all edges in the test set. This split is done by edge creation times: Every edge in the training set is older than every edge in the test set. At the time where the split occurs, the network contains all edges in the training set, and nothing more. The training set is then split into a source and target set, also of sizes 75% and 25%.

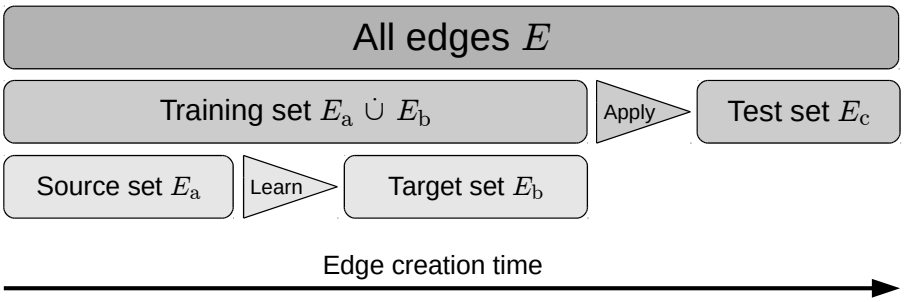


Figure 4.1: The three-way split used in all experiments. The dataset is first split into a training and test set by edge creation time. Then, the training set is a again split into a source and target set of edges. Then, a spectral transformation is learned from the source to the target set. This spectral transformation is then applied to the training set. Finally, the predicted edges are compared with the edges in the test set.

Then, the giant connected component is found in the source set, and all vertices not in the giant component are removed in the source, target and test sets. The rationale here is the following: If an edge in the test or target set connects two vertices that are not connected by a path in the source set, then there is no point in trying to learn this edge. By only keeping the giant connected component in the source set, we make sure that edges in the test and target sets are indirectly connected in the source set. This pruning however results in splits that are not exactly of size 75%/25% for all datasets. If a dataset does not have a connected component in the source set of at least 10% of all edges, then we exclude the dataset from our collection.

We use the following notation to denote the source, target and test sets. In the graph $G = (V, E)$, the set of edges is partitioned as $E = E_\mathrm{a} \dot\cup E_\mathrm{b} \dot\cup E_\mathrm{c}$, where E_a is the source set, E_b is the target set and E_c is the test set. Together, E_a and E_b are the training set. The adjacency matrix $\mathbf{A}$ is then decomposed as $\mathbf{A} = \mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{c}$, where the indexes $\mathrm{a}, \mathrm{b}, \mathrm{c}$ represent the source, target and test set respectively.

4.1.4 Measuring Link Prediction Accuracy

To compare the experimental results of link prediction algorithms, a measure of link prediction accuracy has to be used. Several such measures are used in the literature, of which we only mention the mean average precision (MAP) [NZT07], the area under the curve (AUC) [Bra97] and the normalized discounted cumulated gain (NDCG) [JK02]. These measures of accuracy are similar to each other, and our tests indicate that they are interchangeable for our purpose. Therefore, we will only report prediction accuracy using a single one of these measures, the mean average precision, which we now define.

The mean average precision is based on the ranking implied by link prediction scores. In other words, given predicted scores for all edges, we find the actual edges among them and then their ranking determines a mean average precision value. Since the total number of possible edges is quadratic in the vertex count, this cannot be computed in practice. Instead, we generate a *zero test set*, i.e. a set of edges of the same size as the test set, containing only edges not contained in the original networks. We call this zero test set $E_{\bar{c}}$.

Let $E_{\mathrm{c}\bar{\mathrm{c}}} = E_\mathrm{c} \dot\cup E_{\bar{\mathrm{c}}}$ be the total test set of edges with $|E_{\mathrm{c}\bar{\mathrm{c}}}| = s$. For each test edge $e \in E_{\mathrm{c}\bar{\mathrm{c}}}$, $\mathbf{A}_e = (\mathbf{A}_\mathrm{c})_e$ is the actual edge weight in the test set, and p_e the computed link prediction value. The average precision is then defined in terms of rankings: Edges are ranked by their predicted weight, and evaluated by their actual weight. For each edge in E_c, we compute the precision of the results up to that edge. The average of these precision values then gives the average

precision. The average precision is therefore in the range $[0, 1]$, and the value 1 denotes perfect prediction.

Let $R(k)$ be the edge of rank k defined such that $p_{R(k)} \leq p_{R(l)}$ if $k < l$. Then the average precision is defined as

$$\operatorname{ap}(p) = \left(\sum_{k=1}^{s} \left[\mathbf{A}_{R(k)} \right]_0^1 \right)^{-1} \sum_{k=1}^{s} \left(\left[\mathbf{A}_{R(k)} \right]_0^1 \frac{1}{k} \sum_{l=1}^{k} \left[\mathbf{A}_{R(l)} \right]_0^1 \right), \tag{4.5}$$

where $[x]_0^1 = 1$ if $x > 1/2$ and $[x]_0^1 = 0$ otherwise.

The mean average precision is then given by the mean of average precision values computed over each node separately [MRS08]. Let $\operatorname{ap}_i(p)$ be the average precision computed only over all edges adjacent to vertex i. Then the mean average precision is given by

$$\operatorname{map}(p) = \frac{1}{|V|} \sum_{i \in V} \operatorname{ap}_i(p). \tag{4.6}$$

In information retrieval, the average precision is averaged over all queries. Here, we consider each vertex in the test set a query, and consider the prediction of its incident edges a query. Both definitions are equivalent.

4.2 Learning by Curve Fitting

We now describe a general learning technique for estimating the parameters of any graph kernel or other link prediction function that can be expressed as a spectral transformation. This method works by reducing the learning problem to a one-dimensional curve-fitting problem. As mentioned in Section 3.3, a certain number of graph kernels and other link prediction methods can be expressed as spectral transformations, confirming the spectral evolution model. Our contribution in this section is a method for learning the parameters of these link prediction functions by reducing the resulting high-dimensional parameter learning problem to a one-dimensional curve fitting problem, which can be solved efficiently. The runtime of this method only depends on the chosen reduced rank, and is independent of the original network size.

We show that this generalization is possible under the assumption that the chosen training and test sets are simultaneously diagonalizable, an assumption which, as described in Section 3.3, is implicit in the spectral evolution model. The method presented in this section can be used to learn the parameters of these kinds of link prediction functions. This includes not only the parameters of graph kernels such as the matrix exponential, but also the reduced rank in rank reduction methods and weights used for individual path lengths in path-counting methods. Since we reduce the parameter estimation problem to a one-dimensional curve fitting problem that can be plotted and inspected visually to compare the different prediction algorithms, an informed choice can be made about them without having to evaluate each algorithm on a test set separately.

Let $\mathbf{A}$ be the adjacency matrix of an undirected, unweighted, unipartite graph, and $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ its eigenvalue decomposition. Then the spectral evolution model states that new links can be predicted by computing a spectral transformation $F(\mathbf{A}) = \mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^{\mathrm{T}}$. As described in Section 3.3, several link prediction functions are of this form, in particular all graph kernels based on the adjacency matrix such as the matrix exponential $F(\mathbf{A}) = \exp(\alpha\mathbf{A})$. Most of these graph kernels have parameters, for instance α for the matrix exponential. This section will describe a way to learn these parameters, and at the same time give a simple method for comparing different graph kernels in more depth than simply comparing their performances at link prediction.

Other graph kernels than those presented in the last chapter are not spectral transformations of the adjacency matrix $\mathbf{A}$, but of the Laplacian matrix $\mathbf{L}$, or the normalized adjacency matrix $\mathbf{N}$. The method described in this section can also be applied to these graph kernels, as we will show.

4.2.1 Link Prediction as an Optimization Problem

Given a graph $G = (V, E)$, we want to find a spectral transformation F that performs well at link prediction for this particular graph. To that end, we divide the set of edges E into a training set E_a and a test set E_b, and then look for a function F that maps the training set to the test set with minimal error.

Formally, let $\mathbf{A}_\mathrm{a}$ and $\mathbf{A}_\mathrm{b}$ be the adjacency matrices of the source and target set respectively, such that $\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{b}$ is the complete known adjacency matrix of G. In other words, we know $\mathbf{A}_\mathrm{a}$ and want to find a function that maps $\mathbf{A}_\mathrm{a}$ to $\mathbf{A}_\mathrm{b}$. We will call $\mathbf{A}_\mathrm{a}$ the source matrix and $\mathbf{A}_\mathrm{b}$ the target matrix. Let $\mathcal{S}$ be the set of all spectral transformations from $\mathbb{R}^{|V|\times|V|}$ to $\mathbb{R}^{|V|\times|V|}$. The solution to the following optimization problem then gives the optimal spectral transformation for the task of predicting the edges in the test set.

Problem 1. *Let $\mathbf{A}_\mathrm{a}, \mathbf{A}_\mathrm{b} \in \mathbb{R}^{|V|\times|V|}$ be two symmetric adjacency matrices over the same vertex set V. A spectral transformation that maps $\mathbf{A}_\mathrm{a}$ to $\mathbf{A}_\mathrm{b}$ with minimal error is given by a solution to*

$$\begin{aligned} \min_{F} \quad & \left\| F(\mathbf{A}_\mathrm{a}) - \mathbf{A}_\mathrm{b} \right\|_\mathrm{F} \\ \text{s.t.} \quad & F \in \mathcal{S} \end{aligned} \tag{4.7}$$

The Frobenius norm $\|\cdot\|_\mathrm{F}$ corresponds to the root mean squared error (RMSE) of the mapping from $\mathbf{A}_\mathrm{a}$ to $\mathbf{A}_\mathrm{b}$. While the root mean squared error is common in link prediction problems, other error measures exist, but give more complex solutions to our problem, making it impossible to solve the resulting problem efficiently. We will therefore restrict ourselves to the Frobenius norm in the rest of the section.

Problem 1 can be solved by computing the eigenvalue decomposition $\mathbf{A}_\mathrm{a} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathrm{T}$ and using the fact that the Frobenius norm is invariant under multiplication by an orthogonal matrix.

$$\begin{aligned} & \left\| F(\mathbf{A}_\mathrm{a}) - \mathbf{A}_\mathrm{b} \right\|_\mathrm{F} \\ =& \left\| \mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^\mathrm{T} - \mathbf{A}_\mathrm{b} \right\|_\mathrm{F} \\ =& \left\| F(\mathbf{\Lambda}) - \mathbf{U}^\mathrm{T}\mathbf{A}_\mathrm{b}\mathbf{U} \right\|_\mathrm{F} \end{aligned} \tag{4.8}$$

The Frobenius norm in Expression (4.8) can be decomposed into the sum of squares of off-diagonal entries of $F(\mathbf{\Lambda}) - \mathbf{U}^\mathrm{T}\mathbf{A}_\mathrm{b}\mathbf{U}$, which is independent of F, and into the sum of squares of its diagonal entries. This leads to the following least-squares problem equivalent to Problem 1:

Problem 2. *If $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathrm{T}$ is the eigenvalue decomposition of $\mathbf{A}_\mathrm{a}$, then the solution to Problem 1 is given by $F(\mathbf{\Lambda})_{kk} = f(\mathbf{\Lambda}_{kk})$, where $f(\lambda)$ is a solution to the following minimization problem.*

$$\min_{f} \quad \sum_{k} \left(f(\mathbf{\Lambda}_{kk}) - \mathbf{U}_{\bullet k}^\mathrm{T}\mathbf{A}_\mathrm{b}\mathbf{U}_{\bullet k} \right)^2 \tag{4.9}$$

This is a one-dimensional least-squares curve fitting problem of size n. Since each function $F(\mathbf{A}_\mathrm{a})$ corresponds to a function $f(\lambda)$, we can choose a link prediction function F and learn its parameters by inspecting the corresponding curve fitting problem. As an example, Figure 4.2 shows a plot of $\lambda_k = \mathbf{\Lambda}_{kk}$ against $f(\lambda_k) = \mathbf{U}_{\bullet k}^\mathrm{T}\mathbf{A}_\mathrm{b}\mathbf{U}_{\bullet k}$ for the University Rovira i Virgili email dataset (A@).

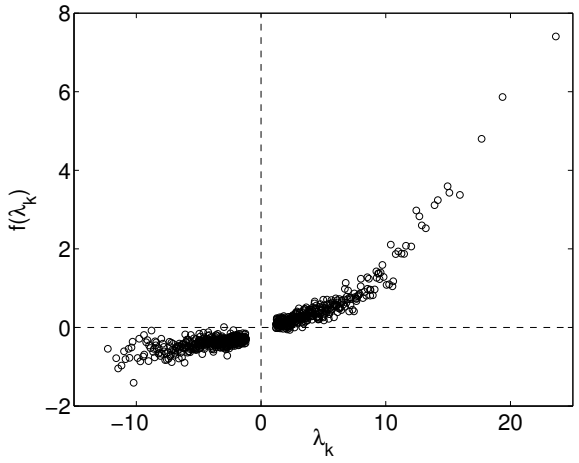


Figure 4.2: The one-dimensional least-squares problem used to learn spectral transformations by curve fitting. The values $\lambda_k = \mathbf{\Lambda}_{kk}$ plotted against $f(\lambda_k) = \mathbf{U}_{\bullet k}^{\mathrm{T}} \mathbf{A}_{\mathrm{b}} \mathbf{U}_{\bullet k}$ for the University Rovira i Virgili email dataset (A@).

4.2.2 Curve Fitting

We have reduced the general matrix regression problem (4.7) to a one-dimensional least-squares curve fitting problem of size n. In practice, computing the full eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ is not possible, first because the decomposition has runtime $O(n^3)$, but also because the dense $n \times n$ matrix $\mathbf{U}$ would take too much memory to store, i.e. $O(n^2)$. Instead, the rank-reduced eigenvalue decomposition $\mathbf{A} \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{U}}^{\mathrm{T}}$ of rank r can be computed. The result is that in Equation (4.9), the curve fitting problem goes only over r points. Since r is usually much smaller than n (see Table A.2 in Appendix A for values), this curve fitting problem is computationally very easy to solve.

We now take each spectral link prediction function $F(\mathbf{A})$ described in Section 3.3 and, for each one, derive its underlying real function $f(\lambda)$ that applies to every eigenvalue separately. Most of the link prediction functions have parameters, e.g. the parameter α in the exponential kernel $\exp(\alpha\mathbf{A})$. These parameters are kept in the real function. Additionally, we insert a multiplicative parameter $\beta > 0$ into every real function $f(\lambda)$, giving $\beta f(\lambda)$, that is to be learned along with the other parameters. Although this parameter will not change the relative link prediction scores and therefore has no influence on the final link prediction accuracy, including it allows the learned curves to better fit the observed eigenvalues, and thus results in more sensible values for the other parameters. The resulting functions are summarized in Table 4.1.

Table 4.1: Link prediction functions based on the adjacency matrix and their corresponding real functions. The outer multiplicative coefficient β (see explanation in the text) is not shown, for clarity.

Method	**Matrix function**	**Real function**	
Triangle closing	$\mathbf{A}^2$	$f(\lambda) = \lambda^2$	Eq. 3.3
Path counting	$p(\mathbf{A}) = \sum_{k=0}^{\infty} \alpha_k \mathbf{A}^k$	$f(\lambda) = \sum_{k=0}^{\infty} \alpha_k \lambda^k$	Eq. 3.4
Exponential kernel	$K_{\mathrm{EXP}}(\mathbf{A}) = \exp(\alpha\mathbf{A})$	$f(\lambda) = e^{\alpha\lambda}$	Eq. 3.5
Neumann kernel	$K_{\mathrm{NEU}}(\mathbf{A}) = (\mathbf{I} - \alpha\mathbf{A})^{-1}$	$f(\lambda) = 1/(1 - \alpha\lambda)$	Eq. 3.6
Rank reduction	$R_r(\mathbf{A}) = \mathbf{U}_{\bullet\,(1\ldots r)} \mathbf{\Lambda}_{(1\ldots r)\,(1\ldots r)} \mathbf{U}_{\bullet\,(1\ldots r)}^{\mathrm{T}}$	$f(\lambda) = \begin{cases} \lambda & \text{if } \lvert\lambda\rvert \geq \lvert\lambda_r\rvert \\ 0 & \text{otherwise} \end{cases}$	Eq. 3.7

Since both $\mathbf{\Lambda}$ and $\mathbf{U}^{\mathrm{T}}\mathbf{A}_{\mathrm{b}}\mathbf{U}$ are available after having computed the eigenvalue decomposition of $\mathbf{A}_{\mathrm{a}}$, the main computational part of our method lies in the rank-reduced eigenvalue decomposition of $\mathbf{A}_{\mathrm{a}}$. Additionally, the computation of $\mathbf{U}^{\mathrm{T}}\mathbf{A}_{\mathrm{b}}\mathbf{U}$ has runtime complexity $O(r|E|)$ and the curve fitting runtime is only dependent on r. By rank reduction, the final least-squares problem has only size r, and the runtime complexity of computing $\mathbf{U}^{\mathrm{T}}\mathbf{A}_{\mathrm{b}}\mathbf{U}$ is $O(rn^2)$. Since the rank-reduced eigenvalue decomposition of $\mathbf{A}_{\mathrm{a}}$ is computed anyway for spectral link prediction methods and because $r \ll n \ll n^2$ since $\mathbf{A}_{\mathrm{b}}$ is sparse, it follows that this curve fitting method has only negligible overhead.

A related idea is to use $\mathbf{A}_{\mathrm{a}} + \mathbf{A}_{\mathrm{b}}$ instead of $\mathbf{A}_{\mathrm{b}}$ as the target matrix for optimization, in the assumption that a link prediction algorithm should return high values for known edges. With this modification, we compute $\mathbf{U}^{\mathrm{T}}(\mathbf{A}_{\mathrm{a}} + \mathbf{A}_{\mathrm{b}})\mathbf{U}$ instead of $\mathbf{U}^{\mathrm{T}}\mathbf{A}_{\mathrm{b}}\mathbf{U}$. Our tests showed that this variant gave worse results in almost all cases. We therefore do not consider this idea any further.

Finally, we must consider the problem of scaling of eigenvalues. The function F is trained on the source matrix $\mathbf{A}_{\mathrm{a}}$ and applied on the target matrix $\mathbf{A}_{\mathrm{b}}$. Since the target adjacency matrix describes more edges than the source matrix, the corresponding spectra of the two matrices will be different: The eigenvalues of $\mathbf{A}_{\mathrm{b}}$ are larger than those of $\mathbf{A}_{\mathrm{a}}$. Therefore, we propose to scale the function F in a way that makes it independent of the largest eigenvalue of the target matrix. Let λ_{a} be the largest absolute eigenvalue of $\mathbf{A}_{\mathrm{a}}$ and λ_{b} the largest absolute eigenvalue of $\mathbf{A}_{\mathrm{b}}$. Then the normalized spectral transformation F' is given by

$$F'(\mathbf{\Lambda}) = F\left(\frac{|\lambda_{\mathrm{a}}|}{|\lambda_{\mathrm{b}}|}\mathbf{\Lambda}\right). \tag{4.10}$$

Figure 4.3 shows our curve fitting method applied to individual combinations of base matrices and graph kernels for the University Rovira i Virgili email (A@). In this example, a look at the fitted curves suggests that path counting fits the data well; the matrix exponential fits less well; and the Neumann kernel and rank reduction fit badly. We will test in Section 4.4 whether this behavior correlates with high and low prediction accuracy as we would expect.

4.2.3 Normalized and Laplacian Kernels

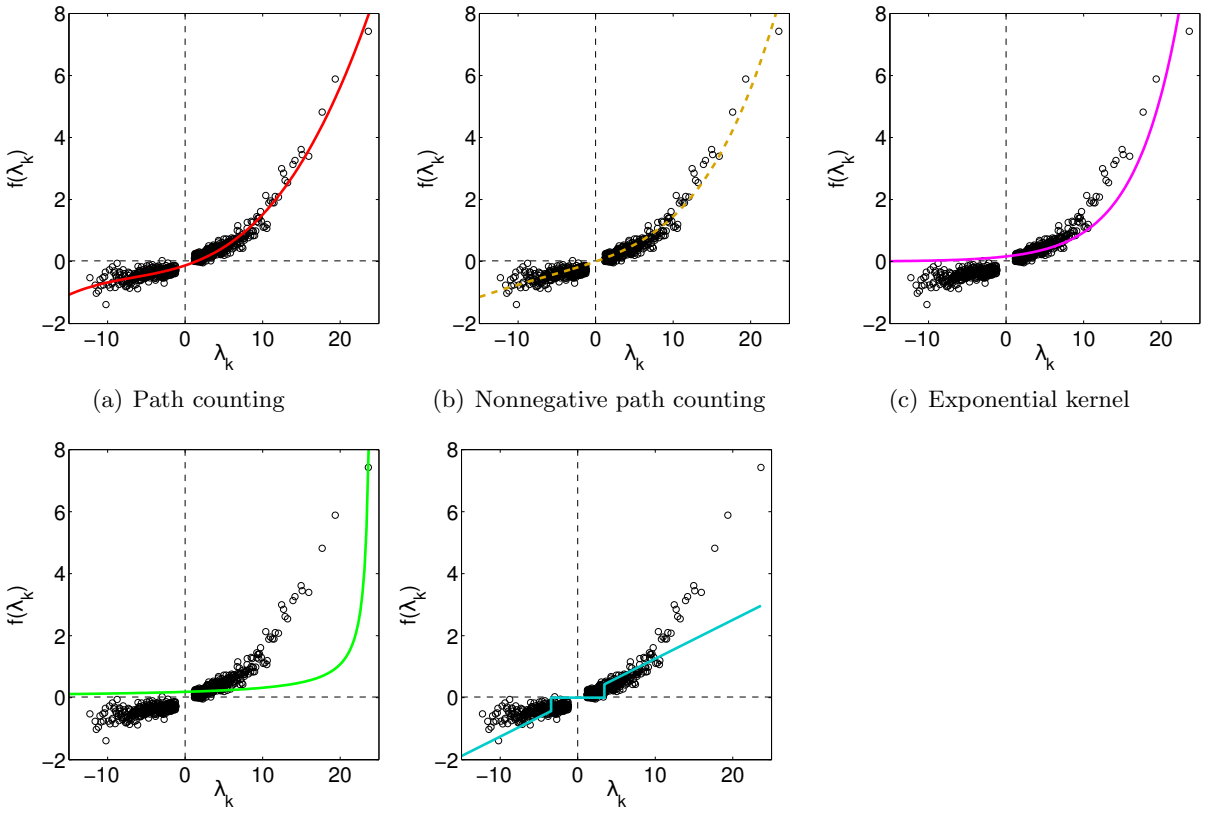
The method of learning a link prediction kernel by curve fitting also applies to the Laplacian and normalized adjacency matrices. Instead of using the non-normalized adjacency matrix $\mathbf{A}$ as the source matrix, we use the normalized adjacency matrix $\mathbf{N}$, the Laplacian $\mathbf{L}$, or the normalized Laplacian $\mathbf{Z}$ as the source matrix. Here, the matrices $\mathbf{N}$, $\mathbf{L}$ and $\mathbf{Z}$ are understood to be computed over the source edge set E_{a}.

Normalized Kernels The exponential and Neumann graph kernels as presented in Section 3.3.3 are spectral transformations of the non-normalized adjacency matrix $\mathbf{A}$. If instead we use the normalized adjacency matrix $\mathbf{N} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, we arrive at the normalized adjacency graph kernels:

$$K_{\mathrm{EXP}}(\mathbf{N}) = \exp(\alpha\mathbf{N}) = \sum_{k=0}^{\infty} \frac{\alpha^k}{k!}\mathbf{N}^k \tag{4.11}$$

$$K_{\mathrm{NEU}}(\mathbf{N}) = (\mathbf{I} - \alpha\mathbf{N})^{-1} = \sum_{k=0}^{\infty} \alpha^k\mathbf{N}^k \tag{4.12}$$

For the normalized exponential kernel, any positive value can be used for the parameter $\alpha > 0$. For the normalized Neumann kernel, the constraint on the parameter is $0 < \alpha < 1$, because the largest eigenvalue of $\mathbf{N}$ is one. As shown in the next section, the two normalized graph kernels



(a) Path counting (b) Nonnegative path counting (c) Exponential kernel

(d) Neumann Kernel (e) Rank reduction

Figure 4.3: Curve fitting with individual functions of the adjacency matrix $\mathbf{A}$, using the University Rovira i Virgili email network (A@).

are both equivalent to graph kernels over the normalized Laplacian matrix $\mathbf{Z} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{N}$.

Similarly to the non-normalized case, we can consider path counts on the graph with normalized edge weights. The result are power series of the matrix $\mathbf{N}$:

$$p(\mathbf{N}) = \sum_{k=0}^{\infty} \alpha_k \mathbf{N}^k \tag{4.13}$$

A power series of the normalized adjacency matrix $\mathbf{N}$ in an unweighted network can be interpreted as the same power series of the non-normalized adjacency matrix of the weighted network in which each edge $\{i, j\}$ is given the weight $1/\sqrt{d(i)d(j)}$, with $d(i)$ being the degree of node i. As in the non-normalized case, good link prediction functions are expected when the coefficients are nonnegative and decreasing, i.e. $\alpha_0 \geq \alpha_1 \geq \ldots \geq 0$.

Laplacian Kernels Instead of adjacency matrices, we can also use Laplacian matrices to define graph kernels. $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the non-normalized Laplacian of the graph, and $\mathbf{Z} = \mathbf{I} - \mathbf{N} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ is the normalized Laplacian. The Laplacian matrices are symmetric and positive-semidefinite. When the network is unweighted, the multiplicity of eigenvalue zero of $\mathbf{L}$ and $\mathbf{Z}$ equals the number of connected components in the network. The Laplacian matrices are thus singular for nonempty networks. The eigenvalues of the normalized Laplacian matrix $\mathbf{Z}$ lie in the interval $[0, 2]$.

Using the eigenvalue decomposition of the Laplacian, several Laplacian graph kernels can be computed. These Laplacian graph kernels are known in the literature and have also be used for link prediction [SK03]. Spectral transformations of the Laplacian matrix have been used for semi-supervised learning [CWS03], where they are called transfer functions. In this setting, the labels of unlabeled nodes are learned in a graph where only few nodes are labeled. In these studies, only polynomials and stepwise linear transfer functions are considered. These graph kernels are all spectral transformations of the Laplacian, and therefore they can be learned using curve fitting.

Moore–Penrose Pseudoinverse Before we introduce commute-time kernels, we first need to introduce the Moore–Penrose pseudoinverse. The Moore–Penrose pseudoinverse (or simply *pseudoinverse*) is a generalization of the matrix inverse which applies also to matrices that do not have a regular inverse, including rectangular matrices. We will only need the Moore–Penrose pseudoinverse for square symmetric matrices, and will therefore only give a definition for these kinds of matrices. Recall that a square matrix $\mathbf{A}$ is invertible when there is a matrix $\mathbf{A}^{-1}$ such that

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}.$$

If such an $\mathbf{A}^{-1}$ exists, then we have $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$. Matrices that do not have an inverse are called *singular*. While singular matrices do not have an inverse, we can define a weaker form of inverse called the Moore–Penrose pseudoinverse. For any matrix $\mathbf{A}$, there is a unique $\mathbf{A}^+$ such that

$$\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}.$$

Such an $\mathbf{A}^+$ always exists and is called the Moore–Penrose pseudoinverse of $\mathbf{A}$. If $\mathbf{A}$ is invertible, then $\mathbf{A}^+ = \mathbf{A}^{-1}$. If $\mathbf{A}$ is square and symmetric, then $\mathbf{A}^+$ can be expressed as a spectral transformation of $\mathbf{A}$ as

$$\begin{aligned}\mathbf{A} &= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}} \\ \mathbf{A}^+ &= \mathbf{U}\mathbf{\Lambda}^+\mathbf{U}^{\mathrm{T}}\end{aligned}$$

where $\mathbf{\Lambda}^+$ is the diagonal matrix defined by

$$(\mathbf{\Lambda}^+)_{ii} = \begin{cases} \mathbf{\Lambda}_{ii}^{-1} & \text{if } \mathbf{\Lambda}_{ii} \neq 0 \\ 0 & \text{if } \mathbf{\Lambda}_{ii} = 0 \end{cases}$$

In other words, the Moore–Penrose pseudoinverse is the spectral transformation that inverts all nonzero eigenvalues and keeps all zero eigenvalues. As an example, the Moore–Penrose pseudoinverses of the zero matrix $\mathbf{0}$ and of the identity matrix $\mathbf{I}$ are

$$\begin{aligned}\mathbf{0}^+ &= \mathbf{0}, \\ \mathbf{I}^+ &= \mathbf{I}^{-1} = \mathbf{I}.\end{aligned}$$

As another example, the Moore–Penrose of the following symmetric 2×2 matrix is given by its eigenvalue decomposition:

$$\begin{aligned}
\begin{bmatrix} 1.6 & 1.2 \\ 1.2 & 0.9 \end{bmatrix} &= \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 2.5 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^{\mathrm{T}} \\
\begin{bmatrix} 1.6 & 1.2 \\ 1.2 & 0.9 \end{bmatrix}^{+} &= \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 2.5 & 0 \\ 0 & 0 \end{bmatrix}^{+} \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^{\mathrm{T}} \\
&= \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 0.4 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^{\mathrm{T}} \\
&= \begin{bmatrix} 0.256 & 0.192 \\ 0.192 & 0.144 \end{bmatrix}
\end{aligned}$$

Commute-time Kernels Since the Laplacian matrix is positive-semidefinite, its Moore–Penrose pseudoinverse is also positive-semidefinite and can be used as a graph kernel [FPRS04]. The commute-time kernel can be applied to both the non-normalized Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ and to the normalized Laplacian $\mathbf{Z} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$:

$$K_{\mathrm{COM}}(\mathbf{L}) = \mathbf{L}^{+} \tag{4.14}$$
$$K_{\mathrm{COM}}(\mathbf{Z}) = \mathbf{Z}^{+} \tag{4.15}$$

This kernel is sensible because the most significant eigenvectors of the Laplacian matrices are those with smallest nonzero eigenvalue, which are mapped to the largest eigenvalues in the Moore–Penrose pseudoinverse. The eigenvalue zero however remains at zero with the Moore–Penrose pseudoinverse. This is what we want, since the corresponding eigenvector is the constant vector and would contribute only a constant term to the link prediction values.

The pseudoinverse $\mathbf{L}^{+}$ can be used to derive a metric:

$$d(i,j) = (\mathbf{L}^{+})_{ii} + (\mathbf{L}^{+})_{jj} - (\mathbf{L}^{+})_{ij} - (\mathbf{L}^{+})_{ji} \tag{4.16}$$

This metric has two interpretations. If the network is interpreted as an electrical network where each edge is a resistor, this metric gives the equivalent resistance between any two nodes [KR93]. The kernel $\mathbf{L}^{+}$ is thus known as the resistance distance kernel and $\mathbf{L}$ as the Kirchhoff matrix. When considering a random walk on the graph G, the average commute time between any two node equals the resistance distance. The equivalence between the two formalisms is studied in [DS84]. Note that the resistance defined in Equation 4.16 can be interpreted as a squared Euclidean distance in the following way: If $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ is the eigenvalue decomposition of the matrix $\mathbf{L}$, then let $\mathbf{W} = \mathbf{U}(\mathbf{\Lambda}^{+})^{1/2}$. Then, interpret each row $\mathbf{W}_{i\bullet}$ of $\mathbf{W}$ as the coordinate of vertex i in a space of dimension n. Then, the squared Euclidean distance between vertices in this spaces equals the resistance distance. For every pair (i,j), we have:

$$\begin{aligned}
(\mathbf{W}_{i\bullet} - \mathbf{W}_{j\bullet})^2 &= \mathbf{W}_{i\bullet}\mathbf{W}_{i\bullet}^{\mathrm{T}} + \mathbf{W}_{j\bullet}\mathbf{W}_{j\bullet}^{\mathrm{T}} - \mathbf{W}_{i\bullet}\mathbf{W}_{j\bullet}^{\mathrm{T}} - \mathbf{W}_{j\bullet}\mathbf{W}_{i\bullet}^{\mathrm{T}} \\
&= \mathbf{U}_{i\bullet}\mathbf{\Lambda}^{+}\mathbf{U}_{i\bullet}^{\mathrm{T}} + \mathbf{U}_{j\bullet}\mathbf{\Lambda}^{+}\mathbf{U}_{j\bullet}^{\mathrm{T}} - \mathbf{U}_{i\bullet}\mathbf{\Lambda}^{+}\mathbf{U}_{j\bullet}^{\mathrm{T}} - \mathbf{U}_{j\bullet}\mathbf{\Lambda}^{+}\mathbf{U}_{i\bullet}^{\mathrm{T}} \\
&= (\mathbf{L}^{+})_{ii} + (\mathbf{L}^{+})_{jj} - (\mathbf{L}^{+})_{ij} - (\mathbf{L}^{+})_{ji} \\
&= d(i,j)
\end{aligned}$$

To impose smoothness on Laplacian graph kernels, they can be regularized by adjusting specific spectral transformations to them [SK03]. In particular, we will call the following kernels

the regularized Laplacian kernels:

$$K_{\mathrm{REG}}(\mathbf{L}) = (\mathbf{I} + \alpha\mathbf{L})^{-1} = \sum_{k=0}^{\infty}(-\alpha)^k\mathbf{L}^k \tag{4.17}$$

$$K_{\mathrm{REG}}(\mathbf{Z}) = (\mathbf{I} + \alpha\mathbf{Z})^{-1} = \sum_{k=0}^{\infty}(-\alpha)^k\mathbf{Z}^k \tag{4.18}$$

The parameter α must be larger than zero in both cases. For the infinite series representation to be defined, α must be less than the inverted largest eigenvalue of $\mathbf{L}$ and $\mathbf{Z}$, i.e. $\alpha < 1/2$ for $\mathbf{Z}$. Otherwise, the kernel is still defined, but does not have a series expansion. The regularized Laplacian kernels characterize the proximity of two vertices by the number of ways they can be connected by random forests [CS97]. For this reason, they are also called random forest kernels. The normalized regularized Laplacian is equivalent to the normalized Neumann kernel by noting that $(\mathbf{I} + \alpha\mathbf{Z})^{-1} = (1 + \alpha)(\mathbf{I} - \alpha\mathbf{N})^{-1}$. This is true because we have $\mathbf{Z} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{N}$.

Heat Diffusion By considering a process of heat diffusion on the network, a Laplacian heat diffusion kernel can be derived [ISKM05]:

$$K_{\mathrm{HEAT}}(\mathbf{L}) = \exp(-\alpha\mathbf{L}) \tag{4.19}$$

$$K_{\mathrm{HEAT}}(\mathbf{Z}) = \exp(-\alpha\mathbf{Z}) \tag{4.20}$$

The normalized heat diffusion kernel is equivalent to the normalized exponential kernel given by $\exp(-\alpha\mathbf{Z}) = e^{-\alpha}\exp(\alpha\mathbf{N})$ [SK03]. The heat diffusion kernel is also known as the Laplacian exponential diffusion kernel [FYPS06].

The normalized Laplacian can be derived from the normalized adjacency matrix by the spectral transformation $\mathbf{Z} = F(\mathbf{N}) = \mathbf{I} - \mathbf{N}$ corresponding to the real function $f(\lambda) = 1 - \lambda$. Thus, the normalized commute-time kernel reduces to the normalized Neumann kernel and the heat diffusion kernel reduces to the normalized exponential kernel.

Generalized Laplacian Kernels The Laplacian graph kernels described above are all spectral transformations of $\mathbf{L}$ or $\mathbf{Z}$. This can be generalized to any inverted power series $p(\lambda)$:

$$K_{\mathrm{GEN}}(\mathbf{L}) = p(\mathbf{L})^+ = \left(\sum_{k=0}^{\infty}\alpha_k\mathbf{L}^k\right)^+ \tag{4.21}$$

$$K_{\mathrm{GEN}}(\mathbf{Z}) = p(\mathbf{I} - \mathbf{Z}) = \sum_{k=0}^{\infty}\alpha_k(\mathbf{I} - \mathbf{Z})^k \tag{4.22}$$

$$\tag{4.23}$$

As shown in Section 3.3.2, power series are spectral transformations and therefore the generalized Laplacian kernels are spectral transformations of $\mathbf{L}$ and $\mathbf{Z}$. We represent generalized Laplacian kernels as inverted Laplacians because it allows us to formulate two requirements: the spectral transformation must have nonnegative eigenvalues for the kernel to be positive-semidefinite, and the spectral transformation function should be decreasing, because eigenvectors with smaller eigenvalues should have more weight in the resulting kernel. These two requirements can be achieved by using inverted power series with nonnegative coefficients. Therefore, we require that $\alpha_k \geq 0$ for all k. These kernels generalize the previously defined Laplacian graph kernels, because their respective inverse spectral transformations can be represented as Taylor series.

Table 4.2: Normalized and Laplacian link prediction functions and their corresponding real functions. The methods marked with †, ‡ and ⋄ represent classes of graph kernels that are equivalent up to a shift and scaling of eigenvalues, as described in the text.

Method	**Matrix function**	**Real function**	
Normalized exponential kernel †	$K_{\mathrm{EXP}}(\mathbf{N}) = \exp(\alpha\mathbf{N})$	$f(\lambda) = e^{\alpha\lambda}$	Eq. 4.11
Normalized Neumann kernel ‡	$K_{\mathrm{NEU}}(\mathbf{N}) = (\mathbf{I} - \alpha\mathbf{N})^{-1}$	$f(\lambda) = 1/(1-\alpha\lambda)$	Eq. 4.12
Normalized path counting ⋄	$p(\mathbf{N}) = \sum_{k=0}^{\infty} \alpha_k \mathbf{N}^k$	$f(\lambda) = \sum_{k=0}^{\infty} \alpha_k \lambda^k$	Eq. 4.13
Commute-time kernel	$K_{\mathrm{COM}}(\mathbf{L}) = \mathbf{L}^+$	$f(\lambda) = \begin{cases} \lambda^{-1} & \text{if } \lambda > 0 \\ 0 & \text{otherwise} \end{cases}$	Eq. 4.14
Normalized commute-time kernel	$K_{\mathrm{COM}}(\mathbf{Z}) = \mathbf{Z}^+$	$f(\lambda) = \begin{cases} \lambda^{-1} & \text{if } \lambda > 0 \\ 0 & \text{otherwise} \end{cases}$	Eq. 4.15
Regularized Laplacian kernel	$K_{\mathrm{REG}}(\mathbf{L}) = (\mathbf{I} + \alpha\mathbf{L})^{-1}$	$f(\lambda) = 1/(1+\alpha\lambda)$	Eq. 4.17
Normalized regularized Laplacian kernel ‡	$K_{\mathrm{REG}}(\mathbf{Z}) = (\mathbf{I} + \alpha\mathbf{Z})^{-1}$	$f(\lambda) = 1/(1+\alpha\lambda)$	Eq. 4.18
Heat diffusion	$K_{\mathrm{HEAT}}(\mathbf{L}) = \exp(-\alpha\mathbf{L})$	$f(\lambda) = e^{-\alpha\lambda}$	Eq. 4.19
Normalized heat diffusion †	$K_{\mathrm{HEAT}}(\mathbf{Z}) = \exp(-\alpha\mathbf{Z})$	$f(\lambda) = e^{-\alpha\lambda}$	Eq. 4.20
Generalized Laplacian kernel	$p(\mathbf{L})^+ = \left(\sum_{k=0}^{\infty} \alpha_k \mathbf{L}^k\right)^+$	$f(\lambda) = \left(\sum_{k=0}^{\infty} \alpha_k \lambda^k\right)^{-1}$	Eq. 4.21
Generalized normalized Laplacian kernel ⋄	$p(\mathbf{I} - \mathbf{Z}) = \sum_{k=0}^{\infty} \alpha_k (\mathbf{I} - \mathbf{Z})^k$	$f(\lambda) = \sum_{k=0}^{\infty} \alpha_k (1-\lambda)^k$	Eq. 4.22

The generalized normalized Laplacian kernel can be written as a spectral transformation of a power series of the normalized adjacency matrix $\mathbf{N}$ due to the equality $\mathbf{Z} = \mathbf{I} - \mathbf{N}$:

$$p(\mathbf{I} - \mathbf{Z}) = p(\mathbf{N})$$

Therefore, the generalized normalized Laplacian kernel is equivalent to path counting with normalized edge weights.

Summary Table 4.2 shows the underlying real functions for the normalized and Laplacian graph kernels. A larger set of Laplacian kernels is described in [FYPS06], most of which contain additional parameters. Note that the Laplacian kernels are all decreasing spectral transformations $f(\lambda)$ of the Laplacian matrices $\mathbf{L}$ and $\mathbf{Z}$. As a result, the dominant eigenvectors of Laplacian graph kernels are the eigenvectors of the smallest nonzero eigenvalue of $\mathbf{L}$ or $\mathbf{Z}$. Since we compute only r eigenvectors and eigenvalues of the Laplacian matrices, we have to make sure that we find the smallest r eigenvalues and their eigenvectors. This is possible with standard linear algebra packages, as described in Appendix C.

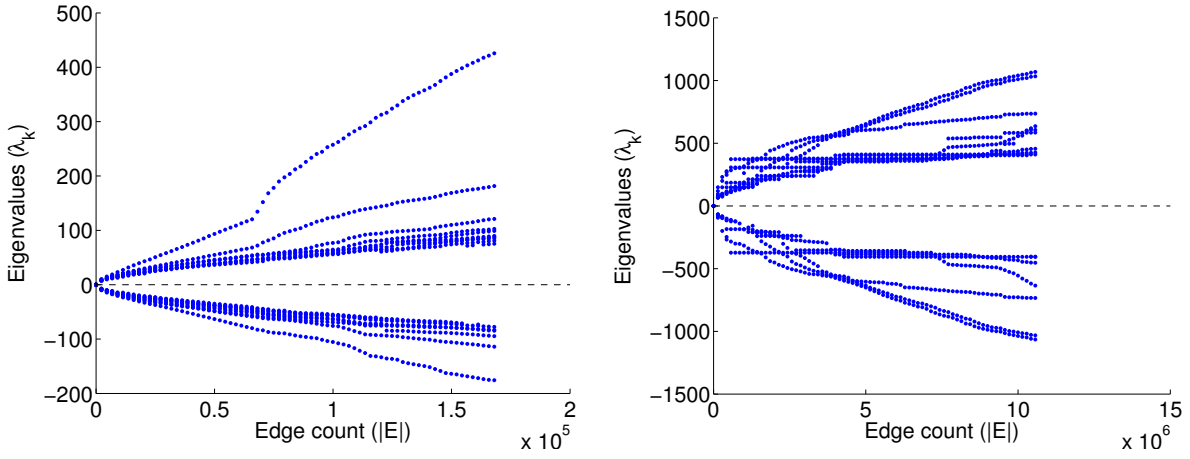
4.3 Learning by Spectral Extrapolation

We now derive another method for link prediction based on the spectral evolution model. According to the spectral evolution model established in the previous chapter, the spectrum of a network evolves over time, while the eigenvectors remain constant. We saw that this assumption is true to a certain point in large real-world networks and used it in the last section to reduce

the link prediction problem to a simple curve-fitting problem. To do this, we chose a time in the past and learned a mapping from the network's structure in the past to the network's structure in the present. Instead, we will now consider infinitesimal changes in the present network, and extrapolate them into the future.

According to the spectral evolution model, only the eigenvalues of a network change significantly over time. Therefore, we will look at the change in eigenvalues of a network and extrapolate it into the future. Because we consider each eigenvalue separately, this method is suitable for cases where eigenvalues of a single network grow at very different speeds. In such a case, graph kernels as learned in the previous section are not accurate, since they assume all eigenvalues to grow according to a single real function, the spectral transformation function. The method presented here can therefore be considered as an extension of graph kernels to irregular growth patterns, while still being conformant to the spectral evolution model.

Link prediction functions such as the matrix exponential assume there is a real function $f(\lambda)$ that applies to all eigenvalues. In the link prediction functions described in Section 3.3, this function is very regular, i.e. it is positive and grows monotonically. Observed spectral transformations such as the one in Figure 4.4(b) are however irregular, and no simple function solves the curve fitting problem well. This is explained by the fact that eigenvalues may cross each other, indicating that a non-monotonous function is needed. Note that the irregularity here is only observed in the growth of eigenvalues. The overall growth is still spectral, and eigenvectors are nearly constant.



(a) Regular growth

(b) Irregular growth

Figure 4.4: Examples of (a) regular and (b) irregular spectral growth. In the first case, the eigenvalues grow regularly, and a graph kernel can be used to model network evolution. In the second case, spectral growth is irregular and graph kernels are not suitable. In both cases, growth is spectral. (a) Internet topology (TO), (b) Twitter user–user "@name" mentions (Wa).

For these reasons, we must consider the growth of each eigenpair separately. If we knew that eigenvalues did not cross each other, we could compare the k^{th} eigenvalue at two points in time and extrapolate a third value. But because eigenvalues pass each other, we must first learn the relation between new and old eigenvalues.

4.3.1 Method

Given the set of edges sorted by creation time, we define the following timestamps: t_{a} is the time of split between the source and target sets, and t_{b} is the time of split between the training

and the test set.

Let $\mathbf{A}_\mathrm{a}$ be the adjacency matrix containing all edges in the source set and $\mathbf{A}_\mathrm{b}$ the adjacency matrix containing all edges in the target set. The adjacency matrix of the training set is then $\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{b}$. We now compute the eigenvalue decompositions of the adjacency matrices at times t_a and t_b:

$$\begin{aligned} \mathbf{A}_\mathrm{a} &= \mathbf{U}_\mathrm{a}\,\mathbf{\Lambda}_\mathrm{a}\,\mathbf{U}_\mathrm{a}^\mathrm{T} \\ \mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{b} &= \mathbf{U}_\mathrm{b}\,\mathbf{\Lambda}_\mathrm{b}\,\mathbf{U}_\mathrm{b}^\mathrm{T} \end{aligned}$$

To find out which eigenpair k at time t_a corresponds to eigenpair l at time t_b, we compute a mean of eigenvalues at time t_a, weighted by the similarity between the two eigenpairs at time t_a and at time t_b. As a similarity measure, we use the absolute dot product between the corresponding normalized eigenvectors of index k and l at times t_a and t_b, as used previously in Equation 3.1. Additionally, we also try out the square of that dot product as another similarity measure. This gives two similarity measures:

$$\mathrm{sim}_\mathrm{abs}(k,l) = |(\mathbf{U}_\mathrm{a})_{\bullet k} \cdot (\mathbf{U}_\mathrm{b})_{\bullet l}| \tag{4.24}$$

$$\mathrm{sim}_\mathrm{squ}(k,l) = ((\mathbf{U}_\mathrm{a})_{\bullet k} \cdot (\mathbf{U}_\mathrm{b})_{\bullet l})^2 \tag{4.25}$$

Thus if $(\mathbf{\Lambda}_\mathrm{b})_{ll}$ is an eigenvalue of $\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{b}$ at time t_b, its conjectured previous value at time t_a is given by

$$(\hat{\mathbf{\Lambda}}_\mathrm{a})_{ll} = \left(\sum_k \mathrm{sim}(k,l)\right)^{-1} \sum_k \mathrm{sim}(k,l)(\mathbf{\Lambda}_\mathrm{a})_{kk},$$

where $(\mathbf{\Lambda}_\mathrm{a})_{kk}$ is the k^th eigenvalue at time t_a. Note that this expression only depends on the eigenvectors at time t_a and not on the eigenvalues at time t_b. This is because we estimate the value of the l^th eigenvalue at time t_a, and the eigenvectors at time t_b are only used for the extrapolation itself in the next step. We then perform linear extrapolation to predict an eigenvalue $(\hat{\mathbf{\Lambda}}_\mathrm{c})_{ll}$ in the future, i.e. at a time $t_\mathrm{c} = 2t_\mathrm{b} - t_\mathrm{a}$.

$$(\hat{\mathbf{\Lambda}}_\mathrm{c})_{ll} = 2(\mathbf{\Lambda}_\mathrm{b})_{ll} - (\hat{\mathbf{\Lambda}}_\mathrm{a})_{ll} \tag{4.26}$$

Using the matrix $\hat{\mathbf{\Lambda}}_\mathrm{c}$, the predicted edge weights are then $\mathbf{U}_\mathrm{b}\hat{\mathbf{\Lambda}}_\mathrm{c}\mathbf{U}_\mathrm{b}^\mathrm{T}$. Note that the computed eigenvalues $(\hat{\mathbf{\Lambda}}_\mathrm{c})_{ll}$ are not necessarily ordered, which is not a problem for link prediction scores.

The extrapolation defined in this way is linear, i.e. we only use a linear approximation to the observed growth to predict future eigenvalues. A simple extension of this method takes into account higher-order terms. For instance, we could observe the change in the growth itself instead of just growth in the eigenvalues and extrapolate that into the future. Our observations however indicate that this will not make the predictions more accurate. Instead, it would make the predictions less accurate because of overfitting. The linear extrapolation method itself is already susceptible to learn growth patterns that are not present in the data, due to the high number of parameters it contains (one for each latent dimension). Higher-order extrapolation methods have more parameters and would therefore be even less well-behaved.

4.4 Experiments

We will now compare both new link prediction methods experimentally with each other, and against baseline algorithms. First, we recall the basic differences in the two methods in the following table.

Table 4.3: Comparison of the two learning methods.

Curve fitting	**Extrapolation**
Uses any characteristic matrices	Uses only the adjacency matrix
Considers only two timepoints	Considers all timepoints
Works well when growth is regular	Works well when growth is irregular

To evaluate the performance of the two link predictions methods, we use the unipartite, unweighted networks of Table A.1 in Appendix A, and compute the link prediction accuracy using the mean average precision as defined in Section 4.1.4.

All decompositions are computed only up to a rank r which depends on the dataset. Values for all datasets are given in Table A.2 in Appendix A. We apply the curve fitting methods and spectral extrapolation methods described in the previous sections to the source and target edge sets of each network, learning several link prediction functions. We then apply these link prediction functions to the training set of each network to compute predictions for the edges in the test set. Since the networks in these experiments are unweighted, the test set contains edges and non-edges, representing the zero test set as described in Section 4.1.3. As a baseline, we use the local link prediction functions described in Section 4.1.1.

The complete list of link prediction methods is given in Table 4.4. In comparison to the previous, theoretical definitions of the graph kernels, this table gives the actual variants we used for evaluation. They differ in that we chose a maximal path length in the path counting method, resulting in a polynomial of the corresponding degree. We also implement nonnegative polynomials. Also, the triangle closing method is augmented with a constant and a linear term, giving a polynomial of degree two. The Laplacian graph kernels from Equations 4.18, 4.20 and 4.22 are equivalent to normalized graph kernels (N-NEU, N-EXP) and normalized path counting (N-POLY) as noted in Table 4.2, and are only given once, using their name as normalized kernels. The common neighbors measure from Equation 4.1 is subsumed by triangle closing (A-TRI), and is therefore omitted.

Best Performing Link Prediction Methods The best performing link prediction methods for a selection of unipartite, undirected, unweighted datasets are shown in Table 4.5. Whenever the best-performing method is a spectral transformation learned by curve fitting or spectral extrapolation, we give the best transformation used, along with the method. The full list of best-fitting curves is given in Appendix B. From the list of best-performing link prediction methods in this table and in the appendix, we conclude that there is no overall best link prediction method, but that a certain number of link prediction methods are best for different datasets. All these link prediction functions are best for at least one dataset: (normalized) (nonnegative) path counting (A-POLY, A-POLYN, N-POLY, N-POLYN), the matrix exponential (A-EXP) and the normalized commute-time kernel (N-COM).

Spectral Extrapolation Figure 4.5 shows the extrapolation method applied to the English Wikipedia hyperlink graph (WP).

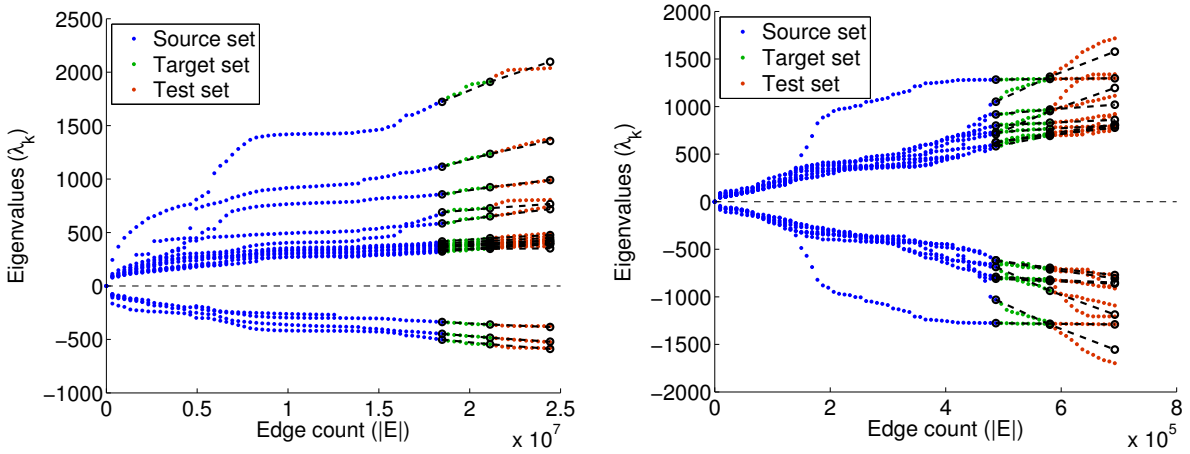
Comparison of Methods The mean average precision is given for six unipartite, undirected, unweighted datasets in Figures 4.6, 4.7 and 4.8. The mean average precision for one representative algorithm from each category is given for the same datasets in Figure 4.9. The complete results for four datasets are given in Figure 4.10.

Table 4.4: The complete list of link prediction methods used in the evaluation. All parameters α, β, γ are required to be positive, except for path counting and the generalized Laplacian kernel.

	Method	**Definition**	
Local link prediction methods			
P-PA	Preferential attachment	$p_{\mathrm{PA}}(i,j)$	Eq. 4.2
P-JAC	Jaccard	$p_{\mathrm{JAC}}(i,j)$	Eq. 4.3
P-AA	Adamic–Adar	$p_{\mathrm{AA}}(i,j)$	Eq. 4.4
Curve fitting with A			
A-TRI	Triangle closing	$\alpha\mathbf{I} + \beta\mathbf{A} + \gamma\mathbf{A}^2$	Eq. 3.3
A-POLY	Path counting	$\sum_{k=1}^{4} \alpha_k \mathbf{A}^k$	Eq. 3.4
A-POLYN	Nonnegative path counting	$\sum_{k=1}^{7} \alpha_k \mathbf{A}^k, \alpha_k \geq 0$	Eq. 3.4
A-EXP	Exponential kernel	$\beta \exp(\alpha\mathbf{A})$	Eq. 3.5
A-NEU	Neumann kernel	$\beta(\mathbf{I} - \alpha\mathbf{A})^{-1}$	Eq. 3.6
A-RR	Rank reduction	$\beta R_r(\mathbf{A})$	Eq. 3.7
Curve fitting with N			
N-POLY	Normalized path counting	$\sum_{k=0}^{4} \alpha_k \mathbf{N}^k$	Eq. 4.13
N-POLYN	Normalized nonnegative path counting	$\sum_{k=0}^{7} \alpha_k \mathbf{N}^k, \alpha_k \geq 0$	Eq. 4.13
N-EXP	Normalized exponential kernel	$\beta \exp(\alpha\mathbf{N})$	Eq. 4.11
N-NEU	Normalized Neumann kernel	$\beta(\mathbf{I} - \alpha\mathbf{N})^{-1}$	Eq. 4.12
N-COM	Normalized commute-time kernel	$\beta(\mathbf{I} - \mathbf{N})^{+}$	Eq. 4.15
Curve fitting with L			
L-COM	Commute-time kernel	$\beta\mathbf{L}^{+}$	Eq. 4.14
L-REG	Regularized Laplacian kernel	$\beta(\mathbf{I} + \alpha\mathbf{L})^{-1}$	Eq. 4.17
L-HEAT	Heat diffusion	$\beta \exp(-\alpha\mathbf{L})$	Eq. 4.19
L-POLY	Generalized Laplacian kernel	$\left(\sum_{k=0}^{\infty} \alpha_k \mathbf{L}^k\right)^{-1}$	Eq. 4.21
Spectral extrapolation			
X-ABS	Absolute similarity weights	$\lvert(\mathbf{U}_{\mathrm{a}})_{\bullet k} \cdot (\mathbf{U}_{\mathrm{b}})_{\bullet l}\rvert$	Eq. 4.24
X-SQU	Squared similarity weights	$((\mathbf{U}_{\mathrm{a}})_{\bullet k} \cdot (\mathbf{U}_{\mathrm{b}})_{\bullet l})^2$	Eq. 4.25

Table 4.5: The best performing spectral transformation for a sample of datasets. For each dataset, we show the source and target matrices, the curve fitting model and the link prediction method that performs best.

Dataset	Best transformation	Best method	MAP
Haggle (CO)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-NEU	0.738
arXiv astro-ph (AP)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-POLY	0.690
Pretty Good Privacy (PG)	$\mathbf{L}_\mathrm{a} \to \mathbf{A}_\mathrm{b}$	L-REG	0.990
Caenorhabditis elegans (PM)	$\mathbf{N}_\mathrm{a} + \mathbf{N}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-TRI	0.940
CAIDA (IN)	$\mathbf{N}_\mathrm{a} + \mathbf{N}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	N-COM	0.984
Route Views (AS)	$\mathbf{L}_\mathrm{a} \to \mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{b}$	L-COM	0.954
U. Rovira i Virgili (A@)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-POLYN	0.972



(a) Flickr friends (FL)

(b) Facebook wall posts (Ow)

Figure 4.5: The result of applying spectral extrapolation to the Flickr (FL) and Facebook (Ow) networks. Spectral extrapolation is computed using only the decompositions at the split points between the source, target and test sets. Individual dots denote actual growth, whereas the dashed lines represent the predicted growth of the eigenvalues.

To find out which link prediction method has the better overall prediction accuracy, we perform pairwise statistical comparisons between individual methods. For a given pair of link prediction methods, we compare their mean average precisions on all network datasets. We use Student's t-test to determine whether a link prediction method performs significantly better than another one. For instance, we find that the exponential kernel A-EXP performs significantly better than the Neumann kernel A-NEU. Student's t-test works as follows: Let $\mathrm{map}_G(p_\mathrm{X})$ be the mean average precision of the link prediction algorithm X on the network dataset G. Say, we want to compare the link prediction accuracy of the link prediction methods X and Y, and let $\mathcal{G}$ be the set of all network datasets. Then, we define the mean difference μ_XY and standard

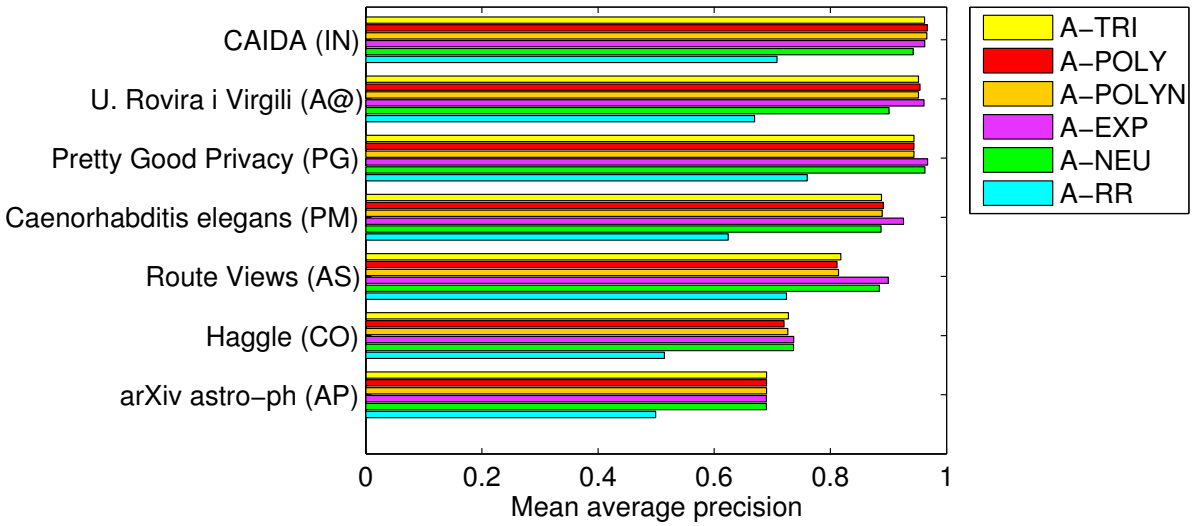


Figure 4.6: The mean average precision for curve fitting methods based on the adjacency matrix $\mathbf{A}$ evaluated on six unipartite, unweighted, undirected datasets.

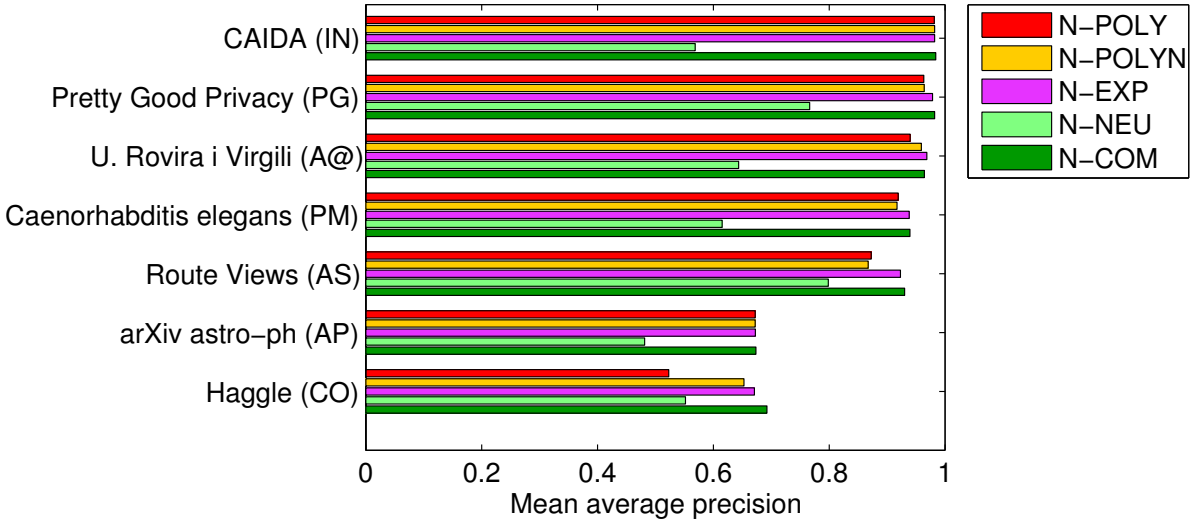


Figure 4.7: The mean average precision for curve fitting methods based on the normalized adjacency matrix $\mathbf{N}$ evaluated on six unweighted, undirected datasets.

deviation of differences σ_{XY} of MAP values as

$$\Delta_{\mathrm{XY}}(G) = \mathrm{map}_G(p_{\mathrm{X}}) - \mathrm{map}_G(p_{\mathrm{Y}}) \tag{4.27}$$

$$\mu_{\mathrm{XY}} = \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \Delta_{\mathrm{XY}}(G) \tag{4.28}$$

$$\sigma_{\mathrm{XY}} = \sqrt{\frac{1}{|\mathcal{G}| - 1} \sum_{G \in \mathcal{G}} (\Delta_{\mathrm{XY}}(G) - \mu_{\mathrm{XY}})^2} \tag{4.29}$$

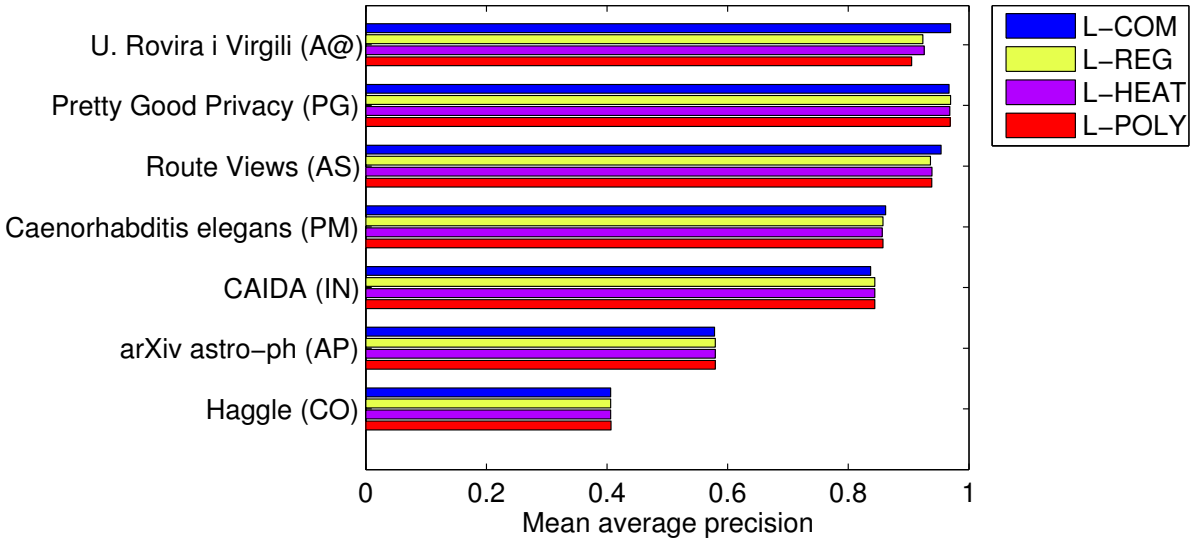


Figure 4.8: The mean average precision for curve fitting methods based on the Laplacian matrix $\mathbf{L}$ evaluated on six unweighted, undirected datasets.

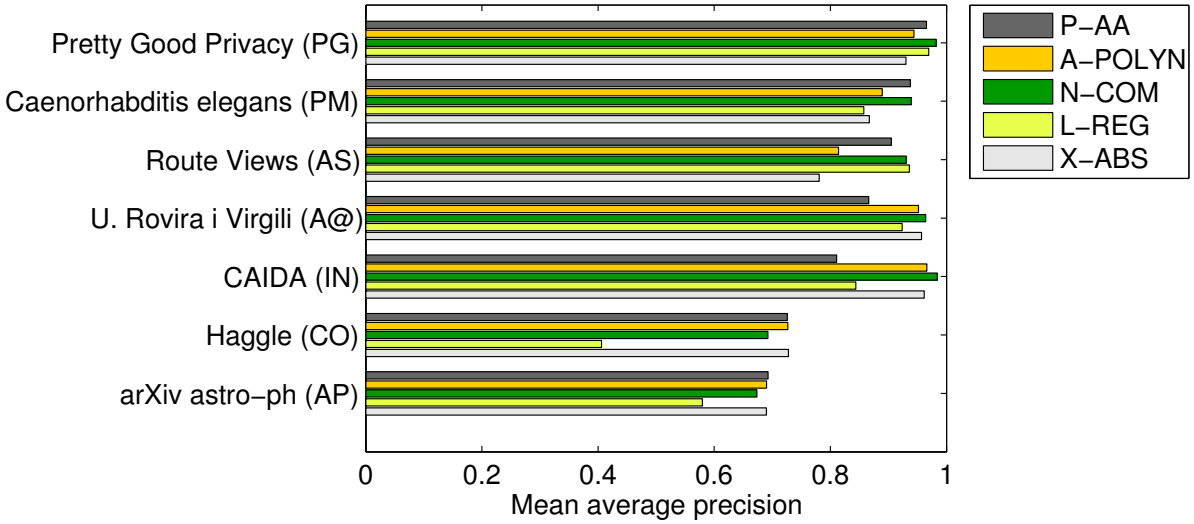
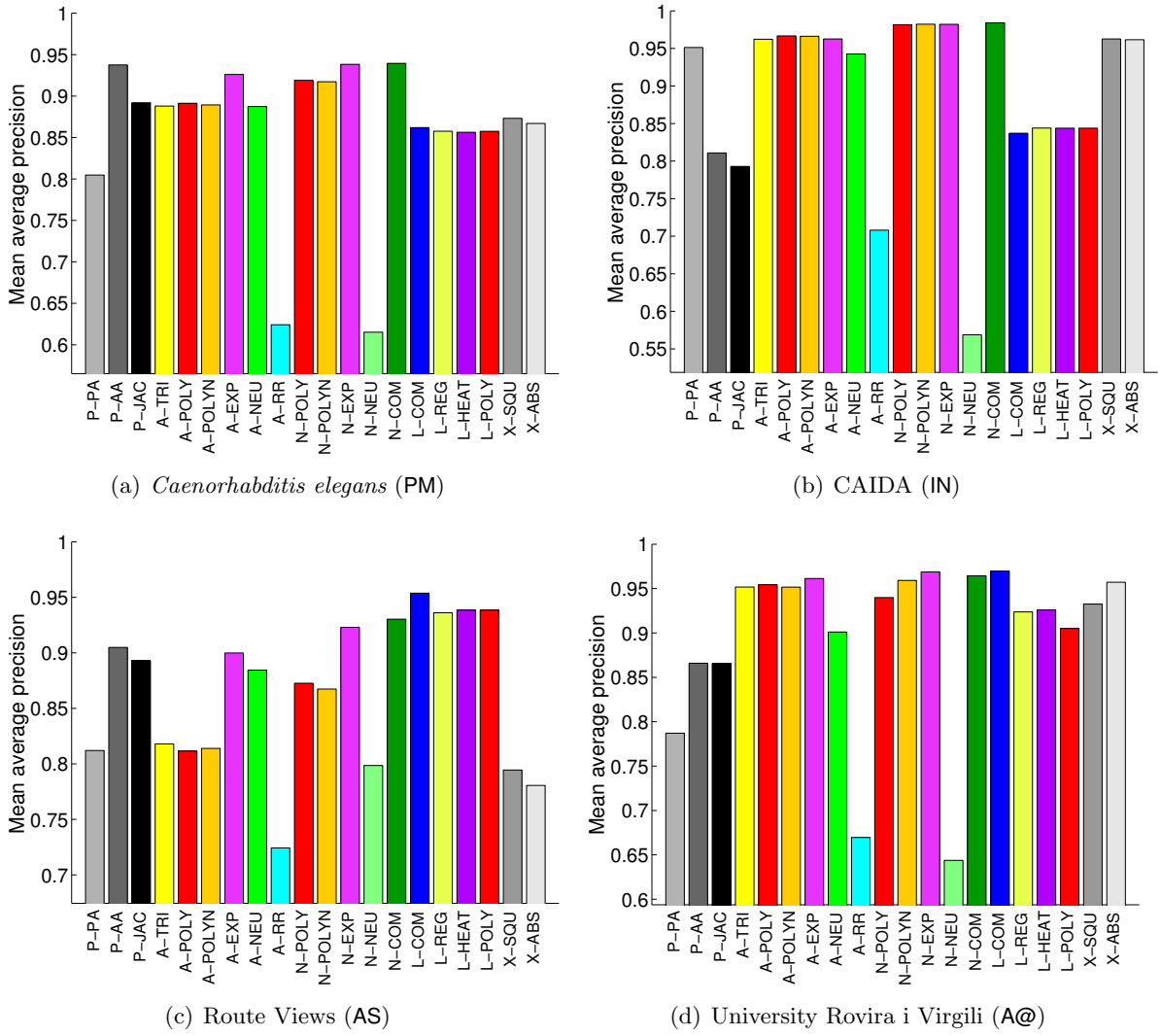


Figure 4.9: A comparison of one link prediction function from each category (local methods, curve fitting with the adjacency matrix, curve fitting with the normalized adjacency matrix, curve fitting with the Laplacian, extrapolation).

Then, we compute the t value as

$$t_{\mathrm{XY}} = \mu_{\mathrm{XY}} \frac{\sqrt{|\mathcal{G}|}}{\sigma_{\mathrm{XY}}} \tag{4.30}$$

We consider method X to be statistically better performing than method Y when t_{XY} is larger than the 95^{th} percentile of Student's t-distribution with degree-of-freedom parameter equal to $|\mathcal{G}| - 1$. We show individual scatter plots of link prediction method pairings in Figure 4.11. In these plots, each dataset is represented as one point, drawn at the coordinates of the performance of two link prediction methods on it.



(a) *Caenorhabditis elegans* (PM)

(b) CAIDA (IN)

(c) Route Views (AS)

(d) University Rovira i Virgili (A@)

Figure 4.10: All evaluation results for four unipartite, unweighted, undirected networks.

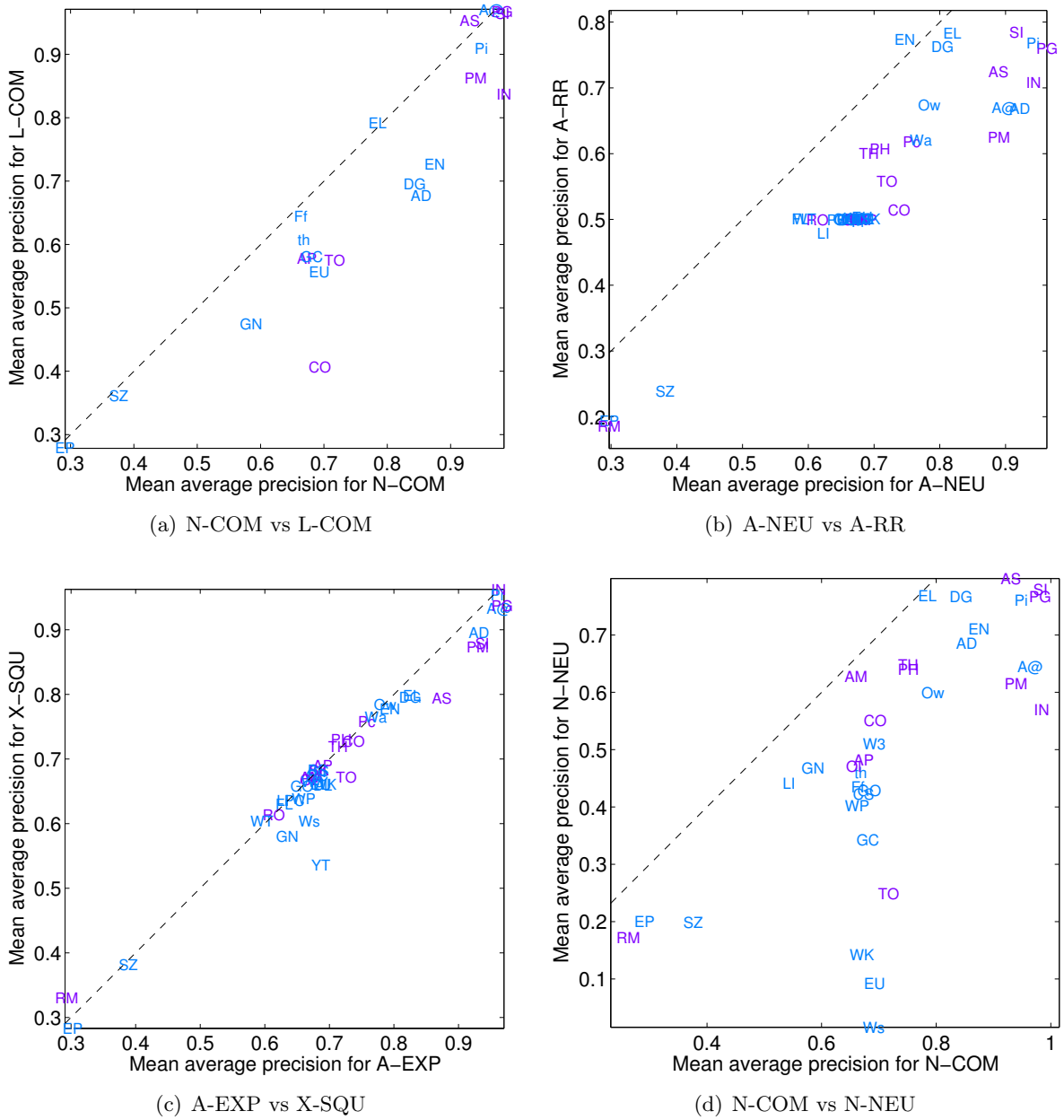
Observations We make the following observations:

- Among the functions of the adjacency matrix $\mathbf{A}$, nonnegative path counting (A-POLYN) has the best prediction accuracy, closely followed by the exponential kernel (A-EXP) and path counting (A-POLY). Triangle closing (A-TRI), the Neumann kernel (A-NEU) and rank reduction (A-RR) all have significantly worse prediction accuracy.

- The best functions of the normalized adjacency matrix $\mathbf{N}$ are normalized path counting (N-POLY and N-POLYN) and the normalized commute-time kernel (N-COM). The normalized exponential kernel and the normalized Neumann kernel have significantly worse prediction accuracy.

- The best function of the adjacency matrix (A-POLYN) and the best function of the normalized adjacency matrix (N-COM) have comparable link prediction accuracy.

- Spectral transformations of the Laplacian $\mathbf{L}$ have significantly worse link prediction accuracy than functions of $\mathbf{A}$ and $\mathbf{N}$. This result is consistent with the absence of Laplacian



(a) N-COM vs L-COM

(b) A-NEU vs A-RR

(c) A-EXP vs X-SQU

(d) N-COM vs N-NEU

Figure 4.11: Pairwise comparisons of link prediction methods by their mean average precision on all datasets. We show only comparisons in which one method is significantly more accurate than the other.

spectral evolution observed in Section 3.4.3, and also with the result from [RLH09] that the resistance distance (which is based on the matrix $\mathbf{L}$) is meaningless for large random geometric graphs. Among functions of $\mathbf{L}$, the regularized Laplacian kernel (L-REG) performs best, but not significantly.

- The spectral extrapolation methods (X-ABS and X-SQU) perform worse than the curve fitting methods.
- Among local link prediction functions, the Jaccard coefficient (P-JAC) and the Adamic–Adar measure (P-AA) correlate to a high degree. They both have a prediction accuracy

comparable to that of preferential attachment (P-PA). All local link prediction methods perform significantly worse than the best curve fitting methods.

As an overall conclusion, we recommend using curve fitting to learn nonnegative path counting (A-POLYN) and the normalized commute-time kernel (N-COM) as candidates for link prediction applications. However, we recommend to generate the curve fitting plot described in Section 4.2.2 in any case to catch eventual new spectral transformations patterns. Also, we recommend to perform the spectral diagonality test described in Section 3.2.4 beforehand, to catch those datasets where spectral transformations do not work.

4.5 Related Learning Methods

Several previous attempts have been made at learning spectral transformations. These attempts have been restricted to specific datasets and problem types, and did not consider the validity of the spectral evolution model.

Joint Diagonalization Simultaneously diagonalizable matrices have been considered in the context of *joint diagonalization*, where given a set of matrices $\{\mathbf{A}_i\}$, a matrix $\mathbf{U}$ is searched that makes the matrices $\{\mathbf{U}\mathbf{A}_i\mathbf{U}^{\mathrm{T}}\}$ as diagonal as possible according to a given matrix norm, as described e.g. in [WS97]. Joint diagonalization methods have been used for link prediction before [STF06]. By construction, the predicted network evolution is spectral, since the same eigenvector matrix $\mathbf{U}$ is used at each timepoint. However, a high number of timepoints must be considered, whereas both methods we have introduced only need to consider two timepoints. Equivalently, this can be described as the decomposition of the vertex-vertex-time tensor [SCAK11]. In fact, our spectral evolution model provides a justification for these advanced methods, since joint diagonalization implies constant or near-constant eigenvectors.

Linear and Quadratic Programming In [ZKGL05] and [ZKLG06], spectral transformations of the Laplacian matrix are considered, where they are called *spectral transforms*. The spectral transformations in that work are learned using quadratically constrained quadratic programming (QCQP). A similar technique is used in [LJD09]. In [LQCL09], a spectral transformation of the Laplacian matrix is learned using linear programming.

4.6 Summary: Learning Spectral Transformations

We have presented two methods for learning link prediction functions in large networks. The first method uses spectral curve fitting and the second one spectral extrapolation. Both methods make different assumptions: The curve fitting method learns a spectral mapping from one matrix to another, and therefore can also be applied to matrices other than the adjacency matrix, such as the Laplacian matrix. The spectral extrapolation method on the other hand considers infinitesimal changes in the adjacency matrix's spectrum. It is therefore suited for network datasets where exact edge creation times are known. However, it cannot be applied to the Laplacian matrix or other characteristic graph matrices other than the adjacency matrix, since it presupposes spectral evolution, which we did not observe for Laplacian matrices.

The spectral extrapolation method suffers from the case where eigenvalues are very near each other. In practice, this is not common, but can be observed in very decentral networks, i.e. networks without a clear core component, such as road networks.

We also found that if a dataset is so big that only few eigenvectors and -values can be computed, the resulting points are not numerous enough to make a meaningful distinction

between the performance of the different curve fitting models. This can be interpreted as a case of overfitting: Using only two or three eigenpairs, a linear function of the spectrum is probably the best prediction we can make. By comparing the results with various types of network datasets, we found that the various link prediction methods in current use are justified in specific cases. We hope that the methods presented here can help make an informed choice as to the right method. The advantage of the curve fitting method lies in the fact that we do not have to rely on the mean average precision of a link prediction function. Instead, we can look at its curve fitting plot and come up with new spectral transformations, if necessary.

Our link prediction experiments on one hundred and eighteen datasets showed that a consistently high prediction accuracy is achieved by our curve fitting method based on the non-normalized and normalized adjacency matrices ($\mathbf{A}$ and $\mathbf{N}$). In particular, we recommend non-negative path counting (A-POLYN) and the normalized commute-time kernel (N-COM).

In the networks we tested, the spectral extrapolation method provides more accurate link prediction in many cases, in particular in networks with irregular—but still spectral—growth. For networks with very regular growth, regular graph kernels perform better however. Across all datasets, the performance of our methods is better than any single link prediction method. Our new extrapolation method is also parameter free: Not only are there no parameters, as in various graph kernels, but our methods make the choice of a specific spectral growth model unnecessary.

Chapter 5

Signed Networks

As we saw in previous chapters, many graph-theoretic data mining problems can be solved by spectral methods which consider matrices associated with a given network and compute their eigenvalues and eigenvectors. Common examples are spectral clustering, graph drawing and graph kernels used for link prediction and link weight prediction. In the usual setting, only graphs with unweighted or positively weighted edges are supported. Some data mining problems however apply to signed graphs, i.e. graphs that contain positively and negatively weighted edges. In this chapter, we extend our spectral machine learning methods to the signed case using specially-defined characteristic matrices.

Intuitively, a positive edge in a graph denotes proximity or similarity. Analogously, a negative edge may denote dissimilarity or distance. Negative edges are found in many types of networks: social networks may contain not just *friend* but also *foe* links, and connections between users and products may denote *like* and *dislike*. In other cases, negative edges may be introduced explicitly, as in the case of constrained clustering, where some pairs of points *must* or *must not* be in the same cluster. These problems can all be modeled with signed graphs. In order to do this, we define variants of the Laplacian and related matrices that result in signed variants of spectral clustering, prediction and visualization methods.

A main contribution in this chapter is the definition of the Laplacian matrix for signed networks, and therefore this chapter puts a larger emphasis on the Laplacian matrix than other chapters. The second contribution of this chapter is a verification of the multiplication rule summarized as *The enemy of my enemy is my friend*, which amounts to multiplying weights of adjacent edges. As we will see, using both the adjacency and the Laplacian matrix of signed graphs assumes this multiplication rule. In this chapter, we will consider only undirected signed graphs. Directed and bipartite signed graphs are introduced in the next chapter. Other contributions in this chapter are a new spectral graph drawing algorithm for signed graphs, the definition of the algebraic conflict ξ, the introduction of signed spectral clustering, the application of Laplacian graph kernels to link sign prediction and the signed resistance distance.

We begin by a short review of related work in Section 5.1. Then, we introduce the Laplacian matrix of signed graphs by considering the problem of graph drawing in Section 5.2. Section 5.3 then gives precise mathematical definitions related to signed graphs. A theoretical analysis of the spectrum of the Laplacian matrix of signed graphs is given in Section 5.4. The related problem of spectral clustering in signed graphs is described in Section 5.5. Section 5.6 then reviews link prediction functions for signed networks, and presents experimental results of the methods presented in the previous chapter for signed graphs. Section 5.7 then introduces the concepts of conflict and centrality in signed networks. Section 5.8 gives alternative derivations of the Laplacian matrix for signed graphs. Finally, Section 5.9 reviews related approaches that are not otherwise covered in the chapter.

5.1 Background

Negative edges can be found in many types of networks. Applications use unipartite signed graphs to model either enmity in addition to friendship, distrust in addition to trust, or positive and negative ratings between users. Early uses of signed graphs can be found in anthropology, where negative edges have been used to denote antagonistic relationships between tribes [HH83]. In this context, the sociological notion of balance is defined as the absence of negative cycles, i.e. the absence of cycles with an odd number of negative edges [Har53, DM96]. Other sociological applications of signed networks include student relationships [HJD+04] and voting processes [LHK10a].

Recent studies [HWSB08] describe the social network extracted from Essembly, an ideological discussion site that allows users to mark other users as *friends*, *allies* and *nemeses*, and discuss the semantics of the three relation types. These works model the different types of edges by means of three subgraphs. Other recent work considers the task of discovering communities from social networks with negative edges [YCL07].

In trust networks, nodes represent persons or other entities, and links represent trust relationships. To model distrust, negative edges are then used. Work in that field has mostly focused on defining global trust measures using path lengths or adapting PageRank [KD08, GKRT04, KSGM03, TB06, MH05].

In applications where users can rate each other, we can model ratings as *like* and *dislike*, giving rise to positive and negative edges. In practice, ratings are given on a scale with more than two levels, and signed edge weights are derived by subtracting the overall mean rating from each edge weight, i.e. by additive normalization. As an example, consider the the Líbímseti.cz dating site, where users can rate each other on a scale from 1 to 10. By subtracting the mean rating 5.9 from every rating, all ratings in the range $\{1, \ldots, 5\}$ result in negative edge weights.

Mathematically, a signed graph can be defined as $G = (V, E, \sigma)$, where V is the vertex set, E is the edge set, and $\sigma : E \to \{+, -\}$ is the sign function [Zas08]. The sign function σ assigns a positive or negative sign to each edge. In this work, we will identify a signed graph $G = (V, E, \sigma)$ with a weighted graph $G = (V, E, w)$ in which the weight function is given by $w(i, j) = +1$ when $\sigma(\{i, j\}) = +$ and $w(i, j) = -1$ when $\sigma(\{i, j\}) = -$. In fact, we can model weighted signed graphs as weighted graphs in which we allow any real value for the weight $w(i, j)$. In the rest of this chapter, all networks will therefore be considered to be weighted and signed. A synthetic example of a signed network is given in Figure 5.1. In all drawings of signed graphs, we will show positive edges as solid green lines and negative edges as dashed red lines.

As for unsigned graphs, the adjacency matrix $\mathbf{A}$ is defined using $\mathbf{A}_{ij} = w(i, j)$ when $\{i, j\} \in E$ and $\mathbf{A}_{ij} = 0$ otherwise. $\mathbf{A}$ is symmetric. The diagonal degree matrix $\mathbf{D}$ for a signed graph is defined using $\mathbf{D}_{ii} = \sum_j |\mathbf{A}_{ij}|$, i.e. as the sum of the absolute weights of incident edges. If all edge weights are in $\{+1, -1\}$, $\mathbf{D}_{ii}$ is simply the number of nodes adjacent to the node i, regardless of edge signs. The Laplacian matrix of a signed graph is then defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The reason for using the absolute value in the definition of $\mathbf{D}$ will be made clear in the next section, when we will derive the signed Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ in the context of graph drawing.

The spectrum of the signed Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is studied in [Hou05], where it is established that the signed Laplacian is positive-definite when each connected component of a graph contains a cycle with an odd number of negative edges. Basic properties of the Laplacian matrix for signed graphs are given in [HLP03].

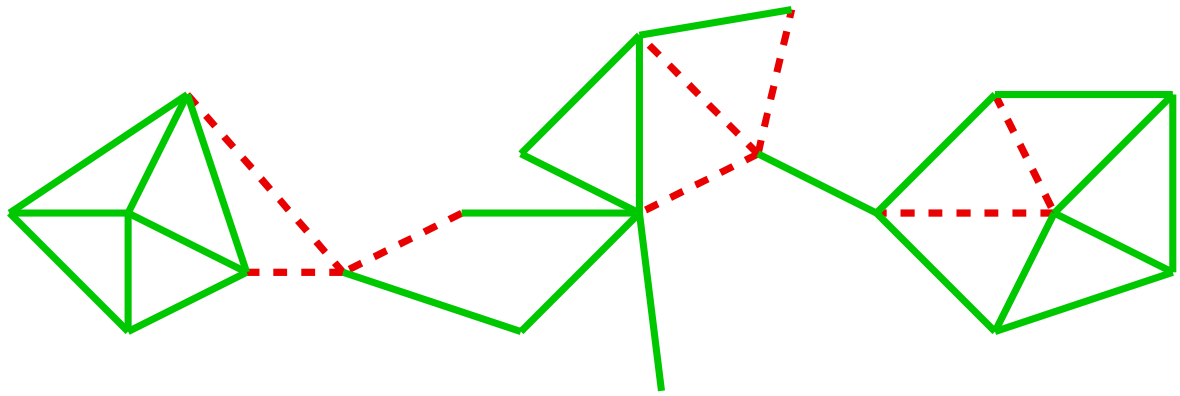


Figure 5.1: An example of an undirected signed graph, with positive edges represented as solid green lines and negative edges represented as dashed red lines. All signed networks considered in this chapter are undirected such as this one.

5.2 Signed Graph Drawing

To motivate signed spectral graph theory, we consider the problem of drawing signed graphs, and show how it naturally leads to the signed normalized Laplacian. Complete definitions of all terms are given in the next section. We will begin by showing how the signed Laplacian matrix arises naturally in the task of drawing graphs with negative edges when one tries to place each node near to its positive neighbors and opposite to its negative neighbors, extending a standard method of graph drawing in the presence of only positive edges.

The Laplacian matrix turns up in graph drawing when we try to find an embedding of a graph into a plane in a way that adjacent nodes are drawn near to each other [BN02]. In the literature, signed graphs have been drawn using eigenvectors of the signed adjacency matrix [BFL06]. Instead, our approach consists of using the Laplacian to draw signed graphs, in analogy with the unsigned case. To do this, we will stipulate that negative edges should be drawn as far from each other as possible. First however, we review the derivation of the Laplacian matrix of unsigned graphs using the problem of graph drawing.

5.2.1 Unsigned Graphs

We now describe the general method for generating an embedding of the nodes of an unsigned graph into the plane using the Laplacian matrix. Let $G = (V, E)$ be a connected unsigned graph with adjacency matrix $\mathbf{A}$. We want to find a two-dimensional drawing of G in which each vertex is drawn near to its neighbors. This requirement gives rise to the following vertex equation, which states that every vertex is placed at the mean of its neighbors' coordinates, weighted by the weight of the connecting edges. Let $\mathbf{X} \in \mathbb{R}^{n \times 2}$ be a matrix whose columns are the coordinates of all nodes in the drawing, then we have for each node i:

$$\mathbf{X}_{i\bullet} = \left(\sum_{\{i,j\} \in E} \mathbf{A}_{ij} \right)^{-1} \sum_{\{i,j\} \in E} \mathbf{A}_{ij} \mathbf{X}_{j\bullet} \tag{5.1}$$

Rearranging and aggregating the equation for all i we arrive at

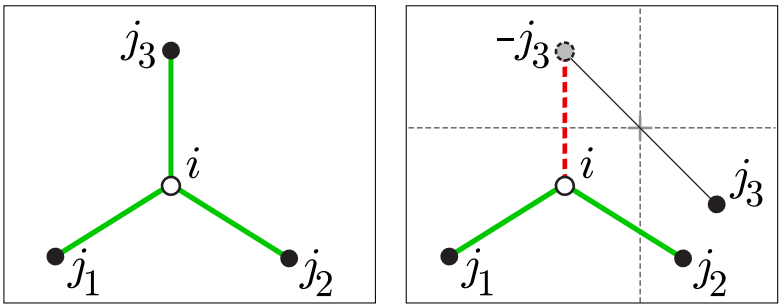
$$\begin{aligned} & \mathbf{DX} = \mathbf{AX} \\ \Leftrightarrow \quad & \mathbf{LX} = \mathbf{0} \end{aligned} \tag{5.2}$$

where $\mathbf{D}$ is the diagonal matrix defined by $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ and $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the non-normalized Laplacian of G. In other words, the columns of $\mathbf{X}$ should belong to the null space[1] of $\mathbf{L}$, which leads to the degenerate solution of $\mathbf{X}_{i\bullet} = \mathbf{1}$ for all i, i.e. each $\mathbf{X}_{i\bullet}$ having all components equal, as $\mathbf{1}$ is an eigenvector of $\mathbf{L}$ with eigenvalue zero. To exclude that solution, we require additionally that the columns $\mathbf{X}$ be orthogonal to the all-ones vector $\mathbf{1}$. Additionally, to avoid the degenerate solution $\mathbf{X}_{i\bullet} = \mathbf{X}_{j\bullet}$ for $i \neq j$, and require that all columns of $\mathbf{X}$ be orthogonal. This leads to $\mathbf{X}_{i\bullet}$ being the eigenvectors associated with the two smallest eigenvalues of $\mathbf{L}$ different from zero. This solution results in a well-known satisfactory embedding of positively weighted graphs. Such an embedding is related to the resistance distance (or commute-time distance) between nodes of the graph [BN02].

Note that Equation (5.2) can also be transformed to $\mathbf{X} = \mathbf{D}^{-1}\mathbf{A}\mathbf{X}$, leading to the eigenvectors of the asymmetric matrix $\mathbf{D}^{-1}\mathbf{A}$. This alternative derivation is not investigated here.

5.2.2 Signed Graphs

We now extend the graph drawing method described in the previous section to graphs with positive and negative edge weights. To adapt Expression (5.1) to negative edge weights, we interpret a negative edge as an indication that two vertices should be placed on opposite sides of the drawing. Therefore, we take the opposite coordinates $-\mathbf{X}_{j\bullet}$ of vertices j adjacent to i through a negative edge, and use the absolute value of edge weights to compute the mean, as pictured in Figure 5.2. We will call this construction *antipodal proximity*.



(a) Unsigned graph (b) Signed graph

Figure 5.2: Drawing the vertex i at the mean coordinates of its neighbors j_1, j_2, j_3 by proximity and antipodal proximity. (a) In unsigned graphs, a vertex i is placed at the mean of its neighbors j_1, j_2, j_3. (b) In signed graphs, a vertex i is placed at the mean of its positive neighbors j_1, j_2 and antipodal points $-j_3$ of its negative neighbors.

This leads to the vertex equation

$$\mathbf{X}_{i\bullet} = \left(\sum_{\{i,j\} \in E} |\mathbf{A}_{ij}| \right)^{-1} \sum_{\{i,j\} \in E} \mathbf{A}_{ij} \mathbf{X}_{j\bullet} \tag{5.3}$$

resulting in a signed Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ in which we define $\mathbf{D}_{ii} = \sum_j |\mathbf{A}_{ij}|$:

$$\begin{aligned} & \mathbf{D}\mathbf{X} = \mathbf{A}\mathbf{X} \\ \Leftrightarrow \quad & \mathbf{L}\mathbf{X} = \mathbf{0} \end{aligned} \tag{5.4}$$

[1] The null space of $\mathbf{L}$, i.e. the set of vectors $\mathbf{x}$ such that $\mathbf{L}\mathbf{x} = \mathbf{0}$, is also called the *kernel* of $\mathbf{L}$. In this work, we however use the word *kernel* only to refer to positive-semidefinite functions of two variables.

This definition of the degree matrix $\mathbf{D}$ for signed graphs reduces to the definition without the absolute value for unsigned graphs. As we will see in the next section, $\mathbf{L}$ is always positive-semidefinite, and is positive-definite for graphs that are unbalanced, i.e. graphs that contain cycles with an odd number of negative edges. To obtain a graph drawing from $\mathbf{L}$, we can thus distinguish three cases, assuming that G is connected:

- If all edges are positive, then $\mathbf{L}$ has one eigenvalue zero, and the eigenvectors of the two smallest nonzero eigenvalues can be used for graph drawing.
- If the graph is unbalanced, $\mathbf{L}$ is positive-definite and we can use the eigenvectors of the two smallest eigenvalues as coordinates.
- If the graph is balanced, its spectrum is equivalent to that of the corresponding unsigned Laplacian matrix, up to signs of the eigenvector components. Using the eigenvectors of the two smallest eigenvalues (including zero), we arrive at a graph drawing with all points being placed on two parallel lines, reflecting the perfect 2-clustering present in the graph.

5.2.3 Synthetic Examples

Figure 5.3 shows four small synthetic signed graphs drawn using the eigenvectors of three characteristic graph matrices. For each synthetic signed graph, let $\mathbf{A}$ be its signed adjacency matrix, $\mathbf{L} = \mathbf{D} - \mathbf{A}$ its signed Laplacian matrix, and $\bar{\mathbf{L}} = \bar{\mathbf{D}} - \bar{\mathbf{A}}$ the Laplacian matrix of the corresponding unsigned graph $\bar{G} = |G|$, i.e. the same graph as G, only that all edges are positive. For $\mathbf{A}$, we use the eigenvectors corresponding to the two largest absolute eigenvalues. For $\mathbf{L}$ and $\bar{\mathbf{L}}$, we use the eigenvectors of the two smallest nonzero eigenvalues. These small synthetic examples are chosen to display the basic spectral properties of these three matrices. All edges have weight ± 1, and all graphs contain cycles with an odd number of negative edges. Column (a) shows all graphs drawn using the eigenvectors of the two largest eigenvalues of the adjacency matrix $\mathbf{A}$. The eigenvectors of largest absolute eigenvalues are taken of $\mathbf{A}$ in analogy with graph kernels based on $\mathbf{A}$, where the largest eigenvalue is mapped to the largest value. Column (b) shows the unsigned Laplacian embedding of the graphs by setting all edge weights to $+1$. Column (c) shows the signed Laplacian embedding. The embedding given by the eigenvectors of $\mathbf{A}$ is clearly not satisfactory for graph drawing. As expected, the graphs drawn using the ordinary Laplacian matrix place nodes connected by a negative edge near to each other. The signed Laplacian matrix produces a graph embedding where negative links span large distances across the drawing, as required.

5.3 Definitions

In this section, we give the definition of the combinatorial and normalized signed Laplacian matrices of a graph and derive their basic properties. The combinatorial Laplacian matrix of a signed graph with edge weights restricted to ± 1 is described in [Hou05] where it is called the Kirchhoff matrix (of a signed graph).

Let $G = (V, E, w)$ be an undirected graph with vertex set V, edge set E, and nonzero edge weights w described by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$. If $\{i, j\}$ is not an edge of the graph, we set $\mathbf{A}_{ij} = 0$. Otherwise, $\mathbf{A}_{ij} > 0$ denotes a positive edge and $\mathbf{A}_{ij} < 0$ denotes a negative edge. Unless otherwise noted, we assume G to be connected.

Unsigned Laplacian Matrix Given a graph G with only positively weighted edges, its ordinary Laplacian matrix is a symmetric $|V| \times |V|$ matrix that, in a general sense, captures

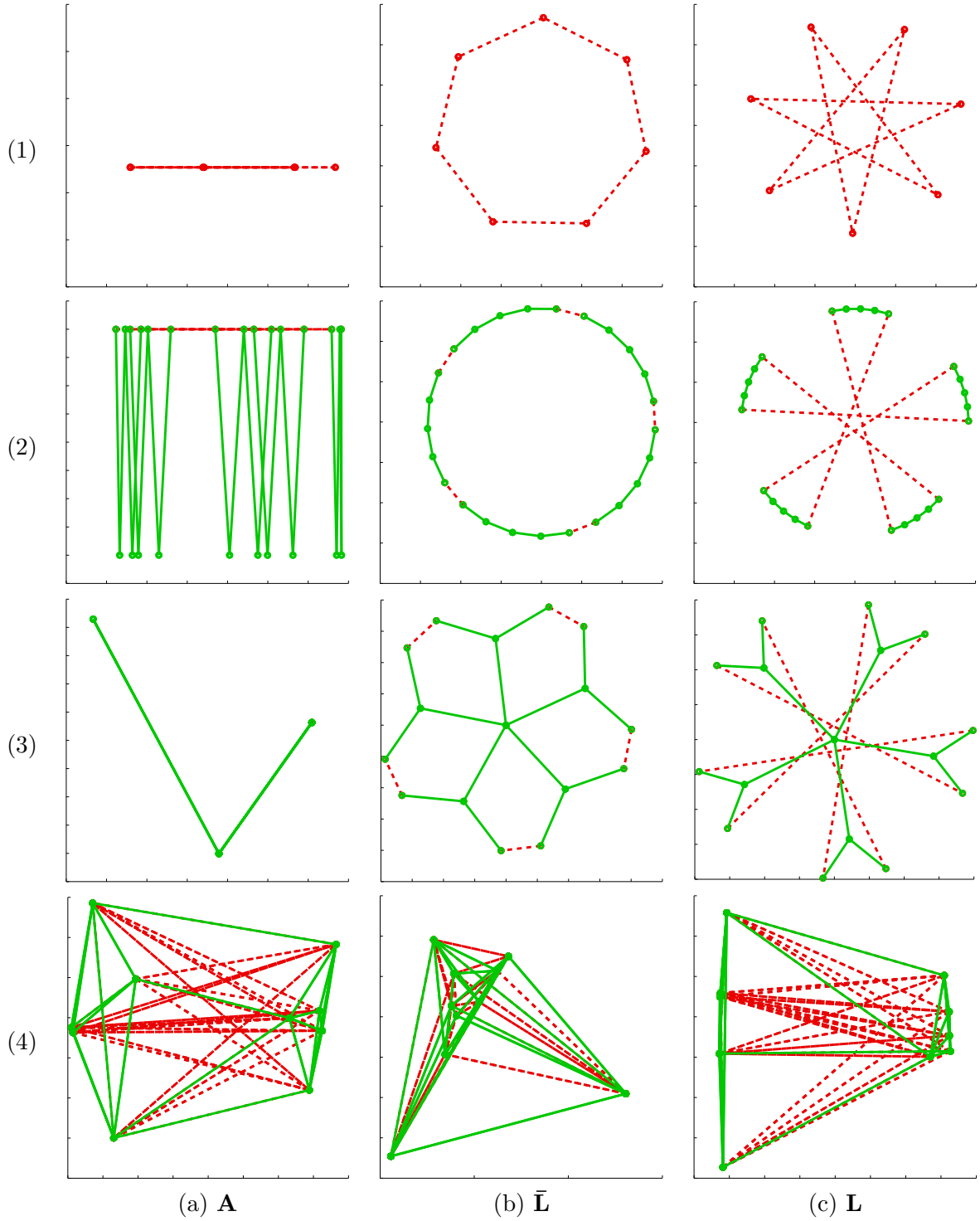


Figure 5.3: Four small synthetic signed graphs [(1)–(4)] drawn using the eigenvectors of three graph matrices. (a) the adjacency matrix $\mathbf{A}$, (b) the Laplacian $\bar{\mathbf{L}} = \bar{\mathbf{D}} - \bar{\mathbf{A}}$ of the underlying unsigned graph $\bar{G}$, (c) the Laplacian $\mathbf{D} - \mathbf{A}$. All graphs shown contain negative cycles, and their signed Laplacian matrices are positive-definite. Edges with weights $+1$ are shown as solid green lines and those with weight -1 as red dashed lines.

relations between individual nodes of the graph. The Laplacian matrix is positive-semidefinite, and its Moore–Penrose pseudoinverse as defined in Section 4.2.3 can be interpreted as a forest count [CS98] and can be used to compute the resistance distance between any two nodes [KR93].

Definition 2. *The Laplacian matrix* $\mathbf{L} \in \mathbb{R}^{|V|\times|V|}$ *of a graph* G *with nonnegative adjacency matrix* $\mathbf{A}$ *is given by*

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{5.5}$$

with the diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{|V|\times|V|}$ *given by*

$$\mathbf{D}_{ii} = \sum_{\{i,j\}\in E} \mathbf{A}_{ij}. \tag{5.6}$$

This definition is only valid for graph without negatively weighted edges. If negatively weighted edges are present, this definition must be extended.

Signed Laplacian Matrix If applied to signed graphs, the Laplacian as defined in Equation (5.5) results in an indefinite matrix, and thus cannot be used as the basis for graph kernels [LW00]. Therefore, we use a modified degree matrix $\mathbf{D}$ containing the sum of absolute edge weights [Hou05].

Definition 3. *The Laplacian matrix* $\mathbf{L} \in \mathbb{R}^{|V|\times|V|}$ *of a signed graph* G *with adjacency matrix* $\mathbf{A}$ *is given by*

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{5.7}$$

where the signed degree matrix $\mathbf{D} \in \mathbb{R}^{|V|\times|V|}$ *is the diagonal matrix given by*

$$\mathbf{D}_{ii} = \sum_{\{i,j\}\in E} |\mathbf{A}_{ij}|. \tag{5.8}$$

We prove in Section 5.4 that the signed Laplacian matrix is positive-semidefinite. Note that for unsigned graphs, the two definitions are equivalent.

Normalized Laplacians Two different matrices are usually called the normalized Laplacian, and both can be extended to signed graphs. One is the matrix $\mathbf{Z} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ as used in the rest of this thesis; the other one is $\mathbf{D}^{-1}\mathbf{A}$. The normalized Laplacian matrix $\mathbf{Z} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ as defined previously in Equation 2.23 is related to the normalized adjacency matrix $\mathbf{N}$ by $\mathbf{Z} = \mathbf{I} - \mathbf{N}$. Alternatively, one can use the random walk Laplacian as defined in [Lux07]. When modeling random walks on an unsigned graph G, the transition probability between node i to j is given by entries of the stochastic matrix $\mathbf{D}^{-1}\mathbf{A}$. This matrix also arises from Equation (5.2) as the eigenvalue equation $\mathbf{u} = \mathbf{D}^{-1}\mathbf{A}\mathbf{u}$. The matrix $\mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$ is called the random walk Laplacian and is positive-semidefinite. The random walk normalized Laplacian arises when considering random walks [DS84], but also when drawing graphs and when clustering using normalized cuts, as explained in Section 5.5.

The normalized Laplacian matrices can be used instead of the combinatorial Laplacian matrices in most settings, with good results reported for graphs with very skewed degree distributions [GMZ03].

5.4 Signed Spectral Analysis

For an unsigned graph, the Laplacian $\mathbf{L}$ is positive-semidefinite, i.e. it has only nonnegative eigenvalues. It can therefore be used as a kernel. To use the Laplacian $\mathbf{L}$ as a kernel when a graph has negative edges, we have to show that it is positive-semidefinite, too. In this section, we prove that the Laplacian matrix $\mathbf{L}$ of a signed graph is indeed positive-semidefinite, characterize the graphs for which it is positive-definite, and give the relationship between the eigenvalue decomposition of the signed Laplacian matrix and the eigenvalue decomposition of the corresponding unsigned Laplacian matrix. Our characterization of the smallest eigenvalue of $\mathbf{L}$ in terms of graph balance is based on [Hou05].

5.4.1 Positive-semidefiniteness

Theorem 1. *The Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ of a signed graph G is positive-semidefinite.*

Proof. We write the Laplacian matrix as a sum over the edges of G:

$$\mathbf{L} = \sum_{\{i,j\} \in E} \mathbf{L}^{\{i,j\}} \tag{5.9}$$

where $\mathbf{L}^{\{i,j\}} \in \mathbb{R}^{|V| \times |V|}$ contains the four following nonzero entries:

$$\begin{aligned} \mathbf{L}^{\{i,j\}}_{ii} &= \mathbf{L}^{\{i,j\}}_{jj} = |\mathbf{A}_{ij}| \\ \mathbf{L}^{\{i,j\}}_{ij} &= \mathbf{L}^{\{i,j\}}_{ji} = -\mathbf{A}_{ij} \end{aligned} \tag{5.10}$$

Let $\mathbf{x} \in \mathbb{R}^{|V|}$ be a vertex vector. By considering the bilinear form $\mathbf{x}^{\mathrm{T}} \mathbf{L}^{\{i,j\}} \mathbf{x}$, we see that $\mathbf{L}^{\{i,j\}}$ is positive-semidefinite:

$$\begin{aligned} \mathbf{x}^{\mathrm{T}} \mathbf{L}^{\{i,j\}} \mathbf{x} &= |\mathbf{A}_{ij}| \mathbf{x}_i^2 + |\mathbf{A}_{ij}| \mathbf{x}_j^2 - 2\mathbf{A}_{ij} \mathbf{x}_i \mathbf{x}_j \\ &= |\mathbf{A}_{ij}| (\mathbf{x}_i - \operatorname{sgn}(\mathbf{A}_{ij}) \mathbf{x}_j)^2 \\ &\geq 0 \end{aligned} \tag{5.11}$$

We now consider the bilinear form $\mathbf{x}^{\mathrm{T}} \mathbf{L} \mathbf{x}$:

$$\mathbf{x}^{\mathrm{T}} \mathbf{L} \mathbf{x} = \sum_{\{i,j\} \in E} \mathbf{x}^{\mathrm{T}} \mathbf{L}^{\{i,j\}} \mathbf{x} \geq 0$$

It follows that $\mathbf{L}$ is positive-semidefinite. □

Another way to prove that $\mathbf{L}$ is positive-semidefinite consists of expressing it using the incidence matrix of G. Assume that for each edge $\{i, j\}$ an arbitrary orientation is chosen. Then we define the incidence matrix $\mathbf{H} \in \mathbb{R}^{|V| \times |E|}$ of G as

$$\mathbf{H}_{i\{i,j\}} = +\sqrt{|\mathbf{A}_{ij}|} \tag{5.12}$$

$$\mathbf{H}_{j\{i,j\}} = -\operatorname{sgn}(\mathbf{A}_{ij}) \sqrt{|\mathbf{A}_{ij}|}. \tag{5.13}$$

Here, the letter $\mathbf{H}$ is the uppercase greek letter Eta, as used in e.g. [GHKZ11]. We now consider the product $\mathbf{H}\mathbf{H}^{\mathrm{T}} \in \mathbb{R}^{|V| \times |V|}$:

$$\begin{aligned} (\mathbf{H}\mathbf{H}^{\mathrm{T}})_{ii} &= \sum_{\{i,j\} \in E} |\mathbf{A}_{ij}| \\ (\mathbf{H}\mathbf{H}^{\mathrm{T}})_{ij} &= -\mathbf{A}_{ij} \end{aligned}$$

for diagonal and off-diagonal entries, respectively. Therefore $\mathbf{H}\mathbf{H}^{\mathrm{T}} = \mathbf{L}$, and it follows that $\mathbf{L}$ is positive-semidefinite. This result is independent of the orientation chosen for $\mathbf{H}$.

5.4.2 Positive-definiteness

We now show that, unlike the ordinary Laplacian matrix, the signed Laplacian matrix is strictly positive-definite for some graphs, including most real-world networks. The theorem presented here can be found in [HLP03], and also follows directly from an earlier result in [Zas82].

As with the ordinary Laplacian matrix, the spectrum of the signed Laplacian matrix of a disconnected graph is the union of the spectra of its connected components. This can be seen by noting that the Laplacian matrix of an unconnected graph has block-diagonal form, with each diagonal entry being the Laplacian matrix of a single component. Therefore, we will restrict ourselves to connected graphs. We begin the analysis by defining the balance of a graph.

Definition 4 (Harary [Har53])**.** *A connected graph with nonzero signed edge weights is balanced when its vertices can be partitioned into two groups such that all positive edges connect vertices within the same group, and all negative edges connect vertices of the two different groups.*

Figure 5.4 shows a balanced graph partitioned into two vertex sets. Equivalently, unbalanced graphs can be defined as those graphs containing a cycle with an odd number of negative edges, as shown in Figure 5.5. To prove that the balanced graphs are exactly those that do not contain cycles with an odd number of edges, consider that any cycle in a balanced graph has to cross sides an even number of times. On the other hand, any balanced graph can be partitioned into two vertex sets by depth-first traversal while assigning each vertex to a partition such that the balance property is fulfilled. Any inconsistency that arises during such a labeling leads to a cycle with an odd number of negative edges.

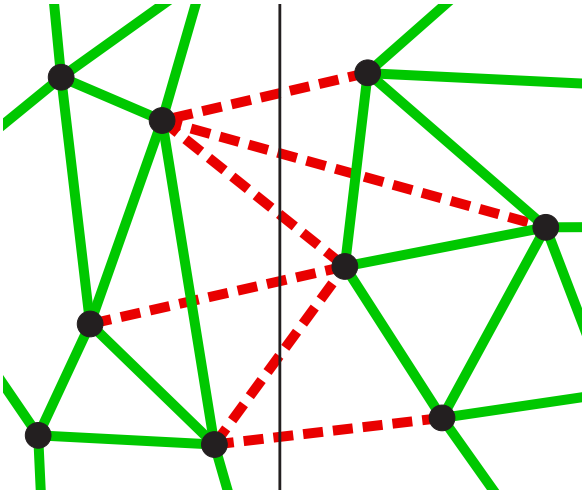


Figure 5.4: The nodes of a graph without negative cycles can be partitioned into two sets such that all edges inside of each group are positive and all edges between the two groups are negative. We call such a graph balanced, and the eigenvalue decomposition of its signed Laplacian matrix $\mathbf{L}$ can be expressed as the modified eigenvalue decomposition of the corresponding unsigned graph's Laplacian $\bar{\mathbf{L}}$.

Using this definition, we can characterize the graphs for which the signed Laplacian matrix is positive-definite.

Theorem 2. *The signed Laplacian matrix of an unbalanced graph is positive-definite.*

Proof. We show that if the bilinear form $\mathbf{x}^{\mathrm{T}}\mathbf{L}\mathbf{x}$ is zero for some vector $\mathbf{x} \neq 0$, then a bipartition of the vertices as described above exists.

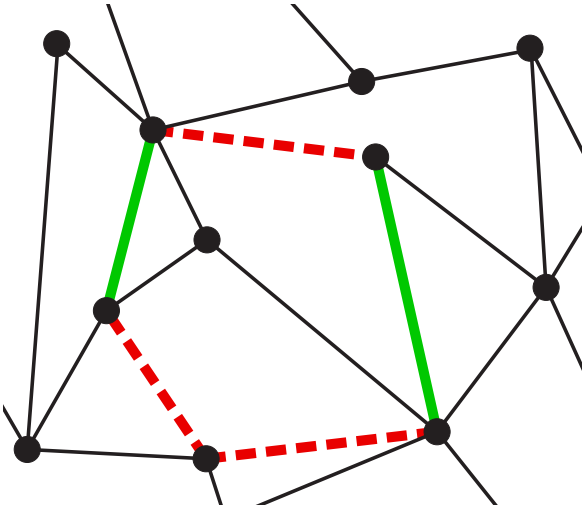


Figure 5.5: An unbalanced graph contains a cycle with an odd number of negatively weighted edges. Negatively weighted edges are shown as dashed red lines, positively weighted edges are shown as solid green lines, and edges that are not part of the cycle in black. The presence of such cycles results in a positive-definite Laplacian matrix $\mathbf{L}$.

Let $\mathbf{x}^{\mathrm{T}}\mathbf{L}\mathbf{x} = 0$. We have seen that for every $\mathbf{L}^{\{i,j\}}$ as defined in Equation 5.10 and any $\mathbf{x}$, $\mathbf{x}^{\mathrm{T}}\mathbf{L}^{\{i,j\}}\mathbf{x} \geq 0$. Therefore, we have for every edge $\{i, j\}$:

$$\begin{aligned} & \mathbf{x}^{\mathrm{T}}\mathbf{L}^{\{i,j\}}\mathbf{x} = 0 \\ \Leftrightarrow \quad & |\mathbf{A}_{ij}|(\mathbf{x}_i - \operatorname{sgn}(\mathbf{A}_{ij})\mathbf{x}_j)^2 = 0 \\ \Leftrightarrow \quad & \mathbf{x}_i = \operatorname{sgn}(\mathbf{A}_{ij})\mathbf{x}_j \end{aligned}$$

In other words, two components of $\mathbf{x}$ are equal if the corresponding vertices are connected by a positive edge, and opposite to each other if the corresponding vertices are connected by a negative edge. Because the graph is connected, it follows that all $|\mathbf{x}_i|$ must be equal. We can exclude the solution $\mathbf{x}_i = 0$ for all i because $\mathbf{x}$ is not the zero vector. Without loss of generality, we assume that $|\mathbf{x}_i| = 1$ for all i.

Therefore, $\mathbf{x}$ gives a bipartition into vertices with $\mathbf{x}_i = +1$ and vertices with $\mathbf{x}_i = -1$, with the property that two vertices with the same value of $\mathbf{x}_i$ are in the same partition and two vertices with opposite sign of $\mathbf{x}_i$ are in different partitions, and therefore G is balanced. Equivalently, the signed Laplacian matrix $\mathbf{L}$ of a connected unbalanced signed graph is positive-definite. □

For a general signed graph that need not be connected, we can therefore make the following statement: The multiplicity of the eigenvalue zero equals the number of balanced connected components in G [GHKZ11].

5.4.3 Balanced Graphs

We now show how the spectrum and eigenvectors of the signed Laplacian of a balanced graph arise from the spectrum and the eigenvalues of the corresponding unsigned graph by multiplication of eigenvector components with ± 1.

Let $G = (V, E, w)$ be a balanced graph with positive and negative edge weights and $\bar{G} = (V, E, \bar{w})$ the corresponding graph with positive edge weights given by $\bar{w}(e) = |w(e)|$ for all edges $e \in E$. Let $\mathbf{A}$ and $\bar{\mathbf{A}}$ be the adjacency matrices of G and $\bar{G}$. Since G is balanced, there is a vector $\mathbf{x} \in \{-1, +1\}^{|V|}$ such that for all edges $\{i, j\}$, $\operatorname{sgn}(\mathbf{A}_{ij}) = \mathbf{x}_i\mathbf{x}_j$.

Theorem 3. *If* $\mathbf{L}$ *is the signed Laplacian matrix of the balanced graph* G *with bipartition* $\mathbf{x}$ *and eigenvalue decomposition* $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$*, then the eigenvalue decomposition of the Laplacian matrix* $\bar{\mathbf{L}}$ *of* $\bar{G}$ *of the corresponding unsigned graph* $\bar{G}$ *of* G *is given by* $\bar{\mathbf{L}} = \bar{\mathbf{U}}\mathbf{\Lambda}\bar{\mathbf{U}}^{\mathrm{T}}$ *where*

$$\bar{\mathbf{U}}_{ik} = \mathbf{x}_i \mathbf{U}_{ik}. \tag{5.14}$$

Proof. To see that $\bar{\mathbf{L}} = \bar{\mathbf{U}}\mathbf{\Lambda}\bar{\mathbf{U}}^{\mathrm{T}}$, note that for diagonal elements, we have $\bar{\mathbf{U}}_{i\bullet}\mathbf{\Lambda}\bar{\mathbf{U}}_{i\bullet}^{\mathrm{T}} = \mathbf{x}_i^2 \mathbf{U}_{i\bullet}\mathbf{\Lambda}\mathbf{U}_{i\bullet}^{\mathrm{T}} = \mathbf{U}_{i\bullet}\mathbf{\Lambda}\mathbf{U}_{i\bullet}^{\mathrm{T}} = \mathbf{L}_{ii} = \bar{\mathbf{L}}_{ii}$. For off-diagonal elements, we have $\bar{\mathbf{U}}_{i\bullet}\mathbf{\Lambda}\bar{\mathbf{U}}_{j\bullet}^{\mathrm{T}} = \mathbf{x}_i\mathbf{x}_j\mathbf{U}_{i\bullet}\mathbf{\Lambda}\mathbf{U}_{j\bullet}^{\mathrm{T}} = \mathrm{sgn}(\mathbf{A}_{ij})\mathbf{L}_{ij} = -\mathrm{sgn}(\mathbf{A}_{ij})\mathbf{A}_{ij} = -|\mathbf{A}_{ij}| = -\bar{\mathbf{A}}_{ij} = \bar{\mathbf{L}}_{ij}$.

We now show that $\bar{\mathbf{U}}\mathbf{\Lambda}\bar{\mathbf{U}}^{\mathrm{T}}$ is an eigenvalue decomposition of $\bar{\mathbf{L}}$ by showing that $\bar{\mathbf{U}}$ is orthogonal. To see that the columns of $\bar{\mathbf{U}}$ are indeed orthogonal, note that for any two column indexes $k \neq l$, we have $\bar{\mathbf{U}}_{\bullet k}^{\mathrm{T}}\bar{\mathbf{U}}_{\bullet l} = \sum_{i\in V}\bar{\mathbf{U}}_{ik}\bar{\mathbf{U}}_{il} = \sum_{i\in V}\mathbf{x}_i^2\mathbf{U}_{ik}\mathbf{U}_{il} = \mathbf{U}_{\bullet k}^{\mathrm{T}}\mathbf{U}_{\bullet l} = 0$ because $\mathbf{U}$ is orthogonal. Changing signs in $\mathbf{U}$ does not change the norm of each column vector, and thus $\bar{\mathbf{L}} = \bar{\mathbf{U}}\mathbf{\Lambda}\bar{\mathbf{U}}^{\mathrm{T}}$ is the eigenvalue decomposition of $\bar{\mathbf{L}}$. □

As shown in Section 5.4.2, the Laplacian matrix of an unbalanced graph is positive-definite and therefore its spectrum is different from that of the corresponding unsigned graph. Aggregating Theorems 2 and 3, we arrive at our main result.

Theorem 4. *The signed Laplacian matrix of a connected graph is positive-definite if and only if the graph is unbalanced.*

Proof. From Theorem 2 we know that every unbalanced connected graph has a positive-definite Laplacian matrix. Theorem 3 implies that every balanced graph has the same Laplacian spectrum as its corresponding unsigned graph. Since the unsigned Laplacian is always singular, the signed Laplacian of a balanced graph is also singular. Together, these imply that the signed Laplacian matrix of a connected graph is positive-definite if and only if the graph is unbalanced. □

The spectra of several large unipartite signed networks are plotted in Figure 5.6.

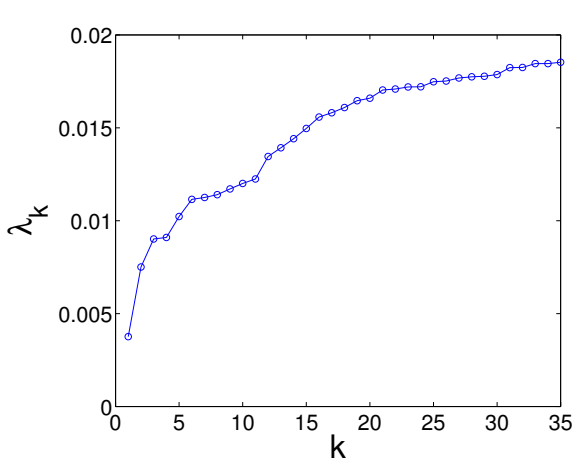
5.4.4 Algebraic Conflict

The smallest eigenvalue of the Laplacian $\mathbf{L}$ of a signed graph characterizes the amount of conflict present in the graph. We will call this number the *algebraic conflict* of the graph and denote it ξ.

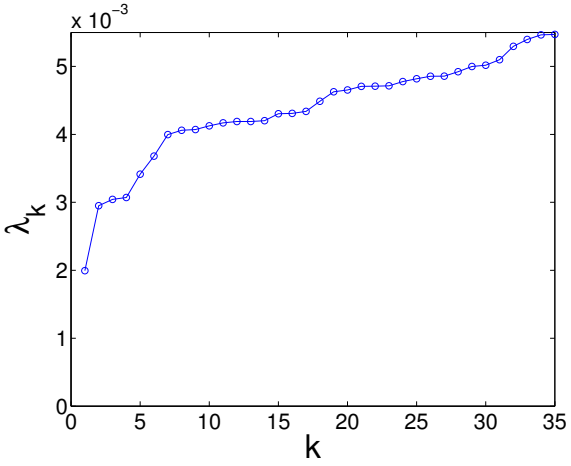
Let $G = (V, E, w)$ be a connected signed graph with adjacency matrix $\mathbf{A}$, degree matrix $\mathbf{D}$ and Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{|V|}$ be the eigenvalues of $\mathbf{L}$. Because $\mathbf{L}$ is positive-semidefinite (Theorem 1), we have $\lambda_1 \geq 0$. According to Theorem 2, λ_1 is zero exactly when G is balanced. Therefore, the value λ_1 can be used as an invariant of signed graphs that characterizes the conflict due to unbalanced cycles, i.e. cycles with an odd number of negative edges. We will call $\xi = \lambda_1$ the *algebraic conflict* of the network. The algebraic conflict is discussed in [Hou05] and [KSLL10], without being given a specific name.

The algebraic conflict ξ for several signed network datasets is compared in Table 5.1. All these large networks are unbalanced, and we can for instance observe that the social networks of the Slashdot Zoo (SZ) and Epinions (EP) are more balanced than the vote network on the English Wikipedia (EL).

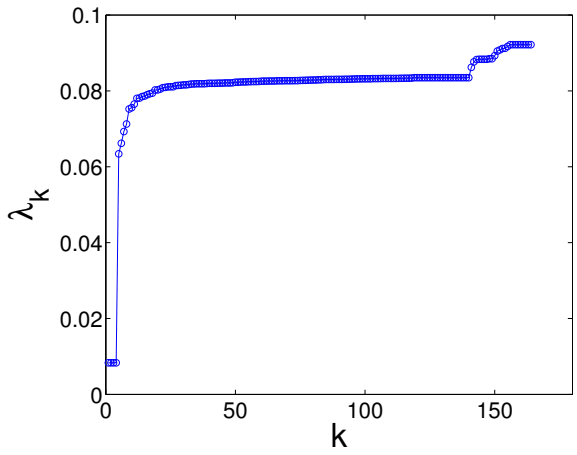
Figure 5.7 plots the algebraic conflict of the signed networks against the network size, including signed bipartite networks. The number of signed datasets is small, and we cannot make out a correlation between ξ and network size, indicating that the algebraic conflict is a meaningful measure of conflict and does not have to be normalized. This conclusion is however not very strong due to the fact that only seven signed networks were available to us.



(a) Slashdot Zoo (SZ)



(b) Epinions (EP)



(c) English Wikipedia (EL)

Figure 5.6: The Laplacian spectra of three signed networks. These plots show the eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots$ of the matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

From the definition of the algebraic conflict ξ, we can derive a simple theorem stating that adding an edge of any weight to a signed graph can only increase the algebraic conflict, not decrease it.

Theorem 5. *Let $G = (V, E, w)$ be an undirected weighted signed graph and $i, j \in V$ two vertices such that $\{i, j\} \notin E$, and λ_1 the smallest eigenvalue of the Laplacian matrix $\mathbf{L}$ of G. Furthermore, let $G' = (V, E \cup \{i, j\}, w')$ with $w'(e) = w(e)$ when $e \in E$ and $w(\{i, j\}) = w$ otherwise be the graph G to which an edge with weight w has been added. Then, let λ_1' be the smallest eigenvalue of the Laplacian matrix $\mathbf{L}'$ of G'. Then, $\lambda_1 \leq \lambda_1'$.*

Proof. We make use of a theorem stated e.g. in [Wil65, p. 97]. This theorem states that when adding a positive-semidefinite matrix $\mathbf{E}$ of rank one to a given symmetric matrix $\mathbf{A}$ with eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, the new matrix $\mathbf{A}' = \mathbf{A} + \mathbf{E}$ has eigenvalues $\lambda_1' \leq \lambda_2' \leq \ldots \leq \lambda_n'$ which interlace the eigenvalues of $\mathbf{A}$:

$$\lambda_1 \leq \lambda_1' \leq \lambda_2 \leq \lambda_2' \leq \ldots \leq \lambda_n \leq \lambda_n'$$

Table 5.1: The algebraic conflict ξ for several signed unipartite networks. Smaller values indicate a more balanced network; larger values indicate more conflict.

Network	ξ	
English Wikipedia (EL)	0.008318	Conflict
Slashdot Zoo (SZ)	0.006183	↕
Epinions (EP)	0.004438	Balance

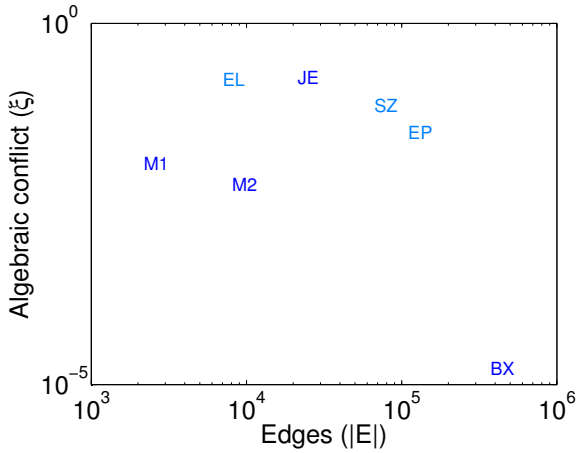


Figure 5.7: The algebraic conflict ξ, i.e. the smallest eigenvalues of the matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ in signed networks. This plot shows the algebraic conflict and the size of signed networks.

The Laplacian $\mathbf{L}'$ of G' can be written as $\mathbf{L}' = \mathbf{L} + \mathbf{E}$, where $\mathbf{E} \in \mathbb{R}^{|V|\times|V|}$ is the matrix defined by $\mathbf{E}_{ii} = \mathbf{E}_{jj} = |w|$ and $\mathbf{E}_{ij} = \mathbf{E}_{ji} = -w$, and $\mathbf{E}_{ij} = 0$ for all other entries. Then let $\mathbf{e} \in \mathbb{R}^{|V|}$ be the vector defined by $\mathbf{e}_i = \sqrt{|w|}$, $\mathbf{e}_j = -\mathrm{sgn}(w)\sqrt{|w|}$ and $\mathbf{e}_k = 0$ for all other entries. We have $\mathbf{E} = \mathbf{e}\mathbf{e}^{\mathrm{T}}$, and therefore $\mathbf{E}$ is positive-semidefinite.

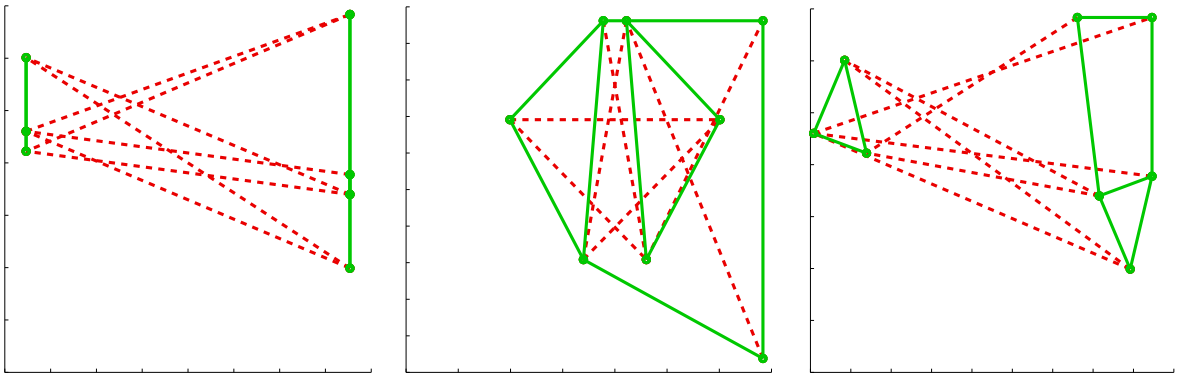
Now, due to the interlacing theorem mentioned above, adding a positive-semidefinite matrix to a given symmetric matrix can only increase each eigenvalue, but not decrease it. Therefore, $\lambda_1 \leq \lambda'_1$. □

5.4.5 Practical Considerations

An eigenvalue of zero may or may not be useful depending on the application. When choosing two eigenvalues of $\mathbf{L}$ to draw a signed graph for instance, we can distinguish two cases depending on the balance of the network:

- Case 1: If the graph is balanced, the smallest eigenvalue of $\mathbf{L}$ is zero. Unlike unsigned graphs however, the corresponding eigenvector is not constant but contains values $\{\pm 1\}$ describing the split into two partitions. If we use that eigenvector to draw the graph, the resulting drawing will place all vertices on two lines. Such an embedding may be satisfactory in cases where the perfect balance of the graph is to be visualized. If however positive edges among each partition's vertices are also to be visible, the eigenvector corresponding to the third smallest eigenvalue can be added with a small weight to the first eigenvector, resulting in a two-dimensional representation of a 3-dimensional embedding. The resulting three methods are illustrated in Figure 5.8.

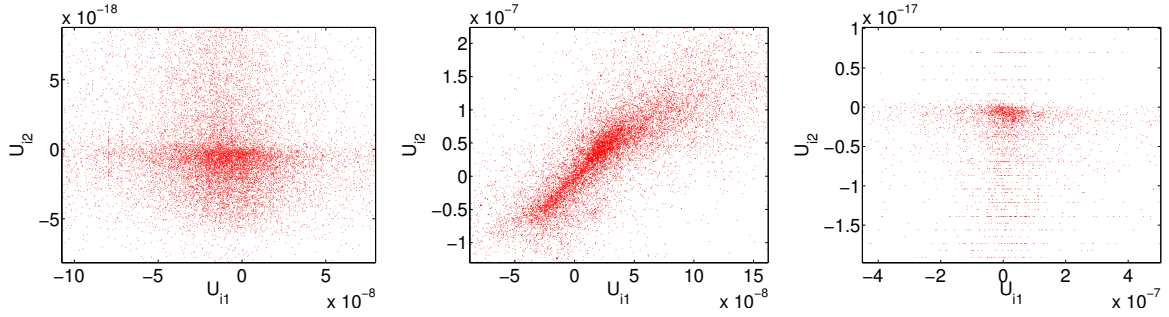
- Case 2: If the graph is unbalanced, all eigenvalues are strictly positive, and the eigenvectors corresponding to the two smallest eigenvalues give an adequate two-dimensional embedding.



(a) (λ_1, λ_2) (b) (λ_2, λ_3) (c) $(\lambda_1 + 0.3\lambda_3, \lambda_2)$

Figure 5.8: Three methods for drawing a balanced signed graph, using a small artificial example network. (a) Using the eigenvector corresponding to the smallest eigenvalue $\lambda_1 = 0$, intra-cluster structure is lost. (b) Ignoring the first eigenvalue misses important information about the clustering. (c) Using a linear combination of both methods gives a good compromise.

In practice, large graphs are almost always unbalanced as shown in Figure 5.6 and Table 5.1, and the two smallest eigenvalues give a satisfactory embedding. Figure 5.9 shows large signed networks drawn using the two eigenvectors of the smallest eigenvalues of the Laplacian matrix $\mathbf{L}$ for three signed networks.



(a) Slashdot Zoo (SZ) (b) Epinions (EP) (c) English Wikipedia (EL)

Figure 5.9: Signed spectral embedding of large networks. For each network, every node is represented as a point whose coordinates are the corresponding values in the eigenvectors of the signed Laplacian $\mathbf{L}$ corresponding to the two smallest eigenvalues.

5.5 Signed Spectral Clustering

One of the main application areas of the graph Laplacian are clustering problems. In spectral clustering, the eigenvectors of matrices associated with a graph are used to partition the vertices

of the graph into well-connected groups. In this section, we show that in a graph with negatively weighted edges, spectral clustering algorithms correspond to finding clusters of vertices connected by positive edges, but not connected by negative edges.

Spectral clustering algorithms are usually derived by formulating a minimum cut problem which is then relaxed [MS01, DGK04, Lux07, SM00, NJW01]. The choice of the cut function results in different spectral clustering algorithms. In all cases, the vertices of a given graph are mapped into the space spanned by the eigenvectors of a matrix associated with the graph.

In this section we derive a signed extension of the ratio cut, which leads to clustering with the signed Laplacian $\mathbf{L}$. Analogous derivations are possible for the normalized Laplacians. We restrict our proofs to the case of clustering vertices into two groups. Higher-order clusterings can be derived analogously.

5.5.1 Unsigned Graphs

We first review the derivation of the ratio cut in unsigned graphs, which leads to a clustering based on the eigenvectors of $\mathbf{L}$. Let $G = (V, E)$ be an unsigned graph with adjacency matrix $\mathbf{A}$. A cut of G is a partition of the vertices V into the nonempty sets V_1 and V_2, whose weight is given by

$$\mathrm{Cut}(V_1, V_2) = \sum_{i \in V_1, j \in V_2} \mathbf{A}_{ij}. \tag{5.15}$$

The cut measures how well the two clusters are connected. Since we want to find two distinct groups of vertices, the cut must be minimized. Minimizing $\mathrm{Cut}(V_1, V_2)$ however leads in most cases to solutions separating very few vertices from the rest of the graph. Therefore, the cut is usually divided by the size of the clusters, giving the ratio cut:

$$\mathrm{RatioCut}(V_1, V_2) = \left(\frac{1}{|V_1|} + \frac{1}{|V_2|} \right) \mathrm{Cut}(V_1, V_2) \tag{5.16}$$

To get a clustering, we then solve the following optimization problem:

$$\min_{V_1 \subset V} \quad \mathrm{RatioCut}(V_1, V \setminus V_1) \tag{5.17}$$

Let $V_2 = V \setminus V_1$. Then this problem can be solved by expressing it in terms of the characteristic vector $\mathbf{u} \in \mathbb{R}^{|V|}$ of V_1 defined by:

$$\mathbf{u}_i = \begin{cases} +\sqrt{|V_2|/|V_1|} & \text{if } i \in V_1 \\ -\sqrt{|V_1|/|V_2|} & \text{if } i \in V_2 \end{cases} \tag{5.18}$$

We observe that $\mathbf{u}\mathbf{L}\mathbf{u}^{\mathrm{T}} = 2|V| \cdot \mathrm{RatioCut}(V_1, V_2)$, and that $\sum_i \mathbf{u}_i = 0$, i.e. $\mathbf{u}$ is orthogonal to the constant vector. Denoting by $\mathcal{U}$ the vectors $\mathbf{u}$ of the form given in Equation (5.18) we have

$$\begin{aligned} \min_{\mathbf{u} \in \mathbb{R}^{|V|}} \quad & \mathbf{u}\mathbf{L}\mathbf{u}^{\mathrm{T}} \\ \text{s.t.} \quad & \mathbf{u} \cdot \mathbf{1} = 0, \mathbf{u} \in \mathcal{U} \end{aligned} \tag{5.19}$$

This can be relaxed by removing the constraint $\mathbf{u} \in \mathcal{U}$, giving as solution the eigenvector of $\mathbf{L}$ having the smallest nonzero eigenvalue [Lux07].

5.5.2 Signed Graphs

We now give a derivation of the ratio cut for signed graphs. Let $G = (V, E, w)$ be a signed graph with adjacency matrix $\mathbf{A}$. We write $\mathbf{A}^{\oplus}$ and $\mathbf{A}^{\ominus}$ for the adjacency matrices containing only the weights of positive and negative edges. In other words, $\mathbf{A}^{\oplus}_{ij} = \max(0, \mathbf{A}_{ij})$, $\mathbf{A}^{\ominus}_{ij} = \max(0, -\mathbf{A}_{ij})$ and $\mathbf{A} = \mathbf{A}^{\oplus} - \mathbf{A}^{\ominus}$.

For convenience we define positive and negative cuts that only count positive and negative edges respectively:

$$\mathrm{Cut}^{\oplus}(V_1, V_2) = \sum_{i \in V_1, j \in V_2} \mathbf{A}^{\oplus}_{ij} \tag{5.20}$$

$$\mathrm{Cut}^{\ominus}(V_1, V_2) = \sum_{i \in V_1, j \in V_2} \mathbf{A}^{\ominus}_{ij} \tag{5.21}$$

In these definitions, we allow V_1 and V_2 to be overlapping. For a vector $\mathbf{u} \in \mathbb{R}^{|V|}$, we consider the bilinear form $\mathbf{u}^{\mathrm{T}}\mathbf{L}\mathbf{u}$. As shown in Equation (5.11), this can be written in the following way:

$$\mathbf{u}^{\mathrm{T}}\mathbf{L}\mathbf{u} = \sum_{\{i,j\} \in E} |\mathbf{A}_{ij}| (\mathbf{u}_i - \mathrm{sgn}(\mathbf{A}_{ij})\mathbf{u}_j)^2$$

For a given partition $V = V_1 \dot{\cup} V_2$, let $\mathbf{u} \in \mathbb{R}^{|V|}$ be the following vector:

$$\mathbf{u}_i = \begin{cases} +\frac{1}{2}\left(\sqrt{\frac{|V_1|}{|V_2|}} + \sqrt{\frac{|V_2|}{|V_1|}}\right) & \text{if } i \in V_1 \\ -\frac{1}{2}\left(\sqrt{\frac{|V_1|}{|V_2|}} + \sqrt{\frac{|V_2|}{|V_1|}}\right) & \text{if } i \in V_2 \end{cases} \tag{5.22}$$

The corresponding bilinear form then becomes:

$$\begin{aligned} \mathbf{u}^{\mathrm{T}}\mathbf{L}\mathbf{u} &= \sum_{\{i,j\} \in E} |\mathbf{A}_{ij}| \left(\mathbf{u}_i - \mathrm{sgn}(\mathbf{A}_{ij})\mathbf{u}_j\right)^2 \\ &= |V| \left(\frac{1}{|V_1|} + \frac{1}{|V_2|}\right) \left(2 \cdot \mathrm{Cut}^{\oplus}(V_1, V_2) + \mathrm{Cut}^{\ominus}(V_1, V_1) + \mathrm{Cut}^{\ominus}(V_2, V_2)\right) \end{aligned}$$

This leads us to define the following signed cut of (V_1, V_2):

$$\mathrm{SignedCut}(V_1, V_2) = 2 \cdot \mathrm{Cut}^{\oplus}(V_1, V_2) + \mathrm{Cut}^{\ominus}(V_1, V_1) + \mathrm{Cut}^{\ominus}(V_2, V_2) \tag{5.23}$$

and to define the signed ratio cut as follows:

$$\mathrm{SignedRatioCut}(V_1, V_2) = \left(\frac{1}{|V_1|} + \frac{1}{|V_2|}\right) \mathrm{SignedCut}(V_1, V_2) \tag{5.24}$$

Therefore, the following minimization problem solves the signed clustering problem:

$$\min_{V_1 \subset V} \quad \mathrm{SignedRatioCut}(V_1, V \setminus V_1) \tag{5.25}$$

We can now express this minimization problem using the signed Laplacian, where $\mathcal{U}$ denotes the set of vectors of the form given in Equation (5.22):

$$\begin{aligned} \min_{\mathbf{u} \in \mathbb{R}^{|V|}} \quad & \mathbf{u}\mathbf{L}\mathbf{u}^{\mathrm{T}} \\ \text{s.t.} \quad & \mathbf{u} \in \mathcal{U} \end{aligned} \tag{5.26}$$

Note that we lose the orthogonality of $\mathbf{u}$ to the constant vector. This can be explained by the fact that if G contains negative edges, the smallest eigenvector can always be used for clustering: If G is balanced, the smallest eigenvalue is zero and its eigenvector equals (± 1) and gives the two clusters separated by negative edges. If G is unbalanced, then the smallest eigenvalue of $\mathbf{L}$ is larger than zero by Theorem 2, and the constant vector is not an eigenvalue.

The signed cut $\text{SignedCut}(V_1, V_2)$ counts the number of positive edges that connect the two groups V_1 and V_2, and the number of negative edges that remain in each of these groups. Thus, minimizing the signed cut leads to clusterings where two groups are connected by few positive edges and contain few negative edges inside each group. This signed ratio cut generalizes the ratio cut of unsigned graphs and justifies the use of the signed Laplacian $\mathbf{L}$ for spectral clustering of signed graphs.

5.5.3 Signed Clustering using Other Matrices

When instead of normalizing with the number of vertices $|V_1|$ we normalize with the number of edges $\text{vol}(V_1)$, the result is a spectral clustering algorithm based on the eigenvectors of $\mathbf{D}^{-1}\mathbf{A}$ introduced by Shi and Malik [SM00]. The cuts normalized by $\text{vol}(V_1)$ are called normalized cuts. In the signed case, the eigenvectors of $\mathbf{D}^{-1}\mathbf{A}$ lead to the signed normalized cut:

$$\text{SignedNormalizedCut}(V_1, V_2) = \left(\frac{1}{\text{vol}(V_1)} + \frac{1}{\text{vol}(V_2)}\right) \text{SignedCut}(V_1, V_2) \tag{5.27}$$

A similar derivation can be made for normalized cuts based on $\mathbf{N} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, generalizing the spectral clustering method of Ng, Jordan and Weiss [NJW01]. The dominant eigenvector of the signed adjacency matrix $\mathbf{A}$ can also be used for signed clustering [BL04]. As in the unsigned case, this method is not suited for very sparse graphs, and does not have an interpretation in terms of cuts. The following section gives an example of clustering a small, signed graph.

5.5.4 Anthropological Example

As an application of signed spectral clustering to real-world data, we show the dataset of [Rea54]. This dataset describes the relations between sixteen tribal groups of the Eastern Central Highlands of New Guinea [HH83]. Relations between tribal groups in the Gahuku–Gama alliance structure can be friendly (*rova*) or antagonistic (*hina*). We model the dataset as a graph with edge weights $+1$ for friendship and -1 for enmity.

The resulting graph contains cycles with an odd number of negative edges, and therefore its signed Laplacian matrix is positive-definite. We use the eigenvectors of the two smallest eigenvalues ($\lambda_1 = 1.04$ and $\lambda_2 = 2.10$) to embed the graph into the plane. The result is shown in Figure 5.10. We observe that indeed the positive (green) edges are short, and the negative (red) edges are long. Looking at only the positive edges, the drawing makes the two connected components easy to see. Looking at only the negative edges, we recognize that the tribal groups can be clustered into three groups, with no negative edges inside any group. These three groups correspond indeed to a higher-order grouping in the Gahuku–Gama society [HH83]. An example on a larger network is shown in the next section, using the genre of movies in a user–item graph with positive and negative ratings.

5.6 Link Sign Prediction

In this section, we introduce the link sign prediction problem, and show how it can be solved using the signed adjacency matrix $\mathbf{A}$ and the signed Laplacian matrix $\mathbf{L}$. Since a signed network

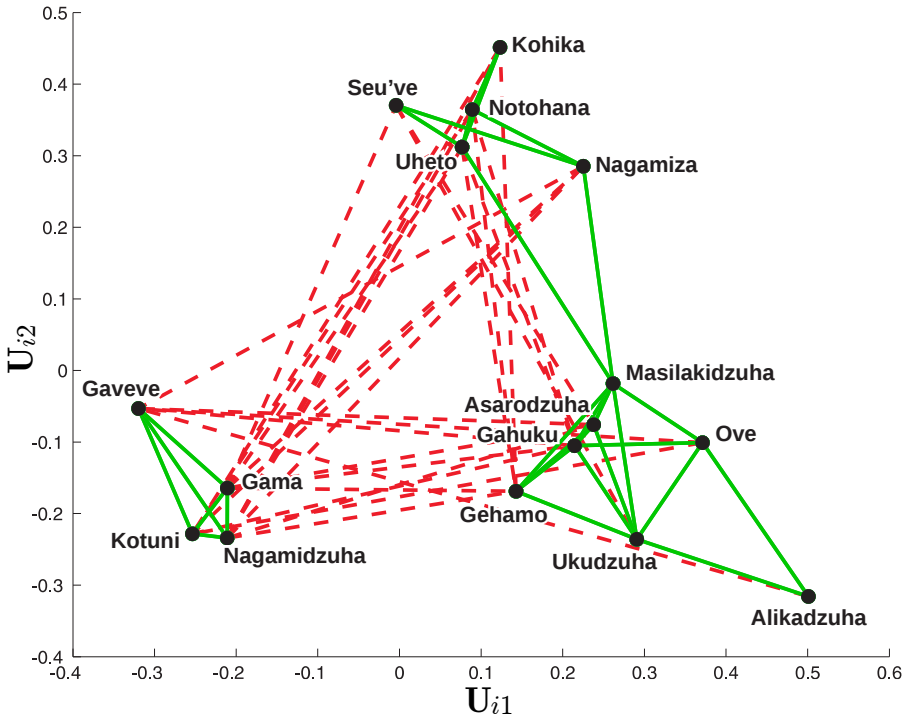


Figure 5.10: The tribal groups of the Eastern Central Highlands of New Guinea from the study of Read [Rea54] drawn using signed Laplacian graph embedding. Individual tribes are shown as vertices of the graphs, with friendly relations shown as green edges and antagonistic relations shown as red edges. The three higher-order groups as described by Hage & Harary in [HH83] are linearly separable.

contains two types of edges, we will not look at the problem of link prediction, but at the problem of link sign prediction. In the link sign prediction problem, a network of signed edges is given, along with a set of edges for which the sign must be predicted. Thus, the task is not to predict which edge is going to appear in a network, but what kind of edge will appear between two given nodes.

The adjacency kernels defined in Section 3.3.3 and the Laplacian kernels defined in Section 4.2.3 were introduced there for unsigned graphs. We will now extend these to signed graphs.

5.6.1 Functions of the Signed Adjacency Matrix

The signed adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$ can be used to define link sign prediction functions. The interpretation is then similar to that of the unsigned case: a weighted sum over all paths between two nodes. However, in the signed case, paths are additionally weighted by their sign, defined as the product of edge weights.

We will first define the weight of a path as the product of its edge weights. The weight of a n-path $p = (i_1, i_2, \ldots, i_{n+1})$ is $w(p) = \prod_{k=1}^{n} \mathbf{A}_{i_k i_{k+1}}$. Since edge weights are positive and negative, the weight of a path can be positive or negative in accordance with the multiplication rule described at the beginning of this chapter. Consider the square $\mathbf{A}^2$. Its entries are given by

$$(\mathbf{A}^2)_{ij} = \sum_k \mathbf{A}_{ik}\mathbf{A}_{kj}$$

In other words, $(\mathbf{A}^2)_{ij}$ contains a sum over 2-paths between i and j, weighted by their path weight. In the simple case of edge weights in $\{-1, +1\}$, $(\mathbf{A}^2)_{ij}$ equals the number of positive 2-paths minus the number of negative 2-paths between i and j. This extends the triangle closing model as a link prediction method to signed networks.

Analogously, the power $\mathbf{A}^k$ contains the sum of all k-paths between any node pair, weighted by path weights. Since the spectral transformations of $\mathbf{A}$ that we consider can be expressed as a power sum over $\mathbf{A}$, they can be interpreted as a sum of all paths between any two nodes, weighted by the product of their path weight and a function of their length. This extends path counting as a link prediction method to signed networks. As in the unsigned case, it follows that any power series with nonnegative and nonincreasing coefficients is a suitable link prediction function. We present two special cases:

Signed Exponential Kernel As for unsigned graphs, the matrix exponential is a special case of path counting, and can be used as a graph kernel.

$$K_{\mathrm{EXP}}(\mathbf{A}) = \exp(\alpha\mathbf{A}) = \sum_{k=0}^{\infty} \frac{\alpha^k}{k!}\mathbf{A}^k \tag{5.28}$$

$\alpha > 0$ is a parameter that has to be learned using the curve-fitting method described in Section 4.2.

Signed Neumann Kernel The Neumann kernel is defined analogously to the unsigned case:

$$K_{\mathrm{NEU}}(\mathbf{A}) = (\mathbf{I} - \alpha\mathbf{A})^{-1} = \sum_{k=0}^{\infty} \alpha^k\mathbf{A}^k \tag{5.29}$$

As in the exponential kernel, α is a parameter that can be learned by curve fitting. For the Neumann kernel to be well-defined we must require $\alpha^{-1} > |\lambda_1|$, where $|\lambda_1|$ is the largest absolute eigenvalue of $\mathbf{A}$. Otherwise, the underlying spectral transformation function $1/(1 - \alpha\lambda)$ would result in negative values for the largest eigenvalues.

5.6.2 Functions of the Normalized Adjacency Matrix

Analogously, all functions of the normalized adjacency matrix $\mathbf{N}$ given in Section 3.3 can be applied to signed networks. The normalized version of the signed exponential and Neumann kernels are given by:

$$K_{\mathrm{EXP}}(\mathbf{N}) = \exp(\alpha\mathbf{N}) = \sum_{k=0}^{\infty} \frac{\alpha^k}{k!}\mathbf{N}^k \tag{5.30}$$

$$K_{\mathrm{NEU}}(\mathbf{N}) = (\mathbf{I} - \alpha\mathbf{N})^{-1} = \sum_{k=0}^{\infty} \alpha^k\mathbf{N}^k \tag{5.31}$$

A power series of $\mathbf{N}$ corresponds to normalized path counting:

$$p(\mathbf{N}) = \sum_{k=0}^{\infty} \alpha_k\mathbf{N}^k \tag{5.32}$$

5.6.3 Signed Laplacian Kernels

We can apply the Laplacian kernels as defined in Section 4.2.3 to signed graphs, giving signed Laplacian kernels. These kernels are all functions of the Laplacian matrix $\mathbf{L}$ and extend readily to signed graphs by using the signed Laplacian matrix $\mathbf{L}$ as given in Definition 3. The normalized versions of these kernels also exist, and are based on the matrix $\mathbf{Z}$. As shown in Section 5.4.1, the signed Laplacian matrix $\mathbf{L}$ is positive-definite and can therefore be used as the basis of kernels.

$$K_{\mathrm{COM}}(\mathbf{L}) = \mathbf{L}^{+} \tag{5.33}$$

$$K_{\mathrm{COM}}(\mathbf{Z}) = \mathbf{Z}^{+} \tag{5.34}$$

As noted in Section 5.4, $\mathbf{L}$ is positive-definite for unbalanced signed graphs, in which case the pseudoinverse reduces to the ordinary matrix inverse, because all eigenvalues of $\mathbf{L}$ are then different from zero. The signed Laplacian graph kernel $\mathbf{L}^{+}$ can also be interpreted as the signed resistance distance kernel [KSB+08]. A separate derivation of the signed resistance distance kernel is given in the next section.

By regularization of the commute-time kernels we arrive at the signed regularized Laplacian kernels:

$$K_{\mathrm{REG}}(\mathbf{L}) = (\mathbf{I} + \alpha\mathbf{L})^{-1} = \sum_{k=0}^{\infty} (-\alpha)^k \mathbf{L}^k \tag{5.35}$$

$$K_{\mathrm{REG}}(\mathbf{Z}) = (\mathbf{I} + \alpha\mathbf{Z})^{-1} = \sum_{k=0}^{\infty} (-\alpha)^k \mathbf{Z}^k \tag{5.36}$$

We extend the heat diffusion kernel to signed graphs giving

$$K_{\mathrm{HEAT}}(\mathbf{L}) = \exp(-\alpha\mathbf{L}) = \sum_{k=0}^{\infty} \frac{(-\alpha)^k}{k!} \mathbf{L}^k \tag{5.37}$$

$$K_{\mathrm{HEAT}}(\mathbf{Z}) = \exp(-\alpha\mathbf{Z}) = \sum_{k=0}^{\infty} \frac{(-\alpha)^k}{k!} \mathbf{Z}^k \tag{5.38}$$

By considering power series of $\mathbf{L}^{+}$ and $\mathbf{I} - \mathbf{Z}$, we arrive and the signed generalized Laplacian kernels:

$$K_{\mathrm{GEN}}(\mathbf{L}) = p(\mathbf{L})^{+} = \left(\sum_{k=0}^{\infty} \alpha_k \mathbf{L}^k \right)^{-1} \tag{5.39}$$

$$K_{\mathrm{GEN}}(\mathbf{Z}) = p(\mathbf{I} - \mathbf{Z}) = \sum_{k=0}^{\infty} \alpha_k (\mathbf{I} - \mathbf{Z})^k \tag{5.40}$$

5.6.4 Experiments

The signed graph kernels can be used for link prediction analogously to the unsigned kernels, implementing the multiplication rule for edge weights. To evaluate their performance at link prediction, we use the signed unipartite networks in our network collection. Four networks in our collection are signed and unipartite:

- The Slashdot Zoo (SZ) is a social network consisting of the relationships *friend* and *foe*. These relations are directed and their opposites are called *fan* and *freak* [KLB09].

- Epinions (EP) is a signed trust network, containing directed trust and distrust links [MA05].
- Líbímseti.cz (LI) is a dating community where users can rate each other's profile. Ratings range from 1 to 10. We subtract the overall mean rating giving positive and negative ratings [BP07].
- The English Wikipedia (EL) votes consists of users that vote for and against each other in administration votes [LHK10a].

As the task for evaluation, we choose the prediction of link sign. The setup is the same as in the unsigned case described in Section 4.1.3, with the difference that we do not try to predict whether a node is present in the test set, but only its weight. In previous evaluations using unsigned networks, we used a test set of edges known to be present in the network and a *zero test set* of node pairs known to not correspond to any edges. These can also be called *non-edges*. Scores are then computed for all edges and non-edges in the test and zero test sets. The mean average precision is then used to compare the computed scores to each edge's or non-edge's membership in the test set and zero test set. Because in the signed case edges are already weighted, we do not need a zero test set of non-edges. Instead, we split the test set of edges into positive and negative edges by their weight, and use the mean average precision to compare the computed scores to each test edge's original sign. In networks with weighted edges that do not assume negative values such as the ten point rating scale ($\{1, \ldots, 10\}$) of Líbímseti.cz (LI), we count an edge as positive when its weight is above or equal to the overall mean edge weight, and as negative otherwise. As link prediction functions, we use the adjacency and Laplacian kernels described in Table 4.4, using the signed variants of the graph characteristic matrices $\mathbf{A}$, $\mathbf{N}$, $\mathbf{L}$ and $\mathbf{Z}$.

Figure 5.11 gives an example of the curve fitting method for signed graphs. The evaluation results are summarized in Figure 5.12, and full evaluation results are given in Appendix B. The best performing spectral transformations for all unipartite signed network datasets is given in Table 5.2.

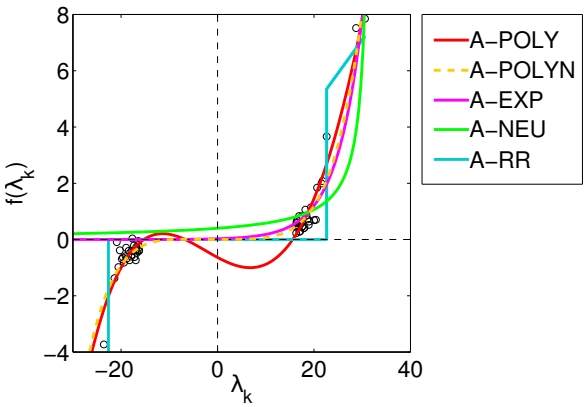


Figure 5.11: Curve fitting in networks with signed edge weights. The plot represents the curve fitting problem described in Section 4.2.2, applied to the signed network of the Slashdot Zoo (SZ) using the adjacency matrix $\mathbf{A}$.

Observations Due to the small number of signed unipartite networks available in this study, we cannot make any statistically significant statement about which link sign prediction performs best. However, an examination of the results of Figure 5.12 suggests that the various link

Table 5.2: The best performing link prediction functions for the signed unipartite datasets. For each dataset, we show the source and target matrices, the curve fitting model and the link prediction method that performs best.

Dataset	Best transformation	Best method	MAP
Slashdot Zoo (SZ)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-POLYN	0.397
Epinions (EP)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-POLY	0.305
Líbímseti.cz (LI)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-POLY	0.636
English Wikipedia (EL)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-EXP	0.828

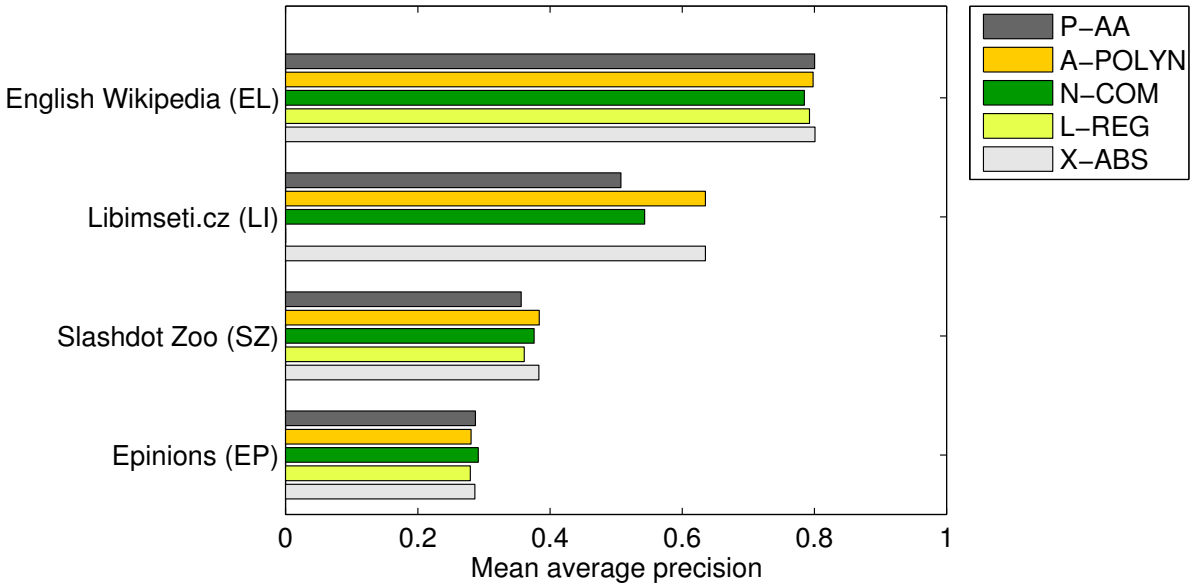


Figure 5.12: The mean average precision of link signed prediction methods on unipartite, signed networks. This figure shows a selection of link sign prediction methods, one representative method for each group.

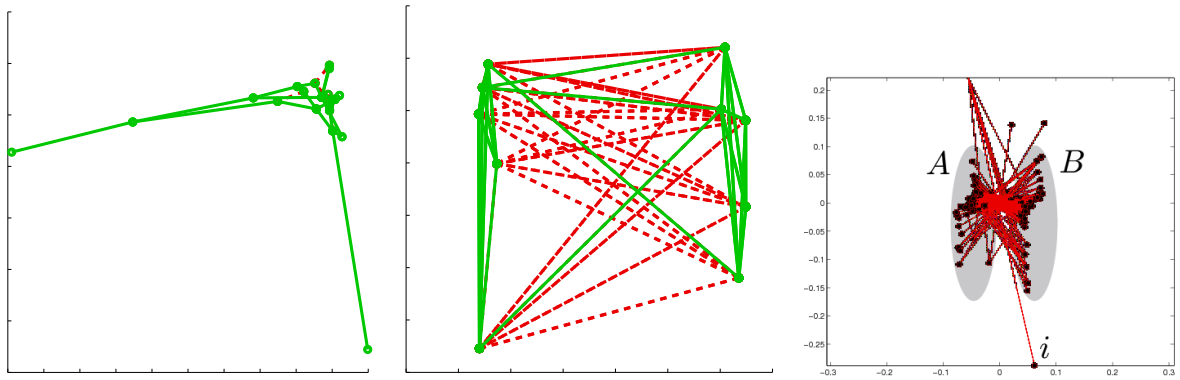
prediction methods applied to unweighted networks perform similarly at the task of link sign prediction in signed networks.

5.7 Interpretation of Conflict and Centrality

In this section, we argue that a signed spectral drawing confers information about the conflict as well as about the centrality of a network, and propose a method to visualize the conflict information independently of the centrality information.

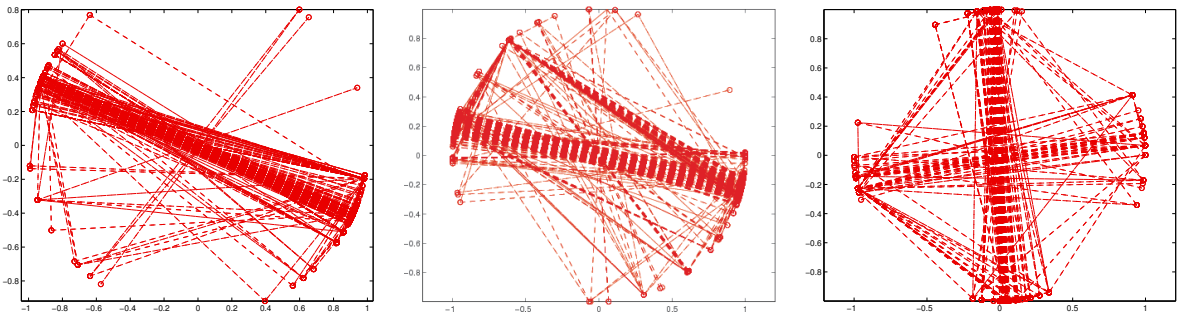
In unsigned networks, spectral drawings show the centrality of nodes: Vertices connected to many other vertices by short paths are placed centrally, whereas badly connected nodes are placed near the edge of the drawing. On the other hand, conflict networks often contain clusters of users connected by negative edges. In order to separate these clusters visually, they should be placed at a certain distance from each other, as illustrated in Figure 5.13. These two requirements are in opposition to each other when we consider users that could be called extreme, i.e. those that have many negative incident edges: They should be placed centrally according to the first requirement, and decentrally according to the second requirement.

Observing the actual embedding produced by using the signed Laplacian $\mathbf{L}$ in Figure 5.13, we see that both requirements are implemented by our signed spectral graph drawing method: The decentral node i is drawn at the edge of the plot, and two antagonistic groups of users A and B are visible. However, the two groups are placed more centrally than i, being better connected with the graph as a whole. If our goal is to visualize the conflict, these two clusters should be placed further from the center of the drawing. For these reasons, we propose to project all points onto the unit circle, discarding the centrality information, and only retaining the conflict information. The resulting drawings are shown in Figure 5.14. We observe some graphs to have two, three, four or no main clusters in conflict with each other.



(a) Centrality (b) Conflict (c) Combination

Figure 5.13: Two conflicting requirements leading to suboptimal drawings: (a) Badly-connected vertices placed at the periphery of the drawing (artificial network), using eigenvectors of the Laplacian $\bar{\mathbf{L}}$ of the underlying unweighted graph $\bar{G}$. (b) Large distance between main conflict cluster (artificial network), using eigenvectors of the Laplacian $\mathbf{L}$. (c) Result: Distance between conflict clusters is less than between outliers (Collaboration network of Wikipedia article *Toilets in Japan* [BKLR09]), using eigenvectors of the Laplacian $\mathbf{L}$. This last network contains only negative edges.



(a) This Charming Man (b) Multiculturalism (c) Criticism of Prem Rawat

Figure 5.14: Drawing signed graphs by projecting points on the unit circle, showing networks with (a) two, (b) three and (c) four conflict clusters. Each plots shows the collaboration network of a single article of the English Wikipedia, as described in [BKLR09].

5.8 Alternative Derivations

This section presents two alternative derivations of the signed Laplacian matrix, using electrical networks and Markov random fields. The purpose of these derivations is to demonstrate that the Laplacian $\mathbf{L}$ of a signed graph arises naturally in multiple independent ways.

5.8.1 Signed Resistance Distance

The unsigned resistance distance and heat diffusion kernels are justified by physical interpretations. In the presence of signed edge weights, these interpretations are still valid using the following formalism, which we describe in terms of electrical resistance networks.

A positive electrical resistance indicates that the potentials of two connected nodes will tend to each other: The smaller the resistance, the more both potentials approach each other. Therefore, an edge with positive weight w can be represented as a resistor with a resistance value of $1/w$. If the edge has negative weight $-w$, we can interpret the connection as consisting of a resistor of the corresponding positive weight $1/w$ in series with an *inverting amplifier* that guarantees its ends to have opposite voltage, as depicted in Figure 5.15. In other words, two nodes connected by a negative edge will tend to opposite voltages.

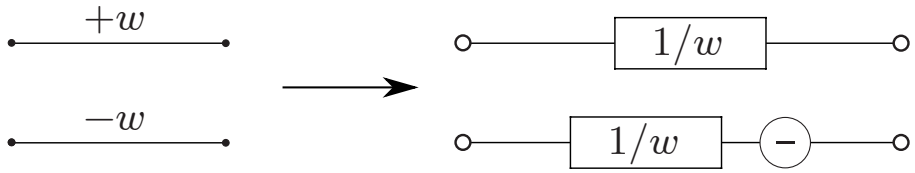


Figure 5.15: An edge with a negative weight is interpreted as a positive resistor in series with an inverting component.

Thus, a positive edge with weight $+w$ is modeled by a resistor of resistance $1/w$ and a negative edge with weight $-w$ is modeled by a resistor of resistance $1/w$ in series with a (hypothetical) electrical component that assures its ends have opposite electrical potential. Electrical networks with such edges can be analysed in the following way: In an electrical network with adjacency matrix $\mathbf{A}$, let $\mathbf{u}_i$ be the electric potential of node i. For each i, we can write $\mathbf{u}_i$ as a mean of the potentials of i's neighbors, weighted by edge weights.

$$\mathbf{u}_i = \left(\sum_{\{i,j\} \in E} \mathbf{A}_{ij} \right)^{-1} \sum_{\{i,j\} \in E} \mathbf{A}_{ij} \mathbf{u}_j$$

If edges have negative weights, the inversion of potentials results in the following equation:

$$\mathbf{u}_i = \left(\sum_{\{i,j\} \in E} |\mathbf{A}_{ij}| \right)^{-1} \sum_{\{i,j\} \in E} \mathbf{A}_{ij} \mathbf{u}_j$$

Thus, electrical networks give the equation $\mathbf{Du} = \mathbf{Au}$ and the matrices $\mathbf{D}^{-1}\mathbf{A}$ and $\mathbf{D} - \mathbf{A}$. Because such an inverting amplifier needs the definition of a specific value of zero voltage, the resulting model loses one degree of freedom, explaining that in the general case the Laplacian $\mathbf{L}$ has rank one greater than the Laplacian $\bar{\mathbf{L}}$ of the underlying unsigned graph. This is reflected in the fact that zero is not an eigenvalue of $\mathbf{L}$ for signed graphs in general.

5.8.2 Markov Random Fields

It is possible to derive the signed Laplacian matrix by considering Markov random fields over bipartite, signed graphs. A Markov random field is a network of random variables with the property that each random variable is directly dependent only on its immediate neighbors [Roz83, Pea88]. Markov random fields are used for instance in computer graphics for image segmentation, image enhancement and reconstruction of missing image parts [CJ93].

In general, Markov random fields can be used whenever a graph or network is given and a prediction has to be made for missing values. The graph underlying a Markov random field is called a Markov network. Let $X = \{X_1, \ldots, X_n\}$ be a set of random variables. A Markov random field is given by:

- A graph $G = (X, E)$ where the vertices are the random variables and edges represent dependencies between the random variables.
- The set C of all cliques of maximal size in G. A clique in G is a subset of X with an edge between all pairs of vertices. A clique is maximal if no vertex can be added retaining the clique property.
- For each maximal clique $c \in C$, a potential function φ_c only dependent on the random variables corresponding to the vertices in c. φ_c is an indicator of the coherence of the global state: A global state is more likely if the potential functions have high values.

The global joint probability distribution of the Markov random field is given by:

$$P(X = x) = \left(\sum_y \exp \sum_{c \in C} \varphi_c(y) \right)^{-1} \exp \sum_{c \in C} \varphi_c(x) \tag{5.41}$$

An alternative definition moves the exponential inside the sum turning it into a product. The potentials φ_c may take on negative values. A set of random variables whose joint probability can be expressed in this way is a Markov random field. The corresponding graph is the Markov network. In the typical prediction scenario, some nodes have known values, and the values of others must be predicted. Finding a global optimum for the assignments of the variables X is hard, and one therefore resorts to stochastic relaxation [Roz83, Pea88]. Correspondingly, the search for an optimal assignment of unknown ratings is done using an iterative algorithm: At each step, the value of each variable is updated to maximize the joint probability of the global assignment. The iteration stops once the assigned values reach a stable state where no single-node update can increase the joint probability.

Algorithm 1 described below represents the basic method used to find a global state having high probability. The algorithm will stop when the joint probability cannot be increased by local changes to single node states.

The Markov Network of Ratings

As an application of Markov random fields, assume a signed, bipartite graph of ratings between users and items. In this section, we will depart from the notational conventions used elsewhere in this thesis and use the notation that is usual for random variables. We define two sets of random variables, those corresponding to users and those corresponding to items:

$$X = \{U_1, \ldots, U_m, I_1, \ldots, I_n\} \tag{5.42}$$

Given a starting user U_1, we let each variable $X_i \in X$ represent the agreement between user U_1's taste and X_i. If X_i is a user vertex, the state of X_i represents the correlation between U_1's

Input: a graph $G = (X, E)$ with given clique potentials
Output: an assignment x of states to nodes in X
initialize all x_i for all random variables X_i
repeat
for all unknown nodes x_i **do**
Update x_i in function of adjacent node states and of incident clique potential functions
end for
until x converges

Algorithm 1: An iterative algorithm used to find a probable global state of a Markov network.

and X_i's tastes. For an item I_j, the value of the corresponding random variable represents the rating U_1 would give to I_j.

Ratings in collaborative filtering databases are usually constrained to a set of numbers with the highest value representing a good rating and the smallest value represents a bad rating. We will assume the ratings are scaled to the range $[-1, +1]$. From this interval, a finite set $K \subset [-1, +1]$ of values is chosen as possible states of the random variables. We first perform the calculation as if the variables were continuous in $[-1, +1]$ and then show how to map values back to discrete states. The random variable corresponding to U_1 is assigned the fixed value of $+1$, representing maximal agreement.

In a bipartite rating graph, the set of maximum cliques corresponds exactly to the set of edges. Each maximal clique contains two adjacent vertices. For each edge $\{i, j\}$, a potential function $\varphi_{\{i,j\}}$ must be given. The potential function is an indicator of the probability for adjacent vertices of taking specific values in function of the edge rating. If users would not only rate items but also other users, we could analyse known rating patterns probabilistically and design a potential function that models available data accurately. However, because user ratings of other users are not available, we will formulate a list of requirements a potential function must fulfill and then find a suitable function.

- The potential must only depend on the edge it covers and the two random variables whose vertices are incident to the edge. The edge will be represented here only by its rating value.
- The potential should be proportional to the absolute value of the rating. Extremal ratings of value $+1$ and -1 are considered stronger than neutral or near-neutral ratings.
- The potential should decrease with the absolute value of the node differences if the rating is positive and increase with it if the rating is negative.

Let w be the rating of an edge, and a and b the value of the two adjacent random variables. If $w = 1$, then a suitable potential function is given by the negated square difference $-(a - b)^2$. If $w = -1$, taking the sum instead of the difference gives the compliant expression $-(a + b)^2$. Both cases can be merged into the potential function $-(a - \mathrm{sgn}(w) \cdot b)^2$. Considering the second requirement, we finally arrive at a form that also covers the case $w = 0$:

$$\varphi_w(a, b) = -|w| \, (a - \mathrm{sgn}(w) \cdot b)^2 \tag{5.43}$$

Note that $\varphi_w(a, b) = \varphi_w(b, a)$. As we will see below, taking the square will simplify later calculations. At each iteration, we search an assignment x'_i to the random variable X_i that

maximizes the joint probability of X.

$$\begin{aligned}
x_i' &= \operatorname{argmax}_{x_i} P(X = x) \\
&= \operatorname{argmax}_{x_i} \left(\sum_y \exp \sum_{c \in C} \varphi_c(y) \right)^{-1} \exp \sum_{c \in C} \varphi_c(x) \\
&= \operatorname{argmax}_{x_i} \sum_{c \in C} \varphi_c(x) \\
&= \operatorname{argmax}_{x_i} \sum_j -|w_{ij}|(x_i - \operatorname{sgn}(w_{ij}) \cdot x_j)^2 \\
&= \operatorname{argmin}_{x_i} x_i^2 \sum_j |w_{ij}| - 2x_i \sum_j w_{ij} x_j + |w_{ij}| \sum_j x_j^2
\end{aligned}$$

This is a polynomial of the form $ax_i^2 + bx_i + c$. Its coefficient a is only zero if all ratings attached to x_i are neutral, in which case the node cannot be part of the calculation. Otherwise, a is strictly positive, and the polynomial takes its global minimum at $-b/2a$, which gives

$$x_i' = \frac{\sum_j w_{ij} x_j}{\sum_j |w_{ij}|}. \tag{5.44}$$

This expression corresponds to a weighted sum of the values x_j, with weights w_{ij}. This weighted sum includes negative weights w_{ij} for some pairs $\{i, j\}$, and uses the sum of absolute weights in the denominator. As a result, Equation (5.44) corresponds to multiplication of x with the matrix $\mathbf{D}^{-1}\mathbf{A}$, since $\mathbf{D}$ itself is defined using the sums of absolute edge weights. Therefore this algorithm converges to the dominant eigenvector of $\mathbf{D}^{-1}\mathbf{A}$. This result justifies the definition of the degree matrix $\mathbf{D}$ using the sum of absolute edge weights.

5.9 Related Work

The degree matrix $\mathbf{D}$ of a signed graph is defined in this chapter using $\mathbf{D}_{ii} = \sum_j |\mathbf{A}_{ij}|$. In some contexts, an alternative degree matrix $\mathbf{D}_{\text{alt}}$ is defined without the absolute value:

$$(\mathbf{D}_{\text{alt}})_{ii} = \sum_j \mathbf{A}_{ij} \tag{5.45}$$

The leads to an alternative Laplacian matrix $\mathbf{L}_{\text{alt}} = \mathbf{D}_{\text{alt}} - \mathbf{A}$ for signed graphs that is not positive-semidefinite. This Laplacian is used in the context of knot theory [LW00], to draw graphs with negative edge weights [KCH02], and to implement constrained clustering, i.e. clustering with *must-link* and *must-not-link* edges [Dav09]. Since $\mathbf{L}_{\text{alt}}$ is not positive-semidefinite in the general case, it cannot be used as a kernel.

If all edges of a graph have negative weights, the matrix $\mathbf{D} + \mathbf{A}$ may be considered, and corresponds to the Laplacian of the underlying unsigned graph [DR94]. However, all large real-world networks contain both positive and negative edges, making this approach only suitable for small isolated networks such as the revert network of individual Wikipedia articles [BKLR09].

Expressions of the form $(\sum_i |\mathbf{w}_i|)^{-1} \sum_i \mathbf{w}_i \mathbf{x}_i$ appeared several times in the preceding sections. These types of expressions represent a weighted mean of the values $\mathbf{x}_i$, supporting negative values of the weights $\mathbf{w}_i$. In fact, these expressions have been used for some time in the collaborative filtering literature without being connected to the signed Laplacian [SKKR01].

5.10 Summary: Signed Graphs

In this chapter, we have introduced the Laplacian matrix $\mathbf{L}$ for signed graphs and used it for building graph kernels, for solving the signed spectral clustering problem, and for drawing signed graphs. We have shown that the signed Laplacian arises naturally in multiple unrelated areas, justifying its importance.

The various applications presented in this chapter show that signed graphs appear in many diverse spectral graph mining applications, and that they can be approached by defining the Laplacian matrix $\mathbf{L}$ for signed graphs. This signed Laplacian does not only exist for the combinatorial Laplacian $\mathbf{L}$ but also for the normalized Laplacian matrix $\mathbf{Z} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

We used the signed Laplacian to derive a new drawing algorithm for signed graphs, in which negative edges are interpreted to denote *antipodal proximity*, i.e. the proximity of one node to the opposite position of the other node. We also showed that spectral clustering of signed graphs is possible using the intuitive measure of cut which counts positive edges between two clusters and negative edges inside each cluster. As for unsigned spectral clustering, the different Laplacian matrices correspond to ratio cuts and normalized cuts. For link prediction, we saw that the signed Laplacians can be used as kernels (since they are positive-semidefinite), and can replace graph kernels based on the adjacency matrix. This is especially true if the sign of edges is to be predicted. Finally, we have derived that in the signed Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$, $\mathbf{D}$ must be defined using the sum of absolute edge weights. We showed that this definition of $\mathbf{D}$ and $\mathbf{L}$ is natural since it arises in signed graph drawing, signed spectral clustering, the signed resistance distance and Markov random fields used for link sign prediction. These derivations should confirm that this definition of $\mathbf{D}$ and $\mathbf{L}$ is to be preferred over one that omits the absolute value.

In all cases, we observed that if a graph is not balanced, zero is not an eigenvalue of its Laplacian matrix and thus its eigenvectors can be used directly unlike the unsigned case if the eigenvector of least eigenvalue is trivial and can be ignored. For graph drawing, this results in the loss of translational invariance of the drawing, i.e. the drawing is placed relative to a point zero.

All these various applications of algebraic graph theory to signed graph are a justification of the multiplication rule *The enemy of my enemy is my friend*, since they are all based on it. We conclude that this rule is valid in general, even though it is not universal, as shown e.g. in [LHK10b].

Chapter 6

Bipartite and Directed Networks

The previous chapters described networks that are unipartite and undirected. In this chapter, the spectral evolution model is applied to the special cases of bipartite and directed networks. These two cases lead to asymmetry in the data, which can be handled by replacing the eigenvalue decomposition with the singular value decomposition. As we will show, the spectral evolution model can then be used analogously.

While at first bipartite networks and directed networks seem to be two completely different classes of networks, there is a relation between them: Both types of network are *asymmetric*. In directed networks, the links themselves are asymmetric, resulting in directed arcs instead of edges. In bipartite networks, the global structure of the network is asymmetric, containing two node types. Both types of asymmetry are related: For instance, Twitter was described as news media rather than a social network due to the strong asymmetry of the *follow* relation [KLPM10], which would indicate that users of Twitter can be partitioned into news writing users and news reading users. In other words, the directed network of *follow* relations is nearly bipartite. On the other hand, every bipartite network can be made directed simply by orienting each edge from the first vertex partition to the second.

Table 6.1 gives an overview of the network types considered in this thesis. The network types described in this chapter are shown in the middle and right columns, whereas the first column covers the networks used in the previous chapters. Note that this classification is independent of edge weights, and therefore we also cover signed bipartite and signed directed networks in this chapter.

Table 6.1: Network types considered in this work by structure of the edges, independently of edge weights. This chapter describes the columns labeled **Directed** and **Bipartite**.

	Undirected	**Directed**	**Bipartite**
Graph type	Unipartite		Bipartite
	Undirected	Directed	Undirected
Matrix type	Square		Rectangular
	Symmetric	Asymmetric	

Our contributions in this chapter are the definition of the hyperbolic sine and odd Neumann pseudokernels. The chapter is structured by the type of networks considered: We begin by extending the spectral evolution model to bipartite networks (Section 6.1), then to signed bipartite networks in which the link prediction problem corresponds to the problem of collaborative filtering (Section 6.2) and finally to directed networks (Section 6.3). Section 6.4 introduces a spectral method for detecting networks that are nearly bipartite.

6.1 Bipartite Networks

Many networks contain edges between two types of entities, for instance item rating graphs, authorship graphs and document–feature networks. These graphs are called bipartite [HLEK03, SLZZ10]. An example is drawn in Figure 6.1: the bipartite network of musical artists and their genres from DBpedia (GE).

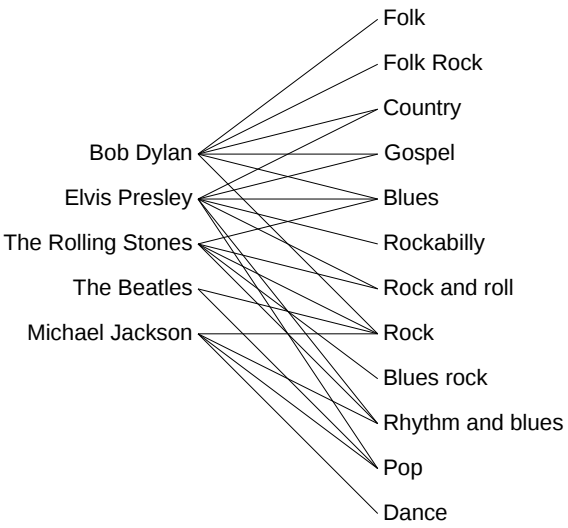


Figure 6.1: A small extract of the DBpedia entity–genre dataset (GE). This network is unweighted, unsigned, undirected and bipartite. It contains musical artists and genres, and each edge connects one artist with one genre. The data was retrieved from the DBpedia project [ABK+08], which in turn extracted it from articles in the English Wikipedia. In drawings of bipartite networks such as this one, each group of nodes is typically placed on one side of the plot.

Although bipartite graphs are a special case of general graphs, link prediction methods cannot be applied to them. As we show in Section 6.1.1, this is the case for all link prediction functions based on the triangle closing model, as well as all positive-semidefinite graph kernels. Instead, we will see that odd spectral transformations must be used on them in Section 6.1.2. For each spectral transformation defined in Section 3.3, we derive the corresponding odd spectral transformation. One example is the exponential graph kernel $\exp(\alpha\mathbf{A})$. Its odd component is $\sinh(\alpha\mathbf{A})$, the hyperbolic sine. We also introduce the bipartite Neumann pseudokernel, and study the bipartite versions of power series with only odd powers. We show experimentally (in Section 6.1.4) how these odd pseudokernels perform on the task of link prediction in bipartite networks in comparison to their positive counterparts, and give an overview of their relative performances.

Given an undirected graph $G = (V, E)$ with vertex set V and edge set E, its adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$ is defined as $\mathbf{A}_{ij} = 1$ if $(i, j) \in E$ and $\mathbf{A}_{ij} = 0$ otherwise. For a bipartite graph $G = (V_1 \dot\cup V_2, E)$, the adjacency matrix can be written as $\mathrm{A} = \left[\mathbf{0}\,\mathbf{B}; \mathbf{B}^{\mathrm{T}}\,\mathbf{0}\right]$, where $\mathbf{B} \in \mathbb{R}^{|V_1|\times|V_2|}$ is the biadjacency matrix of G.

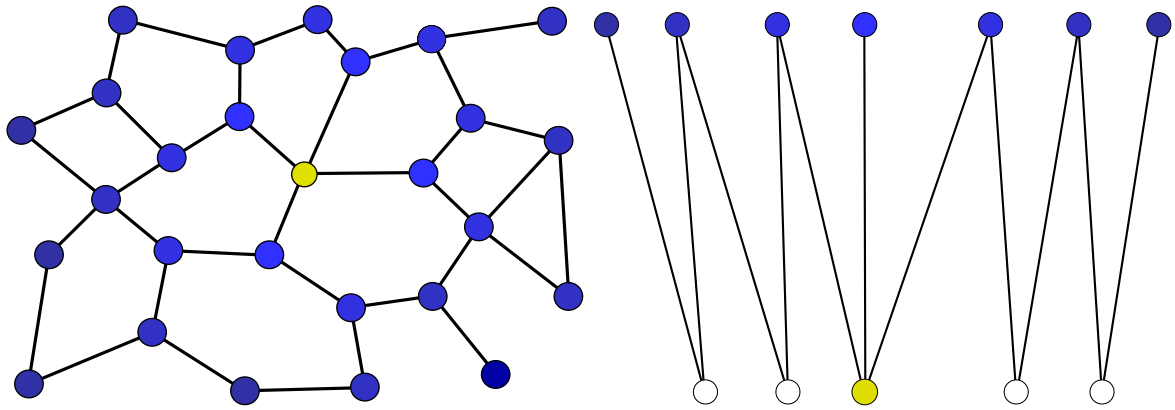
6.1.1 Bipartite Link Prediction

The link prediction problem is usually defined on unipartite graphs, where common link prediction algorithms make several assumptions [LBKT08]:

- Triangle closing: New edges tend to form triangles.
- Clustering: Nodes tend to form well-connected clusters in the graph.

In bipartite graphs these assumptions are not true, since triangles and larger cliques cannot appear. Other assumptions have therefore to be used. While a unipartite link prediction algorithm technically applies to bipartite graphs, it will not perform well. Methods based on common neighbors of two vertices will for instance not be able to predict anything in bipartite graphs, since two vertices that would be connected (from different clusters) do not have any common neighbors. Note however that preferential attachment as defined in Equation 4.2 can be used as a link prediction function even in bipartite graphs.

Several important classes of networks are bipartite: authorship networks, group membership networks, rating networks, features such as country of origin and genre, and many more. Many unipartite networks (such as coauthorship networks) can be reinterpreted as bipartite networks when edges or cliques are modeled as vertices. In these cases, special bipartite link prediction algorithms are necessary.



(a) Unipartite network (b) Bipartite network

Figure 6.2: A sketch of link prediction methods in unipartite and bipartite networks. The brightness of blue nodes indicates a possible link prediction score between that node and the yellow node. In the unipartite case, all paths are used. In the bipartite case, only paths of odd length need to be considered. In both cases, the weight of paths is weighted in inverse proportion to path length.

6.1.2 Odd Spectral Transformations

In Section 3.3, we saw that many link prediction functions are power series of the adjacency matrix $\mathbf{A}$ and can be expressed as a sum over all paths between two nodes, weighted by a function of the path length. As a result, these link prediction functions take into account paths of all lengths. In the case of bipartite networks however, a new edge can only appear between two vertices of different partitions, which themselves can only be connected by paths of odd lengths because a path of even length can only connect vertices of the same partition. Therefore, only odd powers of $\mathbf{A}$ are relevant, and we can restrict the link prediction functions to odd power series of $\mathbf{A}$, i.e. power series with only odd powers.

The resulting spectral transformation is then an odd function and except in the trivial and undesired case of a constant zero function, will be negative at some point. Therefore, such a spectral transformation does not result in positive-semidefinite matrices and therefore cannot

be used to define graph kernels. Instead, we will use pseudokernels, which we define to be equivalent to kernels without the positive-semidefiniteness.

Odd Path Counting When using power series of the adjacency matrices $\mathbf{A}$ or $\mathbf{N}$ to predict links in bipartite networks, we are in fact interested only in paths of odd length and therefore we will use power series with only odd coefficients. Such power series are always odd functions:

$$p_{\mathrm{O}}(\mathbf{A}) = \alpha\mathbf{A} + \beta\mathbf{A}^3 + \gamma\mathbf{A}^5 + \dots \tag{6.1}$$

$$p_{\mathrm{O}}(\mathbf{N}) = \alpha\mathbf{N} + \beta\mathbf{N}^3 + \gamma\mathbf{N}^5 + \dots \tag{6.2}$$

Hyperbolic Sine In unipartite networks, a basic link prediction function is given by the matrix exponential of the adjacency matrix [ISKM05, WC04, KSTC02]. The matrix exponential can be derived by considering the sum

$$\exp(\alpha\mathbf{A}) = \sum_{k=0}^{\infty} \frac{\alpha^k}{k!}\mathbf{A}^k = \mathbf{I} + \alpha\mathbf{A} + \frac{1}{2}\alpha^2\mathbf{A}^2 + \frac{1}{6}\alpha^3\mathbf{A}^3 + \dots$$

where coefficients are decreasing with path length. Keeping only the terms containing odd powers, we arrive at the matrix hyperbolic sine [HJ94, Chapter 6].

$$K_{\mathrm{SINH}}(\mathbf{A}) = \sinh(\alpha\mathbf{A}) = \sum_{k=0}^{\infty} \frac{\alpha^{1+2k}}{(1+2k)!}\mathbf{A}^{1+2k} = \alpha\mathbf{A} + \frac{1}{6}\alpha^3\mathbf{A}^3 + \frac{1}{120}\alpha^5\mathbf{A}^5 + \dots \tag{6.3}$$

Figure 6.3 shows the hyperbolic sine applied to the (positive) spectrum of the bipartite English Wikipedia user–article edit network.

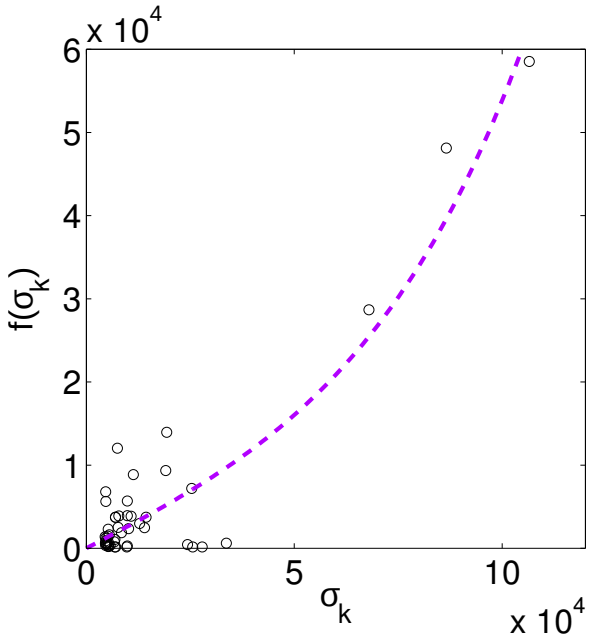


Figure 6.3: The curve fitting plot of the bipartite English Wikipedia edit network (en), using the method described in Section 4.2.2. In this curve fitting plot, the hyperbolic sine is a good match, indicating that the hyperbolic sine pseudokernel performs well.

Odd Neumann Pseudokernel The Neumann kernel for unipartite graphs is given by the following expression [ISKM05]:

$$K_{\text{NEU}}(\mathbf{A}) = (\mathbf{I} - \alpha\mathbf{A})^{-1} = \sum_{k=0}^{\infty} \alpha^k \mathbf{A}^k = \mathbf{I} + \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + \alpha^3\mathbf{A}^3 + \dots$$

Keeping only the terms containing odd powers of $\mathbf{A}$, we arrive at the odd Neumann pseudokernel:

$$K_{\text{NEU-O}}(\mathbf{A}) = \alpha\mathbf{A}(\mathbf{I} - \alpha^2\mathbf{A}^2)^{-1} = \sum_{k=0}^{\infty} \alpha^{1+2k}\mathbf{A}^{1+2k} = \alpha\mathbf{A} + \alpha^3\mathbf{A}^3 + \alpha^5\mathbf{A}^5 + \dots \tag{6.4}$$

The hyperbolic sine and Neumann pseudokernels are compared in Figure 6.4, based on the path weights they produce. Since these two kernels contain only the odd powers of the corresponding non-odd graph kernels, they can be expressed in the following way:

$$K_{\text{NEU-O}}(\mathbf{A}) = \frac{1}{2}\left(K_{\text{NEU}}(\mathbf{A}) - K_{\text{NEU}}(-\mathbf{A})\right) \tag{6.5}$$

$$K_{\text{SINH}}(\mathbf{A}) = \frac{1}{2}\left(K_{\text{EXP}}(\mathbf{A}) - K_{\text{EXP}}(-\mathbf{A})\right) \tag{6.6}$$

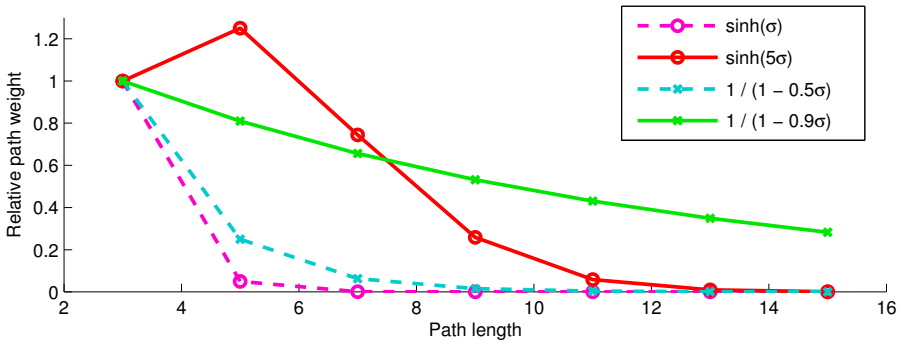


Figure 6.4: Comparison of the hyperbolic sine and the odd Neumann pseudokernels. For each possible path length, the graph shows the relative weight for paths of this lengths for the exponential and Neumann kernels using two parametrizations. These weights correspond to the factors in the Taylor series expansions of these functions.

Odd Normalized Pseudokernels The normalized adjacency matrix $\mathbf{N}$ can be used for link prediction, too. The odd normalized exponential and Neumann pseudokernels are defined analogously to the unnormalized ones:

$$K_{\text{SINH}}(\mathbf{N}) = \sinh(\alpha\mathbf{N}) = \sum_{k=0}^{\infty} \frac{\alpha^{1+2k}}{(1+2k)!}\mathbf{N}^{1+2k} \tag{6.7}$$

$$K_{\text{NEU-O}}(\mathbf{N}) = \alpha\mathbf{N}(\mathbf{I} - \alpha^2\mathbf{N}^2)^{-1} = \sum_{k=0}^{\infty} \alpha^{1+2k}\mathbf{N}^{1+2k} \tag{6.8}$$

The normalized odd commute-time pseudokernel is defined as follows:

$$K_{\text{COM-O}}(\mathbf{N}) = \mathbf{N}(\mathbf{I} - \mathbf{N}^2)^{+} \tag{6.9}$$

Again, these odd pseudokernels can be derived from the corresponding kernels by the expression

$$K_{\text{X-O}}(\mathbf{N}) = \frac{1}{2}\left(K_{\text{X}}(\mathbf{N}) - K_{\text{X}}(-\mathbf{N})\right)$$

for $\text{X} \in \{\text{EXP}, \text{NEU}, \text{COM}\}$, by considering EXP-O = SINH. The kernels based on the Laplacian $\mathbf{L}$ cannot be extended to odd pseudokernels because the Moore–Penrose pseudoinverse is already an odd spectral transformation.

6.1.3 Decomposition of the Biadjacency Matrix

Bipartite graphs have adjacency matrices of the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^{\text{T}} & \mathbf{0} \end{bmatrix}, \tag{6.10}$$

where $\mathbf{B}$ is called the biadjacency matrix of the graph. This form can be exploited to reduce the eigenvalue decomposition of $\mathbf{A}$ to the equivalent singular value decomposition of $\mathbf{B}$ [MF10]. Given the singular value decomposition $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\text{T}}$, the eigenvalue decomposition of $\mathbf{A}$ is given by

$$\mathbf{A} = \begin{bmatrix} \bar{\mathbf{U}} & \bar{\mathbf{U}} \\ \bar{\mathbf{V}} & -\bar{\mathbf{V}} \end{bmatrix} \begin{bmatrix} +\mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0} & -\mathbf{\Sigma} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{U}} & \bar{\mathbf{U}} \\ \bar{\mathbf{V}} & -\bar{\mathbf{V}} \end{bmatrix}^{\text{T}} \tag{6.11}$$

with $\bar{\mathbf{U}} = \mathbf{U}/\sqrt{2}$ and $\bar{\mathbf{V}} = \mathbf{V}/\sqrt{2}$. In this decomposition, each singular value σ corresponds to the eigenvalue pair $\{\pm\sigma\}$. Odd powers of $\mathbf{A}$ then have the form

$$\mathbf{A}^{2k+1} = \begin{bmatrix} \mathbf{0} & (\mathbf{B}\mathbf{B}^{\text{T}})^k\mathbf{B} \\ (\mathbf{B}^{\text{T}}\mathbf{B})^k\mathbf{B}^{\text{T}} & \mathbf{0} \end{bmatrix},$$

where the alternating power $(\mathbf{B}\mathbf{B}^{\text{T}})^k\mathbf{B}$ can be explained by the fact that in the bipartite network, a path will follow edges from one vertex set to the other in alternating directions, corresponding to the alternating transpositions of $\mathbf{B}$.

The same is true in the normalized case: The eigenvalue decomposition of the normalized adjacency matrix $\mathbf{N}$ can be computed using the singular value decomposition of the normalized biadjacency matrix $\mathbf{M}$. Note that there is no corresponding expression relating the decomposition of the bipartite Laplacian matrix $[\mathbf{D}_1 \ -\mathbf{B}; -\mathbf{B}^{\text{T}} \ \mathbf{D}_2]$ and the decomposition of the biadjacency matrix $\mathbf{B}$.

6.1.4 Experiments

As experiments, we run all link prediction algorithms on all unsigned bipartite networks, as given in Table A.1 in Appendix A. The methodology used in the bipartite link prediction experiments is the same as the one described for unipartite networks in Section 4.4. We use all the unipartite link prediction methods from Table 4.4, and the additional bipartite link prediction methods from Table 6.2. The evaluation results are summarized in Figure 6.5, and full evaluation results are given in Table B.3 in Appendix B. The best performing spectral transformation for a sample of signed network datasets is given in Table 6.3.

The two learning methods from Sections 4.2 and 4.3 can be applied to bipartite networks using the singular value decomposition of the biadjacency matrix instead of the eigenvalue decomposition of the adjacency matrix. We show examples of this techniques in Figures 6.6 and 6.7. The main difference to the unipartite case is that all singular values are positive.

Table 6.2: The complete list of bipartite link prediction methods used in the evaluation. The methods given here complete those given in Table 4.4.

	Method	Definition	
Curve fitting with A			
A-POLY-O	Odd path counting	$\sum_{k=1}^{4} \alpha_k \mathbf{A}^{2k-1}$	Eq. 6.1
A-POLYN-O	Odd nonnegative path counting	$\sum_{k=1}^{7} \alpha_k \mathbf{A}^{2k-1}, \alpha_k \geq 0$	Eq. 6.1
A-SINH	Hyperbolic sine pseudokernel	$\beta \sinh(\alpha \mathbf{A})$	Eq. 6.3
A-NEU-O	Odd Neumann pseudokernel	$\beta\alpha\mathbf{A}(\mathbf{I} - \alpha^2\mathbf{A}^2)^{-1}$	Eq. 6.4
Curve fitting with N			
N-POLY-O	Odd normalized path counting	$\sum_{k=0}^{4} \alpha_k \mathbf{N}^{2k-1}$	Eq. 6.2
N-POLYN-O	Odd normalized nonnegative path counting	$\sum_{k=0}^{7} \alpha_k \mathbf{N}^{2k-1}, \alpha_k \geq 0$	Eq. 6.2
N-SINH	Normalized hyperbolic sine pseudokernel	$\beta \sinh(\alpha \mathbf{N})$	Eq. 6.7
N-NEU-O	Odd normalized Neumann pseudokernel	$\beta\alpha\mathbf{N}(\mathbf{I} - \alpha^2\mathbf{N}^2)^{-1}$	Eq. 6.8
N-COM-O	Odd normalized commute-time pseudokernel	$\beta\mathbf{N}(\mathbf{I} - \mathbf{N}^2)^{+}$	Eq. 6.9

Table 6.3: The best performing spectral transformation for a sample of unsigned bipartite datasets. For each dataset, we show the source and target matrices, the curve fitting model and the link prediction method that performs best.

Dataset	Best transformation	Best method	MAP
German Wikipedia (de)	$\mathbf{B}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	X-ABS	0.718
French Wikipedia (fr)	$\mathbf{B}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	A-SINH	0.752
Github (GH)	$\mathbf{M}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	N-COM-O	0.688
Movies (ST)	$\mathbf{M}_\mathrm{a} \to \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	A-SINH	0.682
Filmtipset (Fc)	$\mathbf{B}_\mathrm{a} \to \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	A-POLY-O	0.681
English Wikipedia (WC)	$\mathbf{M}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	N-NEU-O	0.688
BibSonomy (B_{ti})	$\mathbf{B}_\mathrm{a} \to \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	A-POLY-O	0.962
CiteULike (C_{ui})	$\mathbf{L}_\mathrm{a} \to \mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{b}$	L-HEAT	0.873
MovieLens (M_{ti})	$\mathbf{M}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	N-COM-O	0.874
Twitter (W_{ui})	$\mathbf{B}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	A-POLYN-O	0.754
vi.sualize.us (V_{ut})	$\mathbf{M}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	N-COM-O	0.876
Last.fm (Ls)	$\mathbf{B}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	X-ABS	0.851

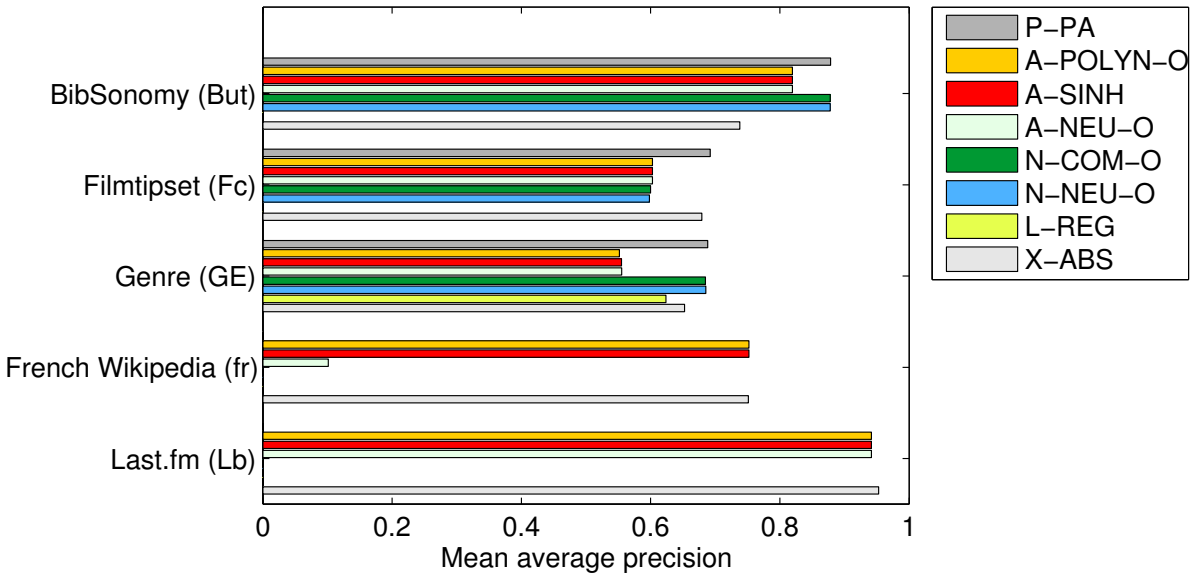


Figure 6.5: Extract of experiment results for bipartite networks. See the text for a description of the datasets and link prediction methods. The full evaluation results are given in Appendix B.

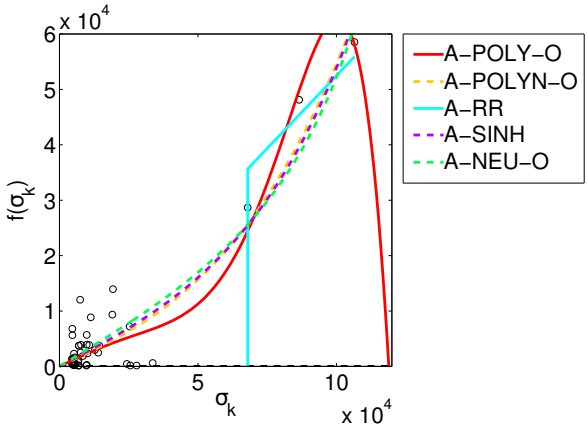


Figure 6.6: Learning an odd spectral transformation that matches an observed spectral transformation in the English Wikipedia edit network (en).

Observations We observe that in most cases, using the unnormalized biadjacency matrix gives better results than using the normalized biadjacency matrix. This is in contrast to the unipartite case, where both variants appear as the best method for specific datasets. The choice of the best graph kernel or spectral transformation function however is different for every dataset.

6.1.5 The Split-complex Numbers

In this section, we present an alternative way of modeling bipartite graphs, using the split-complex numbers. The split-complex numbers are an extension of the real numbers similar to the complex numbers [Ros97]. Instead of including an imaginary number i such that $i^2 = -1$, the split-complex numbers include an imaginary number $\jmath$ such that $\jmath^2 = +1$. In contrast to the complex numbers, the split-complex numbers are not a field, since they include nonzero

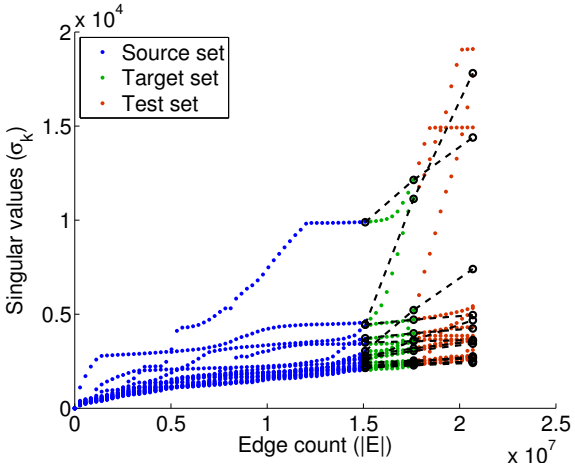


Figure 6.7: The spectral extrapolation method applied to the bipartite Spanish Wikipedia edit network (es). Instead of using eigenvalues, the method uses singular values for bipartite graphs.

elements that are not invertible. However, the split-complex numbers can be used to model bipartite relationships.

Imagine a dating site such as Líbímseti.cz [BP07] where persons can rate each other's profile. We assume, for the sake of the argument, that men will only be interested in women and women only in men. A rating can therefore only exist between a man and a woman and therefore, the rating network is bipartite. Also, we will assume that ratings are always reciprocal, i.e. that rating edges are undirected. However, we may still be interested in links between persons of the same gender as a measure of similarity. For instance, two women are similar if they have rated the same men similarly. Therefore, we really need two kinds of relationships in this network: *like* and *is-similar*. Let us represent these two possible values by e_{like} and $e_{\text{is-similar}}$. Now, remember that in the triangle closing model of Section 3.3.1, we multiply the weights of two adjacent edges to generate a new edge. In the case of a dating site, we can formulate the following natural triangle closing rules:

- Two persons that like the same person are similar.
- Two persons that are similar to the same person are similar.
- A person similar to a person that likes a third person will like that third person.

These rules can be expressed mathematically in the following way:

$$\begin{aligned} e_{\text{like}} \cdot e_{\text{like}} &= e_{\text{is-similar}} \\ e_{\text{is-similar}} \cdot e_{\text{is-similar}} &= e_{\text{is-similar}} \\ e_{\text{like}} \cdot e_{\text{is-similar}} &= e_{\text{like}} \end{aligned}$$

We thus have to find values of e_{like} and $e_{\text{is-similar}}$ that solve these equations, and in which both constants are nonzero. A trivial solution is given by $e_{\text{like}} = e_{\text{is-similar}} = 1$. However, this trivial solution is not satisfactory, since we want the relationships *like* and *is-similar* to be different. From the second and third equations, we can derive that $e_{\text{is-similar}} = 1$. Therefore, e_{like} is a number different from zero and from 1 that squares to 1. Since no real number has these properties, we have to use a non-real value for e_{like} such that $e_{\text{like}}^2 = 1$. This construction

corresponds to the split-complex numbers, where the imaginary unit $\jmath$ squares to one:

$$e_{\text{like}} = \jmath$$
$$e_{\text{is-similar}} = 1$$

Our three requirements then correspond to the identities $\jmath \cdot \jmath = 1$, $1 \cdot 1 = 1$ and $\jmath \cdot 1 = \jmath$ that hold in the split-complex numbers. Thus, the split-complex numbers can be used to model bipartite relationships.

The split-complex numbers were introduced by Clifford in 1873 [Cli73]. They can be defined formally as the set $\mathbb{C}_{\text{s}} = \{a + b\jmath \mid a, b \in \mathbb{R}\}$. Note that there is no established notation for the set of split-complex numbers. In this text we will use $\mathbb{C}_{\text{s}}$. The defining identity of split-complex numbers is $\jmath^2 = +1$. From this, other results can be derived. Unlike the complex numbers, $\mathbb{C}_{\text{s}}$ is not a field. Instead, $\mathbb{C}_{\text{s}}$ is a commutative ring, i.e. all field axioms are valid except for the existence of the multiplicative inverse, which does not exist for numbers of the form $a \pm a\jmath$. As a result, products of two nonzero numbers can be zero, e.g. $(1+\jmath)(1-\jmath) = 0$. Due to these defects, the split-complex are used much less than complex numbers. Split-complex numbers are also called hyperbolic complex numbers because they can represent hyperbolic angles [Huc93]. In the context of special relativity for instance, numbers of the form $a + b\jmath$ with $a^2 - b^2 = 1$ are used to model Lorentz boosts. Other applications of hyperbolic angles are squeeze mappings in geometry, giving them the alternative name *hyperbolic numbers* [Sob95].

Let $G = (V, E)$ be a unipartite, unweighted and undirected network. Instead of giving an explicit partition of the vertices V into $V = V_1 \dot\cup V_2$, we will model bipartite edges explicitly as having weight $\jmath$. Thus, the split-complex adjacency matrix of G is $\mathbf{A}_{\text{s}} \in \mathbb{C}_{\text{s}}^{|V|\times|V|}$, with $(\mathbf{A}_{\text{s}})_{ij} = \jmath$ if $\{i, j\} \in E$ and $(\mathbf{A}_{\text{s}})_{ij} = 0$ otherwise. Due to the multiplication rules of split-complex numbers, a power $\mathbf{A}_{\text{s}}^k$ contains, for each pair (i, j), the number of paths between i and j, separated into paths with an even number of *like* edges in the real part and paths with an odd number of *like* edges in the imaginary part. This is due to the fact that $\jmath^k = 1$ when k is even and $\jmath^k = \jmath$ when k is odd. Equivalently, a split-complex number $a + b\jmath$ can be represented by the 2×2 matrix

$$a + b\jmath = \begin{bmatrix} a & b \\ b & a \end{bmatrix}.$$

In this representation, the addition and multiplication of split-complex numbers corresponds to the addition and multiplication of 2×2 matrices. The units 1 and $\jmath$ then correspond to

$$1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$
$$\jmath = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Using this representation, the split-complex adjacency matrix $\mathbf{A}_{\text{s}} \in \mathbb{C}_{\text{s}}^{|V|\times|V|}$ can be reordered to give the matrix

$$\mathbf{A}_{\text{s}} = \begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^{\text{T}} & \mathbf{0} \end{bmatrix}$$

in which $\mathbf{A}$ is the ordinary adjacency matrix of the network. This is equivalent to considering $\mathbf{A}_{\text{s}}$ as the biadjacency matrix of the bipartite double cover of the original graph, as explained in Section 6.3.1.

6.2 Collaborative Filtering

Some networks are signed and bipartite at the same time. This combination appears in collaborative filtering, where in the rating graph, all edges connect users with items, and denote ratings which can be positive (*like*) and negative (*dislike*).

All eight signed bipartite networks in our collection are rating networks. Note that a single rating network, Líbímseti.cz (LI), is unipartite since it contains ratings of users by users. It can therefore not be used for collaborative filtering. Our collaborative filtering networks contain ratings given by users to movies (MovieLens (M1, M2, M3), Netflix (NX), Filmtipset (Fr)), books (BookCrossing (BX)), products (Epinions (ER)) and jokes (Jester (JE)). A summary of all collaborative filtering datasets is given in Table 6.4.

Table 6.4: The list of collaborative filtering datasets used in this thesis. All datasets are bipartite and have weighted edges connecting users with various kinds of items.

Code	Dataset	Items	Rating scale	Users $\lvert V_1\rvert$	Items $\lvert V_2\rvert$	Ratings $\lvert E\rvert$
BX	BookCrossing	Books	$\{1,\ldots,10\}$	105,283	340,532	1,149,742
ER	Epinions	Products	$\{1,2,3,4,5\}$	120,492	755,760	13,668,320
Fr	Filmtipset	Movies	$\{1,2,3,4,5\}$	80,482	64,189	19,554,219
JE	Jester	Jokes	$[-1,+1]$	24,938	100	616,912
M1	MovieLens 100k	Movies	$\{1,2,3,4,5\}$	943	1,682	100,000
M2	MovieLens 1M	Movies	$\{1,2,3,4,5\}$	6,040	3,706	1,000,209
M3	MovieLens 10M	Movies	$\{0.5,1.0,\ldots,5.0\}$	71,567	65,133	10,000,054
NX	Netflix	Movies	$\{1,2,3,4,5\}$	480,189	17,770	100,480,507

Let $G = (V_1 \dot\cup V_2, E, w)$ be the weighted bipartite rating graph of a collaborative filtering dataset, in which V_1 is the set if users and V_2 is the set of items. As described in Section 4.1.2, we normalize the data additively in the following way. Let μ be the overall mean rating given by

$$\mu = \frac{1}{|E|} \sum_{\{i,j\}\in E} w(i,j).$$

Then the nonzero entries of the biadjacency matrix $\mathbf{B}$ are given by

$$\mathbf{B}_{ij} = w(i,j) - \mu$$

for every $\{i,j\} \in E$. Note that $\mathbf{B}$ corresponds to the rating matrix.

For link prediction in signed bipartite networks, we can use all the bipartite link prediction methods introduced in the previous section. Just as in the unipartite case, link prediction methods for unsigned networks generalize to signed networks by allowing negative values in the biadjacency matrix $\mathbf{B}$ and defining the two degree matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ using the sum of absolute edge weights:

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 \end{bmatrix} \tag{6.12}$$

$$(\mathbf{D}_1)_{ii} = \sum_j |\mathbf{B}_{ij}| \tag{6.13}$$

$$(\mathbf{D}_2)_{ii} = \sum_j |\mathbf{B}_{ji}| \tag{6.14}$$

6.2.1 Experiments

We use the experimental methodology of link sign prediction described in Section 4.4. In collaborative filtering, the corresponding problem is usually called *rating prediction*. We show the experimental results in Table 6.5 and in Figure 6.8. Due to the small number of signed bipartite networks available, we cannot make general statements about the accuracy of different methods, although the odd normalized commute-time kernel (N-COM-O) stands out for three datasets (M1, M2 and NX), and that the extrapolation methods (X-ABS and X-SQU) seem to work better for collaborative filtering than for other types of datasets.

Table 6.5: The best performing collaborative filtering algorithms for the signed bipartite datasets. For each dataset, we show the source and target matrices, the curve fitting model and the link prediction method that performs best.

Dataset	Best transformation	Best method	MAP
BookCrossing (BX)	$\mathbf{M}_\mathrm{a} \to \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	N-POLYN-O	0.278
Epinions (ER)	$\mathbf{B}_\mathrm{a} \to \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	A-POLYN-O	0.461
Filmtipset (Fr)	$\mathbf{B}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	X-SQU	0.655
Jester (JE)	$\mathbf{M}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	X-SQU	0.590
MovieLens 10M (M3)	$\mathbf{B}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	X-SQU	0.614
Netflix (NX)	$\mathbf{B}_\mathrm{a} \to \mathbf{B}_\mathrm{a} + \mathbf{B}_\mathrm{a}^\mathrm{T} + \mathbf{B}_\mathrm{b} + \mathbf{B}_\mathrm{b}^\mathrm{T}$	X-ABS	0.568

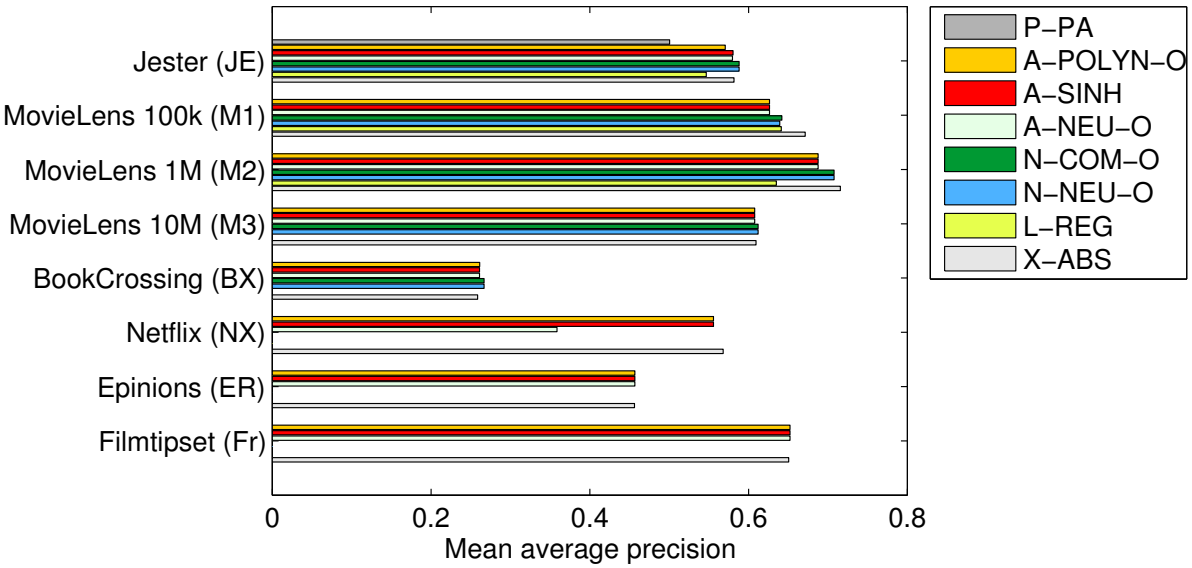


Figure 6.8: The mean average precision for a selection of link sign prediction methods applied to signed bipartite networks.

6.2.2 Bipartite Algebraic Conflict

The algebraic conflict ξ defined in Section 5.4.4 can be applied to signed bipartite networks to denote the amount of conflict present in the network. The algebraic conflict ξ is defined as the smallest eigenvalue λ_1 of the Laplacian matrix $\mathbf{L}$ of a signed network. As we showed for unipartite networks, $\mathbf{L}$ is positive-semidefinite and its smallest eigenvalue is zero exactly

when the graph is balanced, i.e. when all its cycles contain an even number of negative edges. Table 6.6 shows the algebraic conflict ξ for the signed bipartite networks in our collection.

Table 6.6: The algebraic conflict ξ for several large signed networks. Smaller values indicate a more balanced network.

Network	ξ	
Jester (JE)	0.6725	Conflict
MovieLens 10M (M3)	0.02302	↕
MovieLens 100k (M1)	0.005807	↕
MovieLens 1M (M2)	0.005564	Balance

6.3 Directed Networks

In the previous sections, we assumed that relations between entities were symmetric. In practice, many relations are asymmetric: For instance, links between webpages are generally unidirectional. The fact that webpage i links to webpage j does not imply that webpage j links to webpage i. An example of such an asymmetric hyperlink network is shown in Figure 6.9.

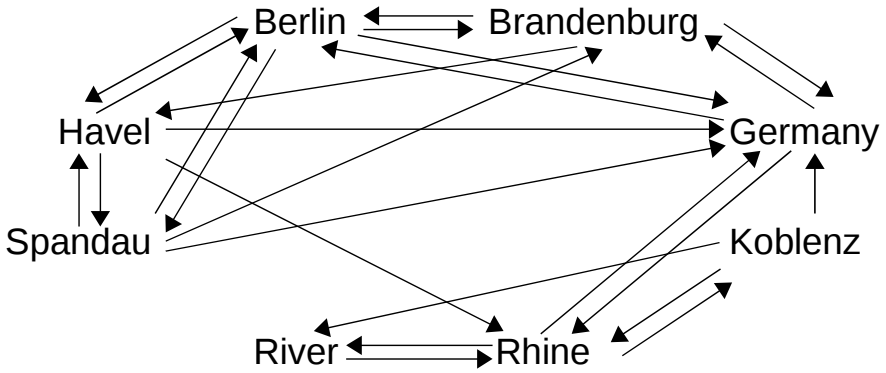


Figure 6.9: A small extract of the hyperlink network of the English Wikipedia (DL). This network is directed, unweighted and unsigned. It contains articles of the English Wikipedia, and each arc represents one hyperlink between two articles. The data was retrieved from the DBpedia project [ABK+08], which in turn extracted it from articles in the English Wikipedia.

The corresponding graph is directed, and its edges are called *arcs*. Directed graphs (or *digraphs*) are denoted $D = (V, A)$, where V is the set of edges and A is the set of arcs. An arc $a \in A$ is denoted as $a = (i, j)$ if it goes from vertex i to vertex j. We allow self-loops, i.e. arcs of the form $a = (i, i)$. Self-loops may arise for instance in email networks, when a users sends an email to himself.

Like an undirected graph, a directed graph has an adjacency matrix. Unlike undirected graphs, the adjacency matrix of a directed graph is asymmetric in the general case. When building the adjacency matrix $\mathbf{A}$ of a directed graph, we set $\mathbf{A}_{ij} = 1$ if there is a directed edge from vertex i to vertex j, and $\mathbf{A}_{ij} = 0$ otherwise. The matrix $\mathbf{A}$ is thus square and, in general, asymmetric.

In a directed graph $D = (V, A)$ with multiple arcs, the set of arcs A is a multiset, and $m(i, j)$ denotes the number of arcs between the vertices i and j. The adjacency matrix of a directed

graph with multiple arcs is defined as $\mathbf{A}_{ij} = m(i,j)$. Directed graphs may also be weighted. A weighted directed graph is denoted by $D = (V, A, w)$, where $w(i,j)$ is a function giving the weight of the arc (i,j). The adjacency matrix of a weighted directed graph is then defined as $\mathbf{A}_{ij} = w(i,j)$ when $(i,j) \in A$ and $\mathbf{A}_{ij} = 0$ otherwise.

Not a single network in the network collection is bipartite and directed at the same time. Therefore, we can assume that all directed graphs are unipartite.

6.3.1 Spectral Analysis

Since the matrix $\mathbf{A}$ is asymmetric, its eigenvalue decomposition is generally undefined. Instead, we can do one of the following:

- (sym) Ignore arc orientations and apply the eigenvalue decomposition to $\mathbf{A} + \mathbf{A}^{\mathrm{T}}$
- (asym) Apply the singular value decomposition to $\mathbf{A}$
- (back) Apply the singular value decomposition to $\mathbf{A} + \alpha\mathbf{A}^{\mathrm{T}}$, where $0 < \alpha < 1$

The first case (sym) is trivial as it views the network as undirected, and therefore reduces to the symmetric case which was described in Chapters 2 to 4.

The second solution (asym) is more interesting. It is equivalent to considering a bipartite network, where each original node i has two equivalent nodes i^{in} and i^{out}, and each original arc (i,j) is mapped to a bipartite edge $\{i^{\mathrm{out}}, j^{\mathrm{in}}\}$. The resulting graph is also known as the bipartite double cover of G [DM58]. This makes it impossible to follow chains of incident directed arcs, and only allows following alternating paths, i.e. paths of arcs in alternating directions. It is useful if the network is almost bipartite. The second solution also arises by considering a directed clustering problem as shown in [MP07].

The third solution (back) allows us to retain the orientation of arcs and to follow chains of incident arcs at the same time. The method is used in [GKRT04] where it is called *transpose trust* and is applied to trust prediction in directed trust networks.

The spectral evolution model applies to all three methods. The first case reduces to computing the eigenvalue decomposition as described in Chapters 2 to 4. The second and third reduce to computing the singular value decomposition as described in Section 6.1.

Directed Laplacian There is no known way of defining an asymmetric Laplacian matrix. However, certain definitions of the Laplacian matrix for directed graphs exist, all being symmetric [Chu05, Maj06, MYC+10]. These Laplacian matrices are not studied here.

6.3.2 Experiments

We use the experimental setup described in Section 4.4 and apply it to the directed unipartite networks in our collection. The best performing spectral transformations for a selection of directed datasets is given in Table 6.7. Figure 6.10 compares the performance of the three decomposition types for directed networks. For the third method (back), we use the value $\alpha = 0.2$. The full evaluation results are given in Table B.2 in Appendix B.

Observations While the spectral evolution model can be observed in all types of networks, not all types of networks follow it equally well. By comparing the experimental results in this chapter with the experimental results in Chapter 4, we observe two effects. First, bipartite networks seem to follow the spectral evolution model better than unipartite networks. Second,

Table 6.7: The best performing spectral transformations for a selection of directed datasets. For each dataset, we show the source and target matrices, the curve fitting model and the link prediction method that performs best.

Dataset	Best transformation	Best method	MAP
Slashdot Zoo (SZ)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-POLYN	0.397
English Wikipedia (WK)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-NEU-O	0.693
Líbímseti.cz (LI)	$\mathbf{A}_\mathrm{a} \to \mathbf{A}_\mathrm{b}$	A-NEU-O	0.639
CiteSeer (CS)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} + \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-EXP	0.684
English Wikipedia (WP)	$\mathbf{N}_\mathrm{a} + \mathbf{N}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	N-EXP	0.664
Google (GO)	$\mathbf{N}_\mathrm{a} + \alpha\mathbf{N}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{b}$	N-NEU-O	0.685
US patents (PC)	$\mathbf{A}_\mathrm{a} \to \mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{b}$	X-ABS	0.665
World Wide Web (W3)	$\mathbf{N}_\mathrm{a} + \alpha\mathbf{N}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b}$	N-NEU-O	0.692
English Wikipedia (EL)	$\mathbf{A}_\mathrm{a} + \mathbf{A}_\mathrm{a}^\mathrm{T} \to \mathbf{A}_\mathrm{b} + \mathbf{A}_\mathrm{b}^\mathrm{T}$	A-EXP	0.828

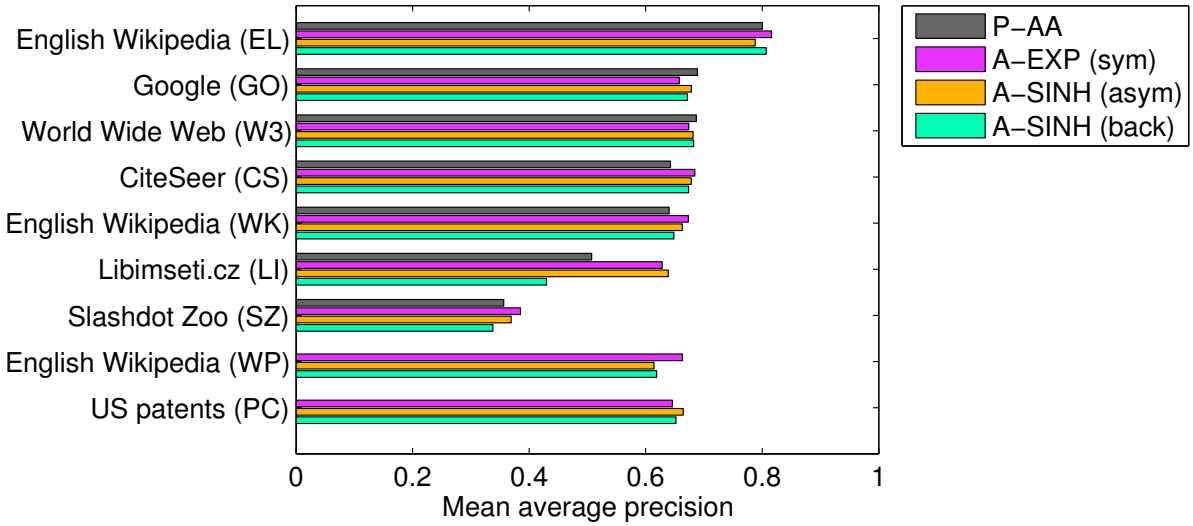


Figure 6.10: The evaluation results for link prediction on directed networks. See the text for a description of the datasets and link prediction methods. (sym) Using the eigenvalue decomposition of $\mathbf{A} + \mathbf{A}^\mathbf{T}$ (asym) Using the singular value decomposition of $\mathbf{A}$ (back) Using the singular value decomposition of $\mathbf{A} + \alpha\mathbf{A}^\mathrm{T}$.

networks with parallel arcs follow the model better than networks where only simple arcs are allowed.

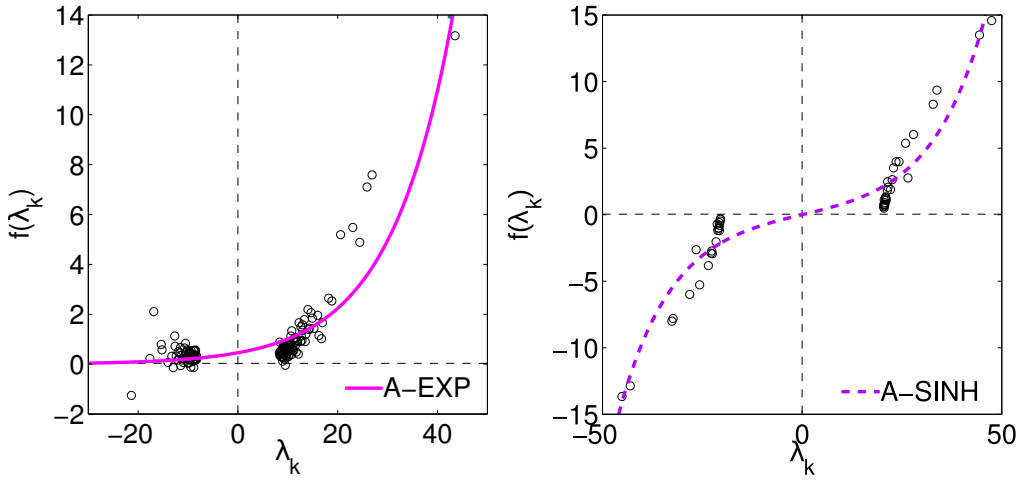
None of the three methods sym, asym and back is significantly better than the other two. The most accurate prediction is achieved by the sym method, but not by a wide margin.

6.4 Detecting Near-bipartite Networks

Some networks are not bipartite, but nearly so. An example would be a network of *fan* relationships between persons where there are clear *hubs* and *authorities*, i.e. popular persons and multiple fans. While these networks are not strictly bipartite, they are mostly bipartite in a sense that has to be made precise. Measures for the level of bipartivity exist in several forms [HLEK03, ERV05], and the curve fitting method described in this thesis offers another

method. Using the link prediction method described in Section 4.2.2, nearly bipartite graphs can be recognized by the centrally symmetric shape of the learned curve fitting function. This is true because if the shape of the curve is centrally symmetric, then the corresponding spectral transformation is an odd function, which contains only odd powers. Therefore, only paths of odd length will be useful to predict new edges in such a network, making the network (almost) bipartite.

Figure 6.11 shows the method applied to two unipartite networks: the Advogato trust network (AD) and the Twitter social network (Ws). The curves indicate that the Advogato trust network is *not* bipartite, while the Twitter social network network is nearly so. This result about Twitter confirms the characterization of Twitter as news media of [KLPM10], since the set of Twitter users can be divided into users reading news and users writing news, with most links from the first to the second group.



(a) Advogato trust network (AD)

(b) Twitter following network (Ws)

Figure 6.11: Detecting near-bipartite and non-bipartite networks: If the hyperbolic sine fits, the network is nearly bipartite; if the exponential fits, the network is not nearly bipartite. These graphs show the learned transformation of a graph's eigenvalues; see the text for a detailed description.

6.5 Summary: Bipartite and Directed Networks

While technically the link prediction problem in bipartite graphs is a subproblem of the general link prediction problem, the special structure of bipartite graphs makes common link prediction algorithms ineffective. In particular, none of the methods based on the triangle closing model can work in the bipartite case. Of the simple local link prediction methods, only the preferential attachment model can be used in bipartite networks.

Algebraic link prediction methods can be used instead, by restricting spectral transformations to odd functions, leading to odd power series, the matrix hyperbolic sine and an odd variant of the Neumann kernel as link prediction functions. As in the unipartite case, no single link prediction method is best for all datasets.

For directed networks, we conclude that the best matrix decomposition depends on the type of network. If the network has a high clustering coefficient, then edge directions can be ignored and the eigenvalue decomposition can be used. Otherwise, the network is likely to be nearly bipartite, and the singular value decomposition will lead to accurate prediction.

Chapter 7

Conclusion

This thesis has analysed the link prediction problem in large networks algebraically, and came to the conclusion that network growth follows a *spectral evolution model*. The link prediction problem is a general problem that generalizes the prediction of social links, the prediction of ratings, the prediction of link signs, and many other prediction problems encountered in networks. We showed that all these types of problems can be modeled by the spectral evolution model. While the spectral evolution model is based on the eigenvalue decomposition of a network's adjacency matrix, we could show that it is in fact much more universal than its definition using spectral graph theory suggests. For instance, it can be derived from a preferential attachment model restricted to individual subnetworks of a given set of networks. Other approaches that give rise to the spectral evolution model include diffusion in networks, sums over paths, rank reduction approaches, common graph kernels such as the matrix exponential and the Neumann kernel, and methods as simple as triangle closing. Given these findings, we suspect that many more derivations exist, and can state that the spectral evolution model plays a fundamental role for the link prediction problem.

To make sure however that the spectral evolution model is an actual feature of real-world networks, we had to make sure that it is specifically not implied by random graph growth models. Indeed, we found that the spectral evolution model does not hold in random graphs. Given these observations, we can state that spectral growth is a fundamental feature of the dynamics of real networks.

To prove the universality of the spectral evolution model, we also had to make sure it is observed in many networks. By studying a collection of one hundred and eighteen network datasets, we could confirm that the spectral evolution model is indeed universal. The spectral evolution model was shown to hold for networks of many types: weighted, unweighted, signed, unipartite, bipartite, directed and undirected networks. These networks types encompass almost all networks present on the Web and elsewhere, confirming the universality of the spectral evolution model.

7.1 Findings

The main result of this thesis is the spectral evolution model, which was found to hold for almost all real-world networks. Beyond this result, this thesis includes a certain number of additional novel ideas.

New Datasets The experiments in this thesis were performed on one hundred and eighteen large network datasets. This corpus of network datasets presents a valuable study in itself. In Chapter 2, we were able to confirm a certain number of known network properties on this

dataset collection, making this thesis one of the largest study of this kind that we know of. In particular, we introduced the Slashdot Zoo dataset, a large signed social network of *friends* and *foes* of the technology site Slashdot. This dataset is available publicly online[1].

Spectral Evolution Model We introduced the spectral evolution model, which states that the temporal evolution of real-world networks can be described by the change in the eigenvalue decomposition of their adjacency matrix. We validated this model empirically in over hundred datasets, as well as theoretically by showing that it generalizes a certain number of link prediction methods such as graph kernels, rank reduction, path counting and triangle closing. We also showed how it applies to bipartite and directed networks using the singular value decomposition. We also introduced the latent preferential attachment model and showed it to be equivalent to the spectral evolution model.

New Link Prediction Methods By systematically applying spectral transformations to different network types we were able to identify several as yet unknown link prediction methods. Of note are the signed Laplacian graph kernels, which apply to networks with positive and negative edges, and the hyperbolic sine and odd Neumann pseudokernels, which apply to bipartite networks. All three methods perform competitively on the respective task of link prediction.

Learning Spectral Transformations The spectral evolution model describes growth as a spectral transformation, but does not predict any graph kernel in particular. Based on this observation, we developed two new link prediction methods, both assuming the spectral evolution model and learning a spectral transformation using historical data. The first method is based on the reduction of the link prediction function learning problem to a one-dimensional curve fitting problem that can be solved efficiently, the second uses extrapolation of the spectrum to learn new eigenvalues. The methods differ by additional assumptions they make about graph growth: The curve fitting method needs a graph kernel to be chosen and learns its parameters, while the extrapolation method does not assume any specific graph kernel, and can also learn irregular graph growth. We observed graph growth to be sometimes regular and sometimes irregular, and therefore both methods have their justification, and indeed their relative accuracy depends on the dataset chosen.

Signed Laplacian Matrix While the Laplacian matrix of signed graphs was known before in the mathematics literature, we gave, to our knowledge, the first application of it to network analysis. Based on this, we introduced a new drawing algorithm for signed graphs, defined the algebraic conflict which characterizes the amount of conflict present in signed graphs, introduced signed cuts to solve the signed clustering problem, introduced signed Laplacian graph kernels and introduced the signed resistance distance.

7.2 Outlook

Nonorthogonal and Nondiagonal Decompositions In this thesis we have only looked at the eigenvalue and singular value decompositions. Both decompositions result in a product of orthogonal and diagonal matrices. As an alternative, other matrix decompositions exist, and are sometimes applied to adjacency matrices.

[1] dai-labor.de/IRML/datasets

Of particular note are nonnegative decompositions, where orthogonal matrices are replaced by nonnegative matrices, and square asymmetric decompositions such as the Schur decomposition. In both cases, one loses the equivalence between power sums of the original matrix corresponding to power sums of the diagonal matrix, which is the basis for the spectral evolution model described in this thesis.

However, these alternative decompositions also have their advantages. A nonnegative factorization is usually justified by the fact that negative weights (which are inevitable in orthogonal matrices) are hard to interpret as weights in an eigenpair. As mentioned in Section 3.3.5, individual eigenvector components can be interpreted as degrees, making a case for nonnegative factors in a decomposition, at least on a conceptual level. In light of the equivalence between the joint diagonalization of matrices and tensor decomposition (see Section 4.5), an obvious open question is whether nonnegative tensor factorization can be applied to unweighted link prediction.

As for nondiagonal decompositions such as the Schur decomposition, they still allow a power sum of the adjacency matrix to be mapped to a power sum of the central matrix in the decomposition, but powers of this matrix are then harder to compute. The Schur decomposition is by its nature applicable to directed, unipartite networks, and an open task is to generalize the kernel learning methods of Chapter 4 to nondiagonal decompositions.

Link Prediction Algebra The work in this thesis applies to unipartite and bipartite networks, unweighted or weighted by real numbers. Implicitly, all algebraic link prediction functions can be understood to apply the operations of real addition and multiplication to edge weights, using the following meaning: The weight of adjacent edges is multiplied to compute the weight of paths, and the weights of parallel paths is then added. A conceptually simple but computationally nontrivial extension consists in using more complex operations instead of addition and multiplication. This results in link prediction functions that are not based on the eigenvalue decomposition. For instance, shortest paths in networks can be understood algebraically by taking powers of the adjacency matrix under the min-plus algebra [GM08].

Semantic Networks A recent trend in network mining is the labeling of edges with relationship types, giving multirelational or semantic networks. The algebraic approach described in this thesis can be applied to such networks provided that a mapping from relationship types to edge weights can be found. This represents a learning problem in itself, which is related to the previous approach, since the question of which algebraic structure to use is still open in this case.

Appendix A

List of Datasets

This is the complete list of network datasets used in this thesis. The *Prop.* column refers to the network properties given in Table 2.1.

Table A.1: The list of all one hundred and eighteen datasets.

Ref.	Prop.	Code	Dataset	Node and edge types	$\lvert V\rvert$	$\lvert E\rvert$ or $\lvert A\rvert$
Authorship						
[Wik10]	B=⊜	ar	Arabic Wikipedia	User–article edit	45,796 + 749,714	4,578,676
[LKF07]	U−	AP	arXiv astro-ph	Author–author collaboration	18,772	396,160
[LKF07]	U−⊜	PH	arXiv hep-ph	Author–author collaboration	28,093	12,730,098
[LKF07]	U−⊜	TH	arXiv hep-th	Author–author collaboration	22,908	11,209,368
[Wik10]	B=⊜	eu	Basque Wikipedia	User–article edit	4,163 + 129,814	1,572,704
[Wik10]	B=⊜	bn	Bengali Wikipedia	User–article edit	2,313 + 107,110	667,265
[Wik10]	B=⊜	br	Breton Wikipedia	User–article edit	2,018 + 77,915	744,824
[Wik10]	B=⊜	ca	Catalan Wikipedia	User–article edit	21,807 + 655,913	5,401,060
[Wik10]	B=⊜	zh	Chinese Wikipedia	User–article edit	113,194 + 1,095,945	11,645,429
[Ley02]	B−	Pa	DBLP	Author–publication authorship	849,449 + 2,275,386	4,600,857
[Ley02]	U=⊜	Pc	DBLP	Author–author collaboration	801,474	9,510,516
[Wik10]	B=⊜	nl	Dutch Wikipedia	User–article edit	112,771 + 1,518,053	18,930,176
[Wik10]	B=⊜	ben	English Wikibooks	User–article edit	32,583 + 134,942	1,164,576
[Wik10]	B=⊜	nen	English Wikinews	User–article edit	10,764 + 163,008	901,416
[Wik10]	B=⊜	en	English Wikipedia	User–article edit	3,819,691 + 21,504,191	266,769,613
[Wik10]	B=⊜	qen	English Wikiquote	User–article edit	21,607 + 94,756	549,210
[Wik10]	B=⊜	men	English Wiktionary	User–article edit	29,348 + 2,104,544	8,998,641
[Wik10]	B=⊜	eo	Esperanto Wikipedia	User–article edit	7,211 + 296,542	2,849,624
[Wik10]	B=⊜	bfr	French Wikibooks	User–article edit	2,884 + 28,113	201,727
[Wik10]	B=⊜	nfr	French Wikinews	User–article edit	1,408 + 25,138	193,618
[Wik10]	B=⊜	fr	French Wikipedia	User–article edit	288,275 + 4,022,276	46,168,355
[Wik10]	B=⊜	mfr	French Wiktionary	User–article edit	5,017 + 1,907,247	7,399,298
[Wik10]	B=⊜	gl	Galician Wikipedia	User–article edit	5,200 + 138,647	1,497,798
[Wik10]	B=⊜	de	German Wikipedia	User–article edit	425,842 + 3,195,148	57,323,775
[Wik10]	B=⊜	mde	German Wiktionary	User–article edit	5,824 + 146,158	1,229,501
[Cha09]	B−	GH	Github	User–project membership	56,519 + 120,867	440,237
[Wik10]	B=⊜	el	Greek Wikipedia	User–article edit	13,029 + 136,875	1,837,141
[Wik10]	B=⊜	ht	Haitian Wikipedia	User–article edit	934 + 59,020	373,818
[Wik10]	B=⊜	it	Italian Wikipedia	User–article edit	137,693 + 2,255,875	26,241,217
[Wik10]	B=⊜	ja	Japanese Wikipedia	User–article edit	190,752 + 1,865,207	20,411,185
[Wik10]	B=⊜	lv	Latvian Wikipedia	User–article edit	4,099 + 107,416	946,173
[Wik10]	B=⊜	nds	Low German Wikipedia	User–article edit	1,692 + 34,074	394,632
[ABK+08]	B−	ST	Movies	Movie–actor starring	54,803 + 69,819	210,591
[Wik10]	B=⊜	oc	Occitan Wikipedia	User–article edit	1,605 + 58,276	714,503
[Wik10]	B=⊜	pl	Polish Wikipedia	User–article edit	106,116 + 1,306,026	17,959,638
[Wik10]	B=⊜	pt	Portuguese Wikipedia	User–article edit	147,287 + 2,346,137	16,082,508
[Wik10]	B=⊜	ru	Russian Wikipedia	User–article edit	135,409 + 2,069,952	21,374,632
[Wik10]	B=⊜	sr	Serbian Wikipedia	User–article edit	11,794 + 388,830	3,243,236
[Wik10]	B=⊜	sk	Slovak Wikipedia	User–article edit	12,166 + 266,816	2,812,157
[Wik10]	B=⊜	es	Spanish Wikipedia	User–article edit	335,344 + 2,953,054	27,011,506
[Wik10]	B=⊜	sv	Swedish Wikipedia	User–article edit	71,516 + 970,835	10,084,464
[Wik10]	B=⊜	vi	Vietnamese Wikipedia	User–article edit	24,153 + 447,280	2,756,694
[Wik10]	B=⊜	cy	Welsh Wikipedia	User–article edit	2,419 + 70,287	819,482
Communication						
[CSJS09]	D=⊜	DG	Digg	User–user reply	30,398	87,627
[LHK10a]	D−	WK	English Wikipedia	User–user userpage message	2,394,385	5,021,410
[KY04]	D=⊜	EN	Enron	User–user email	87,365	1,149,884
[LKF07]	D−	EU	EU institution	User–user email	265,214	420,045
[VMCG09]	D=⊜	Ow	Facebook New Orleans	User–user wall post	63,891	876,993
[SDLA10]	B=⊜	Fc	Filmtipset	User–movie comment	29,530 + 45,830	1,266,753
[BPSDGA04]	U=	PG	Pretty Good Privacy	User–user interaction	10,680	24,340
[CLS+10]	D=⊜	Wa	Twitter	User–user "@name" mention	2,919,613	12,887,063
[GDDG+03]	D−	A@	U. Rovira i Virgili	User–user email	1,133	10,902
Co-occurrence						
[LAH07]	U−	AM	Amazon	Item–item co-purchase	403,394	3,387,388
[ABK+08]	U−	SI	Similarity	Entity–entity similarity	1,944	3,050
Features						
[ABK+08]	B−	CN	Countries	Entity–country location/origin	357,872 + 2,411	383,891
[Wik10]	B=	EX	English Wikipedia	Excellent article–word frequency	2,780 + 273,959	2,941,902
[Wik10]	B−	WC	English Wikipedia	Article–category membership	1,853,493 + 182,947	3,795,796
[MMG+07]	B−	FG	Flickr	User–group membership	395,979 + 103,631	8,545,307
[ABK+08]	B−	GE	Genre	Entity–genre style	227,040 + 10,970	378,380
[MMG+07]	B−	LG	LiveJournal	User–group membership	3,201,203 + 7,489,073	112,307,385

Continued on next page

Ref.	Prop.	Code	Dataset	Node and edge types	$\|V\|$	$\|E\|$ or $\|A\|$
[MMG+07]	B−	OG	Orkut	User–group membership	2,783,196 + 8,730,857	327,037,487
[Pro10]	B−	PS	Prosper.com	Member–group support	6,582 + 1,013	21,017
[Pro10]	B−	PW	Prosper.com	Member–listing watch	1,732 + 22,233	35,377
[ABK+08]	B−	RL	Record labels	Artist–record label membership	140,446 + 18,767	191,558
[LYRL04]	B=	R2	Reuters-21578	Article–word frequency	21,557 + 38,677	978,446
[LYRL04]	B=	RE	Reuters	Message–word inclusion	781,265 + 283,911	60,569,726
[ABK+08]	B−	TM	Teams	Athlete–team membership	97,469 + 29,640	401,494
[Nat10]	B=	TR	TREC (disks 4–5)	Document–word frequency	556,077 + 1,173,225	83,629,405
[Mis09]	B−	YG	YouTube	User–group membership	94,238 + 30,087	293,360
Folksonomy						
[HJSS06]	T−⊖	B	BibSonomy	User–tag–publication assignment	5,794 + 204,673 + 771,290	2,555,080
[EC07]	T=⊖	C	CiteULike	User–tag–publication assignment	22,715 + 153,277 + 731,769	2,411,819
[WZB08]	T=⊖	D	Delicious	User–tag–URL assignment	833,081 + 4,512,303 + 33,778,416	301,187,709
[Gro06]	T=⊖	M	MovieLens	User–tag–movie assignment	4,009 + 16,528 + 7,601	95,580
[CLS+10]	T=⊖	W	Twitter	User–hashtag–URL usage	175,214 + 530,418 + 9,129,669	4,664,605
[NO10]	T=	V	vi.sualize.us	User–tag–picture assignment	17,122 + 82,035 + 495,402	2,298,816
Interaction						
[CHC+07]	U=⊖	CO	Haggle	Person–person contact	274	28,244
[DA05]	U=	PM	*Caenorhabditis elegans*	Metabolite–metabolite interaction	453	4,596
[Cel10]	B=⊖	Lb	Last.fm	User–band listening event	992 + 174,077	19,150,868
[Cel10]	B=⊖	Ls	Last.fm	User–song listening event	992 + 1,084,620	19,150,868
[ESP06]	U=⊖	RM	Reality Mining	Person–person proximity	96	1,086,404
Physical						
[LKF07]	U−	IN	CAIDA	AS–AS connection	26,475	106,762
[LLDM08]	U−	RO	California	Road network	1,965,206	5,533,214
[RFI02]	D−	GN	Gnutella	Host–host connection	62,586	147,892
[ZLMZ05]	U=⊖	TO	Internet topology	AS–AS connection	34,761	171,403
[LKF07]	U−	AS	Route Views	AS–AS connection	6,474	13,895
[LKF07]	U−	SK	Skitter	AS–AS connection	1,696,415	11,095,298
Ratings						
[ZMKL05]	B⋆	BX	BookCrossing	User–book rating	105,283 + 340,532	1,149,742
[MA05]	B⋆⊖	ER	Epinions	User–product rating	120,492 + 755,760	13,668,320
[SDLA10]	B⋆⊖	Fr	Filmtipset	User–movie rating	80,482 + 64,189	19,554,219
[GRGP01]	B±	JE	Jester	User–joke rating	24,938 + 100	616,912
[BP07]	D⋆	LI	Líbímseti.cz	User–user profile rating	220,970	17,359,346
[Gro06]	B⋆⊖	M1	MovieLens 100k	User–movie rating	943 + 1,682	100,000
[Gro06]	B⋆⊖	M3	MovieLens 10M	User–movie rating	71,567 + 65,133	10,000,054
[Gro06]	B⋆⊖	M2	MovieLens 1M	User–movie rating	6,040 + 3,706	1,000,209
[BL07]	B⋆⊖	NX	Netflix	User–movie rating	480,189 + 17,770	100,480,507
Reference						
[GGK03]	D−	th	arXiv hep-th	Publication–publication citation	27,770	352,807
[LLDM08]	D−	BS	Berkeley/Stanford	Webpage–webpage hyperlink	685,230	7,600,595
[BLG98]	D−	CS	CiteSeer	Publication–publication citation	723,131	1,764,929
[Ley02]	D−	Pi	DBLP	Publication–publication citation	12,591	49,793
[Mis09]	D−⊖	WP	English Wikipedia	Article–article link	1,870,709	39,953,145
[ABK+08]	D−	DL	English Wikipedia	Article–article link	15,172,739	130,166,251
[LLDM08]	D−	GO	Google	Webpage–webpage hyperlink	875,713	5,105,039
[PFP+07]	D−	GC	Google.com internal	Webpage–webpage hyperlink	15,763	171,206
[BCH03]	D−	WT	TREC WT10g	Webpage–webpage hyperlink	1,601,787	8,063,026
[HJT01]	D−	PC	US patents	Patent–patent citation	3,774,768	16,522,438
[AJB99]	D−	W3	World Wide Web	Webpage–webpage hyperlink	325,729	1,497,135
Social						
[VMCG09]	U−⊖	OI	Facebook New Orleans	User–user friendship	63,731	1,545,686
[SDLA10]	D−	Ff	Filmtipset	User–user friendship	39,199	143,310
[MKG+08]	D−⊖	FL	Flickr	User–user friendship	2,302,925	33,140,018
[LLDM08]	D−	LJ	LiveJournal	User–user friendship	4,847,571	68,993,773
[MMG+07]	D−	OR	Orkut	User–user link	3,072,441	223,534,301
[KLB09]	D±	SZ	Slashdot Zoo	User–user friend/foe	79,120	515,581
[KLPM10]	D−	TW	Twitter	User–user following	41,652,230	1,468,365,182
[CLS+10]	D−	Ws	Twitter	User–user following	465,017	835,423
[Mis09]	D−⊖	YT	YouTube	User–user friendship	3,223,643	18,524,095
Trust						
[Ste05]	D=	AD	Advogato	User–user trust	6,535	51,397
[LHK10a]	D±⊖	EL	English Wikipedia	Editor–editor vote	8,297	107,071
[MA05]	D±⊖	EP	Epinions	User–user trust/distrust	131,828	841,372

Table A.2 below gives basic statistics for all datasets. The individual statistics are defined in Section 2.2.2. Statistics that cannot be computed for the dataset are marked with a dash (—), for instance the clustering coefficient cannot be computed for bipartite graphs. In that table, the rank r is the rank used for all matrix decompositions of each network. The effective diameter $\delta_{0.9}$ is defined in Section 2.2.2. The density d is defined in Section 2.2.1. The clustering coefficient is defined in Equation 2.34, and only exists for unipartite networks. For signed networks, we give the signed clustering coefficient as defined in [KLB09]. The power-law exponent γ is defined in Section 2.2.2. For directed network, we give the outdegree and indegree power-law exponents separately. For bipartite networks, we give the power-law exponents of the two vertex sets separately. The largest connected component (CC) and largest strongly connected component (SCC) are given relatively to the total number of vertices. The strongly connected component is only defined for directed networks.

Table A.2: Statistics for all one hundred and eighteen datasets.

Dat.	Rank (r)	Eff. diam. ($\delta_{0.9}$)	Density (d)	Clust. coeff. (c)	Power-law (γ)	Largest CC	Largest SCC
Authorship							
Arabic Wikipedia (ar)	49	4.0	100.0, 6.1	—	1.7, 2.5	0.984	—
arXiv astro-ph (AP)	73	5.5	42.2	0.39	2.0	0.954	—
arXiv hep-ph (PH)	49	3.7	906.3	0.29	1.4	0.998	—
arXiv hep-th (TH)	49	3.7	978.6	0.28	1.4	0.992	—
Basque Wikipedia (eu)	49	3.9	377.8, 12.3	—	1.5, 2.7	0.983	—
Bengali Wikipedia (bn)	49	4.2	288.5, 6.2	—	1.5, 1.8	0.991	—
Breton Wikipedia (br)	49	4.4	369.1, 9.6	—	1.5, 3.0	0.988	—
Catalan Wikipedia (ca)	49	4.2	247.7, 8.3	—	1.6, 2.3	0.989	—
Chinese Wikipedia (zh)	49	4.7	102.9, 10.7	—	1.6, 2.3	0.990	—
DBLP (Pa)	49	15.1	5.4, 2.0	—	2.2, 5.2	0.853	—
DBLP (Pc)	49	7.7	23.7	0.11	2.2	0.857	—
Dutch Wikipedia (nl)	49	4.1	167.9, 12.5	—	1.7, 2.4	0.991	—
English Wikibooks (ben)	49	5.2	35.7, 8.8	—	1.9, 2.3	0.952	—
English Wikinews (nen)	49	5.1	83.7, 5.6	—	1.7, 2.7	0.963	—
English Wikipedia (en)	49	4.5	69.8, 12.5	—	1.8, 1.9	0.977	—
English Wikiquote (qen)	49	4.1	25.4, 5.9	—	1.9, 2.3	0.953	—
English Wiktionary (men)	49	4.0	306.6, 4.3	—	1.6, 2.5	0.993	—
Esperanto Wikipedia (eo)	49	4.0	395.2, 9.8	—	1.5, 2.4	0.981	—
French Wikibooks (bfr)	102	5.1	69.9, 7.3	—	1.8, 2.5	0.967	—
French Wikinews (nfr)	123	4.5	137.5, 7.7	—	1.6, 3.4	0.980	—
French Wikipedia (fr)	49	4.5	160.2, 11.6	—	1.7, 1.8	0.984	—
French Wiktionary (mfr)	49	4.0	1474.8, 3.9	—	1.5, 3.4	0.999	—
Galician Wikipedia (gl)	49	4.0	288.0, 11.0	—	1.5, 2.4	0.979	—
German Wikipedia (de)	49	5.1	134.6, 18.4	—	1.6, 1.7	0.963	—
German Wiktionary (mde)	49	4.0	211.1, 8.5	—	1.6, 3.4	0.987	—
Github (GH)	49	6.1	7.8, 3.6	—	1.8, 2.0	0.788	—
Greek Wikipedia (el)	49	4.0	141.0, 13.5	—	1.6, 2.6	0.983	—
Haitian Wikipedia (ht)	66	4.5	400.2, 6.7	—	1.5, 2.1	0.944	—
Italian Wikipedia (it)	49	4.2	190.6, 11.8	—	1.6, 2.3	0.983	—
Japanese Wikipedia (ja)	49	5.1	107.0, 11.6	—	1.6, 2.4	0.942	—
Latvian Wikipedia (lv)	49	4.0	230.8, 8.8	—	1.5, 2.8	0.989	—
Low German Wikipedia (nds)	91	4.8	233.2, 11.7	—	1.6, 2.1	0.975	—
Movies (ST)	49	11.5	3.8, 3.0	—	5.2, 2.0	0.835	—
Occitan Wikipedia (oc)	59	3.9	445.2, 12.3	—	1.5, 1.9	0.991	—
Polish Wikipedia (pl)	49	4.7	169.2, 14.0	—	1.6, 2.4	0.975	—
Portuguese Wikipedia (pt)	49	4.3	109.2, 6.9	—	1.7, 2.5	0.986	—
Russian Wikipedia (ru)	49	4.8	157.9, 10.5	—	1.6, 2.3	0.980	—
Serbian Wikipedia (sr)	49	4.0	275.0, 8.4	—	1.6, 2.6	0.991	—
Slovak Wikipedia (sk)	49	4.0	231.1, 10.6	—	1.6, 2.8	0.991	—
Spanish Wikipedia (es)	49	4.0	80.5, 9.3	—	1.7, 2.3	0.973	—
Swedish Wikipedia (sv)	49	4.7	141.0, 10.5	—	1.7, 2.6	0.982	—
Vietnamese Wikipedia (vi)	49	4.3	114.1, 6.3	—	1.7, 2.0	0.971	—
Welsh Wikipedia (cy)	49	4.0	338.8, 11.7	—	1.5, 2.8	0.989	—
Communication							
Digg (DG)	58	5.8	5.8	0.009	2.8, 3.0	0.975	0.222
English Wikipedia (WK)	9	3.9	4.2	0.02	1.6, 2.5	0.998	0.047
Enron (EN)	49	5.8	26.3	0.089	1.5, 1.7	0.967	0.105
EU institution (EU)	49	5.0	3.2	0.23	3.0, 2.7	0.848	0.129
Facebook New Orleans (Ow)	49	7.5	37.4	0.096	2.1, 2.1	0.688	0.000
Filmtipset (Fc)	49	3.8	42.9, 27.6	—	1.8, 1.6	0.998	—
Pretty Good Privacy (PG)	155	10.0	4.6	0.41	2.2	1.000	—
Twitter (Wa)	49	5.9	8.8	0.026	1.9, 1.9	0.991	0.000
U. Rovira i Virgili (A@)	700	4.6	19.2	0.17	4.1, 4.1	1.000	1.000
Co-occurrence							
Amazon (AM)	49	7.7	16.8	0.37	3.9	1.000	—
Similarity (SI)	100	14.3	3.1	0.46	3.9	0.568	—
Features							
Countries (CN)	49	5.8	1.1, 159.2	—	4.4, 1.5	0.971	—
English Wikipedia (EX)	49	1.8	1058.2, 10.7	—	7.2, 1.6	1.000	—
English Wikipedia (WC)	49	13.8	2.0, 20.7	—	7.1, 2.1	0.930	—
Flickr (FG)	49	5.3	21.6, 82.5	—	1.5, 1.7	0.941	—
Genre (GE)	49	7.3	1.7, 34.5	—	5.8, 1.6	0.954	—
LiveJournal (LG)	49	3.8	35.1, 15.0	—	1.4, 3.1	0.982	—

Continued on next page

Dat.	Rank (r)	Eff. diam. ($\delta_{0.9}$)	Density (d)	Clust. coeff. (c)	Power-law (γ)	Largest CC	Largest SCC
Orkut (OG)	49	3.9	117.5, 37.5	—	1.3, 1.6	0.999	—
Prosper.com (PS)	395	3.9	3.2, 20.7	—	2.0, 1.6	0.965	—
Prosper.com (PW)	155	6.4	20.4, 1.6	—	1.5, 10.7	0.903	—
Record labels (RL)	49	7.0	1.4, 10.2	—	6.0, 1.8	0.876	—
Reuters-21578 (R2)	49	1.9	49.5, 25.3	—	2.0, 1.6	0.970	—
Reuters (RE)	49	2.1	77.5, 213.3	—	1.3, 1.6	1.000	—
Teams (TM)	49	7.9	4.1, 13.5	—	9.0, 1.6	0.913	—
TREC (disks 4–5) (TR)	49	1.9	151.6, 71.3	—	1.2, 2.2	0.998	—
YouTube (YG)	49	5.7	3.1, 9.8	—	2.8, 2.2	0.913	—
Folksonomy							
BibSonomy (B_{ut})	49	3.9	441.0, 12.5, 3.3	—	1.4, 1.8, 3.5	0.995	—
CiteULike (C_{ut})	49	3.8	106.2, 15.7, 3.3	—	1.4, 1.7, 2.6	0.989	—
Delicious (D_{ut})	49	3.6	361.5, 66.7, 8.9	—	1.2, 2.2, 2.0	0.999	—
MovieLens (M_{ut})	146	5.3	23.8, 5.8, 12.6	—	1.7, 1.9, 2.6	0.958	—
Twitter (W_{ut})	49	4.8	26.6, 8.8, 1.4	—	2.2, 1.8, 2.3	0.979	—
vi.sualize.us (V_{ut})	49	3.6	134.3, 28.0, 4.6	—	1.7, 1.8, 3.1	0.993	—
Interaction							
Haggle (CO)	5	3.6	206.2	0.74	1.4	1.000	—
Caenorhabditis elegans (PM)	150	3.7	20.3	0.077	2.2	1.000	—
Last.fm (Lb)	49	1.8	19305.3, 110.0	—	1.2, 1.6	1.000	—
Last.fm (Ls)	49	1.8	19305.3, 17.7	—	1.2, 2.0	1.000	—
Reality Mining (RM)	96	1.8	22633.4	0.75	1.1	1.000	—
Physical							
CAIDA (IN)	63	5.0	8.1	0.42	2.1	1.000	—
California (RO)	15	495.8	5.6	0.06	7.0	0.996	—
Gnutella (GN)	49	7.3	4.7	0	4.3, 5.6	1.000	0.226
Internet topology (TO)	49	4.6	9.9	0.086	1.9	1.000	—
Route Views (AS)	247	5.0	4.3	0.077	2.1	1.000	—
Skitter (SK)	49	5.9	13.1	0.059	1.6	0.999	—
Ratings							
BookCrossing (BX)	49	6.4	10.9, 3.4	—	1.8, 2.1	0.942	—
Epinions (ER)	49	5.4	113.4, 18.1	—	1.6, 2.0	0.999	—
Filmtipset (Fr)	49	2.0	243.0, 304.6	—	1.5, 1.5	1.000	—
Jester (JE)	100	1.8	24.7, 6169.1	—	5.4, 1.1	1.000	—
Líbímseti.cz (LI)	19	3.7	157.1	-0.00028	1.8, 1.7	1.000	0.367
MovieLens 100k (M1)	100	1.9	106.0, 59.5	—	1.8, 1.8	1.000	—
MovieLens 10M (M3)	49	2.0	143.1, 936.6	—	1.8, 1.4	0.589	—
MovieLens 1M (M2)	100	1.9	165.6, 269.9	—	1.7, 1.5	1.000	—
Netflix (NX)	49	2.4	209.3, 5654.5	—	1.2, 1.2	1.000	—
Reference							
arXiv hep-th (th)	75	5.7	25.4	0.27	3.5, 2.4	0.987	0.269
Berkeley/Stanford (BS)	49	10.2	22.2	0.81	2.7, 2.0	0.956	0.038
CiteSeer (CS)	75	8.1	9.2	0.16	4.0, 2.5	0.505	0.022
DBLP (Pi)	126	5.6	7.9	0.16	3.6, 2.7	0.992	0.000
English Wikipedia (DL)	49	5.7	17.2	0.031	1.8, 2.1	0.995	0.312
English Wikipedia (WP)	49	3.9	42.7	0.021	1.4, 1.6	1.000	0.871
Google.com internal (GC)	92	3.5	21.7	0.56	3.2, 2.0	1.000	0.784
Google (GO)	49	7.9	11.7	0.39	3.9, 2.5	0.977	0.497
TREC WT10g (WT)	9	10.8	10.1	0.067	2.1, 2.7	0.910	0.294
US patents (PC)	49	9.6	8.8	0.11	3.4, 3.7	0.997	0.000
World Wide Web (W3)	75	9.8	9.2	0.72	2.1, 2.0	1.000	0.039
Social							
Facebook New Orleans (OI)	70	5.5	48.5	0.15	1.9	0.995	—
Filmtipset (Ff)	49	7.5	7.3	0.24	3.8, 3.7	0.931	0.697
Flickr (FL)	49	7.1	28.8	0.15	2.1, 2.2	0.944	0.697
LiveJournal (LJ)	49	6.9	28.5	0.17	1.6, 1.6	0.999	0.790
Orkut (OR)	49	5.4	145.5	0.061	1.3, 1.3	1.000	0.976
Slashdot Zoo (SZ)	60	5.3	13.0	0.02	1.9, 2.2	1.000	0.341
Twitter (TW)	49		70.5			—	
Twitter (Ws)	49	5.6	3.6	0.0089	1.4, 2.5	1.000	0.000
YouTube (YT)	49	6.0	11.5	0.016	2.2, 2.2	0.998	0.983
Trust							
Advogato (AD)	150	3.9	15.7	0.1	2.9, 2.9	0.771	0.482
English Wikipedia (EL)	164	3.8	30.1	0.084	1.5, 2.2	0.852	0.157
Epinions (EP)	70	5.5	12.8	0.1	1.7, 1.7	0.904	0.314

Appendix B

Complete Experimental Results

The following tables give evaluation results for all datasets. Values indicate the mean average precision. The first two tables B.1 and B.2 shows unipartite networks; Table B.2 shows additional link prediction methods for directed networks; and Table B.3 shows bipartite networks. Datasets are referred to by their code given in Table A.1. The best performing link prediction methods are highlighted in each line. All experiments were run on a 64-bit Linux machine with 16 cores (Intel Xeon X5550, 2.67 GHz) and 72 GiB of RAM. Despite the large amount of memory and processing power available, it was not possible to compute the eigenvalue or singular value decomposition of all networks. Therefore, some combinations are left out in the table. In general, it took more runtime to decompose the normalized adjacency matrix than the non-normalized adjacency matrix, and it took more memory to decompose the Laplacian matrix than it took to decompose either normalized or non-normalized adjacency matrix.

Table B.1: Evaluation results for all square datasets (default methods).

Dat.	P-AA	A-POLYN	A-EXP	A-NEU	N-POLYN	N-EXP	N-NEU	N-COM	L-COM	L-REG	L-HEAT	X-SQU
Authorship												
AP	**0.693**	0.690	0.690	0.690	0.672	0.673	0.481	0.673	0.578	0.580	0.579	0.690
PH	**0.765**	0.728	0.726	0.714	0.752	0.752	0.639	0.751	—	—	—	0.730
TH	**0.766**	0.720	0.712	0.691	0.751	0.751	0.648	0.752	—	—	—	0.720
Pc	—	0.758	**0.759**	0.758	—	—	—	—	—	—	—	0.758
Communication												
DG	0.771	0.793	0.811	0.804	0.837	0.842	0.767	**0.843**	0.696	0.697	0.697	0.795
WK	0.640	0.665	0.673	0.673	**0.684**	0.676	0.142	0.671	—	—	—	0.661
EN	0.872	0.779	0.775	0.741	**0.875**	**0.875**	0.711	**0.875**	0.727	0.729	0.729	0.777
EU	0.671	0.663	0.672	0.665	0.692	**0.693**	0.093	**0.693**	0.556	0.561	0.561	0.661
Ow	**0.795**	0.783	0.786	0.784	**0.795**	**0.795**	0.599	0.794	—	—	—	0.785
PG	0.965	0.944	0.967	0.963	0.964	0.979	0.766	**0.982**	0.967	0.970	0.968	0.937
Wa	—	0.765	0.768	**0.769**	—	—	—	—	—	—	—	0.764
A@	0.866	0.952	0.961	0.901	0.959	0.969	0.644	0.964	**0.970**	0.924	0.926	0.933
Co-occurrence												
AM	**0.686**	0.672	0.673	0.672	0.660	0.660	0.627	0.660	—	—	—	0.672
SI	0.893	0.880	0.937	0.910	0.957	0.968	0.779	0.982	0.964	**0.995**	**0.995**	0.879
Interaction												
CO	0.726	0.727	**0.737**	**0.737**	0.653	0.671	0.552	0.693	0.406	0.406	0.406	0.727
PM	0.938	0.889	0.926	0.887	0.917	0.938	0.615	**0.939**	0.862	0.858	0.856	0.873
RM	0.287	0.324	0.293	0.296	0.275	0.259	0.172	0.268	—	—	—	**0.330**
Physical												
IN	0.811	0.966	0.963	0.943	0.982	0.982	0.569	**0.984**	0.837	0.844	0.844	0.963
RO	0.532	**0.614**	**0.614**	**0.614**	—	—	—	—	—	—	—	**0.614**
GN	0.507	0.580	0.615	**0.637**	0.581	0.583	0.468	0.585	0.475	0.475	0.475	0.580
TO	**0.728**	0.683	0.723	0.721	0.711	0.711	0.248	0.717	0.575	0.579	0.579	0.672
AS	0.905	0.814	0.900	0.885	0.867	0.923	0.799	0.930	**0.954**	0.936	0.939	0.794
SK	—	0.683	**0.684**	0.678	—	—	—	—	—	—	—	0.683
Ratings												
LI	0.507	**0.635**	0.628	0.623	0.556	0.542	0.441	0.543	—	—	—	**0.635**
Reference												
th	**0.690**	0.682	0.685	0.681	0.667	0.667	0.460	0.668	0.608	0.610	0.609	0.680
BS	—	**0.686**	0.682	0.681	—	—	—	—	—	—	—	0.683
CS	0.643	0.682	**0.684**	**0.684**	0.673	0.673	0.423	0.674	—	—	—	0.681
Pi	0.908	0.955	**0.959**	0.941	0.950	0.949	0.760	0.949	0.910	0.913	0.913	0.954
WP	—	0.641	0.663	0.660	**0.664**	**0.664**	0.402	0.662	—	—	—	0.639
GC	**0.688**	0.682	0.680	0.671	0.665	0.669	0.342	0.681	0.581	0.604	0.604	0.675

Continued on next page

Dat.	P-AA	A-POLYN	A-EXP	A-NEU	N-POLYN	N-EXP	N-NEU	N-COM	L-COM	L-REG	L-HEAT	X-SQU
GO	**0.689**	0.660	0.658	0.657	0.684	0.684	0.429	0.683	—	—	—	0.658
WT	—	**0.604**	0.596	0.594	—	—	—	—	—	—	—	0.603
PC	—	0.636	**0.646**	0.644	—	—	—	—	—	—	—	0.636
W3	0.687	0.674	0.674	0.671	**0.692**	**0.692**	0.510	**0.692**	—	—	—	0.664
Social												
OI	**0.671**	0.667	0.667	0.664	0.656	0.656	0.472	0.656	—	—	—	0.668
Ff	0.675	0.675	**0.679**	**0.679**	0.663	0.664	0.435	0.664	0.644	0.644	0.644	0.675
FL	—	0.630	0.621	0.584	—	—	—	—	—	—	—	**0.631**
SZ	0.356	0.384	**0.385**	0.380	0.371	0.375	0.199	0.376	0.362	0.361	0.361	0.382
Ws	0.525	0.612	0.638	0.643	0.672	**0.691**	0.016	**0.691**	—	—	—	0.603
YT	—	0.544	0.624	**0.628**	—	—	—	—	—	—	—	0.536
Trust												
AD	**0.929**	0.903	0.927	0.916	0.806	0.857	0.686	0.853	0.677	0.681	0.681	0.895
EL	0.800	0.798	**0.816**	0.813	0.791	0.782	0.768	0.785	0.792	0.793	0.793	0.799
EP	0.287	0.281	**0.295**	0.293	0.286	0.294	0.200	0.291	0.280	0.280	0.279	0.283

Table B.2: Evaluation results for all square datasets (directed methods from Section 6.3).

	asym						back					
Dat.	A-POLY-O	A-POLYN-O	A-SINH	A-NEU-O	A-RR	X-SQU	A-POLY-O	A-POLYN-O	A-SINH	A-NEU-O	A-RR	X-SQU
Authorship												
AP	0.687	0.687	0.687	0.687	0.499	0.687	**0.690**	**0.690**	**0.690**	**0.690**	0.499	0.689
PH	0.727	**0.730**	**0.730**	**0.730**	0.606	0.725	0.727	**0.730**	**0.730**	**0.730**	0.606	0.725
TH	0.719	**0.720**	**0.720**	**0.720**	0.600	0.715	0.719	0.719	**0.720**	0.719	0.600	0.715
Pc	**0.758**	**0.758**	**0.758**	**0.758**	0.619	0.754	0.757	0.757	0.757	0.757	0.619	0.753
Communication												
DG	0.784	0.784	0.784	0.784	0.762	0.802	0.787	0.786	0.786	0.786	0.762	**0.813**
WK	**0.667**	0.663	0.663	0.663	0.500	0.663	0.654	0.649	0.649	0.649	0.500	0.666
EN	0.761	0.762	0.763	0.763	0.773	0.758	**0.776**	0.774	0.775	0.775	0.773	0.737
EU	0.633	0.636	0.636	0.636	0.498	**0.671**	0.622	0.623	0.623	0.623	0.498	0.639
Ow	**0.787**	**0.787**	**0.787**	**0.787**	0.673	0.784	0.781	0.781	0.781	0.781	0.673	0.761
PG	0.898	0.897	0.897	0.897	0.760	0.909	**0.946**	**0.946**	**0.946**	0.945	0.760	0.925
Wa	0.755	0.755	0.755	0.755	0.621	0.750	**0.765**	**0.765**	**0.765**	**0.765**	0.621	0.760
A@	0.811	**0.826**	0.825	0.819	0.659	0.820	0.813	0.814	0.814	0.813	0.659	0.717
Co-occurrence												
AM	0.663	0.663	0.663	0.663	0.500	0.664	**0.668**	**0.668**	**0.668**	**0.668**	0.500	0.667
SI	0.881	0.881	0.867	0.867	0.750	**0.918**	0.857	0.857	0.857	0.857	0.750	0.830
Interaction												
CO	0.720	0.719	0.720	0.719	0.534	0.722	**0.730**	0.728	0.728	0.728	0.534	0.721
PM	**0.888**	0.872	0.872	0.873	0.629	0.883	0.842	0.841	0.841	0.841	0.629	0.719
RM	0.450	0.487	0.473	**0.498**	0.189	0.425	0.233	0.233	0.233	0.233	0.189	0.213
Physical												
IN	**0.839**	0.838	0.835	0.836	0.709	0.838	0.801	0.797	0.788	0.788	0.709	0.825
RO	**0.542**	**0.542**	**0.542**	**0.542**	0.500	**0.542**	**0.542**	**0.542**	**0.542**	**0.542**	0.500	**0.542**
GN	0.576	0.575	0.575	0.575	0.500	**0.587**	0.548	0.548	0.548	0.548	0.500	0.559
TO	0.661	0.659	0.659	0.659	0.560	**0.683**	0.623	0.619	0.619	0.618	0.560	0.510
AS	0.776	0.777	0.777	0.776	0.766	0.756	**0.880**	**0.880**	**0.880**	**0.880**	0.766	0.852
SK	**0.678**	0.675	0.675	0.674	0.500	0.674	0.625	0.620	0.619	0.619	0.500	0.583
Ratings												
LI	**0.639**	**0.639**	**0.639**	**0.639**	0.479	**0.639**	0.429	0.430	0.430	0.430	0.479	0.427
Reference												
th	0.673	0.672	0.672	0.672	0.501	**0.678**	0.670	0.670	0.670	0.670	0.501	0.673
BS	**0.687**	0.686	0.686	0.685	0.500	0.563	0.664	0.659	0.659	0.658	0.500	0.558
CS	0.678	0.678	0.678	0.678	0.501	**0.680**	0.674	0.674	0.674	0.674	0.501	0.675
Pi	0.947	0.946	0.946	0.946	0.779	**0.952**	0.909	0.908	0.908	0.908	0.779	0.903
WP	0.616	0.614	0.614	0.614	0.500	0.592	**0.619**	**0.619**	**0.619**	**0.619**	0.500	0.603
GC	**0.685**	0.680	0.680	0.679	0.500	0.679	0.595	0.586	0.586	0.585	0.500	0.508
GO	**0.679**	**0.679**	**0.679**	**0.679**	0.500	0.678	0.672	0.672	0.672	0.672	0.500	0.670
WT	**0.636**	**0.636**	**0.636**	**0.636**	0.500	**0.636**	0.619	0.619	0.619	0.619	0.500	0.619
PC	0.664	0.664	0.664	0.664	0.500	**0.665**	0.652	0.652	0.652	0.652	0.500	0.652
W3	0.681	0.681	0.681	0.682	0.500	0.680	**0.683**	**0.683**	**0.683**	**0.683**	0.500	**0.683**
Social												
OI	0.657	0.657	0.657	0.657	0.501	0.658	0.662	0.662	0.662	0.662	0.501	**0.664**
Ff	0.631	0.633	0.633	0.633	0.504	0.629	**0.667**	**0.667**	**0.667**	**0.667**	0.504	**0.667**
FL	0.613	0.612	0.612	0.612	0.500	**0.630**	0.616	0.615	0.615	0.615	0.500	0.617
SZ	0.369	0.369	0.369	0.369	0.241	**0.377**	0.338	0.338	0.338	0.338	0.241	0.334
Ws	0.564	0.560	0.559	0.559	0.501	**0.590**	0.451	0.448	0.448	0.448	0.501	0.476
YT	0.541	0.541	0.535	0.535	0.500	**0.568**	0.468	0.470	0.460	0.460	0.500	0.501
Trust												
AD	0.849	0.848	0.848	0.847	0.669	0.862	0.885	0.885	0.885	0.885	0.669	**0.886**
EL	0.788	0.788	0.788	0.788	0.790	0.801	**0.807**	**0.807**	**0.807**	**0.807**	0.790	0.775
EP	0.270	0.270	0.270	0.270	0.193	**0.308**	0.265	0.265	0.265	0.265	0.193	0.232

Table B.3: Evaluation results for all bipartite datasets.

Dat.	P-PA	A-POLY-O	A-POLYN-O	A-SINH	A-NEU-O	A-RR	N-COM-O	N-NEU-O	L-REG	L-HEAT	X-SQU
Authorship											
ar	**0.780**	0.683	0.692	0.692	0.692	0.621	0.776	0.776	0.491	0.490	0.645
eu	**0.779**	0.646	0.636	0.637	0.638	0.620	0.767	0.763	0.403	0.403	0.565
bn	**0.780**	0.608	0.660	0.661	0.661	0.622	0.677	0.669	0.669	0.668	0.708
br	**0.777**	0.747	0.719	0.719	0.718	0.616	0.757	0.754	0.412	0.413	0.673
ca	**0.769**	0.698	0.703	0.703	0.704	0.607	0.768	0.767	0.352	0.351	0.709
zh	**0.806**	0.735	0.733	0.733	0.733	0.660	0.802	0.802	0.421	0.421	0.733
Pa	0.665	0.656	0.656	0.656	0.656	0.499	**0.679**	**0.679**	—	—	0.642
nl	**0.773**	0.707	0.703	0.704	0.703	0.610	—	—	—	—	0.717
ben	**0.863**	0.703	0.710	0.712	0.714	0.756	0.847	0.845	0.698	0.698	0.730
nen	0.843	0.797	0.790	0.791	0.790	0.513	**0.857**	0.845	0.451	0.451	0.774
qen	**0.838**	0.627	0.680	0.683	0.683	0.709	0.738	0.726	0.610	0.610	0.543
men	**0.759**	0.605	0.646	0.646	0.646	0.593	0.662	0.649	0.497	0.497	0.550
eo	**0.763**	0.628	0.611	0.613	0.613	0.597	0.734	0.733	0.463	0.466	0.638
bfr	0.741	0.721	0.721	0.725	0.724	0.457	**0.807**	0.802	0.524	0.525	0.703
nfr	0.608	0.645	0.653	0.656	0.656	0.334	**0.730**	**0.730**	0.589	0.589	0.576
mfr	**0.754**	0.735	0.735	0.735	0.735	0.586	0.729	0.724	0.692	0.692	0.658
gl	**0.777**	0.655	0.668	0.668	0.668	0.617	0.769	0.769	0.451	0.451	0.656
mde	**0.736**	0.597	0.608	0.607	0.605	0.558	0.703	0.701	0.492	0.492	0.423
GH	**0.690**	0.603	0.599	0.600	0.599	0.501	0.688	0.688	0.413	0.413	0.638
el	**0.786**	0.691	0.700	0.700	0.700	0.629	0.785	0.785	0.408	0.407	0.675
ht	**0.773**	0.705	**0.773**	**0.773**	**0.773**	0.612	0.574	0.572	0.736	0.736	0.620
it	—	0.758	**0.759**	**0.759**	**0.759**	0.629	—	—	—	—	0.711
ja	—	0.753	**0.754**	**0.754**	**0.754**	0.619	—	—	—	—	0.736
lv	**0.780**	0.698	0.712	0.712	0.711	0.619	0.762	0.758	0.317	0.317	0.638
nds	**0.790**	0.738	0.760	0.776	0.777	0.636	**0.790**	0.789	0.585	0.586	0.723
ST	0.663	0.643	0.643	0.643	0.643	0.502	0.681	**0.682**	0.631	0.631	0.650
oc	**0.787**	0.784	0.783	0.783	0.783	0.630	0.606	0.597	0.547	0.546	0.625
pl	—	**0.738**	0.733	0.733	0.733	0.608	—	—	—	—	0.699
pt	**0.771**	0.680	0.676	0.677	0.677	0.606	—	—	—	—	0.675
ru	**0.798**	0.740	0.742	0.742	0.742	0.647	—	—	—	—	0.751
sr	**0.783**	0.705	0.705	0.703	0.703	0.625	**0.783**	**0.783**	0.436	0.437	0.709
sk	**0.767**	0.665	0.614	0.614	0.614	0.602	0.760	0.760	0.386	0.384	0.608
es	**0.801**	0.768	0.757	0.757	0.757	0.654	—	—	—	—	0.759
sv	**0.764**	0.687	0.689	0.690	0.690	0.596	—	—	—	—	0.693
vi	**0.766**	0.690	0.695	0.695	0.695	0.599	0.765	0.764	0.589	0.589	0.693
cy	**0.780**	0.769	0.759	0.758	0.758	0.623	0.755	0.749	0.496	0.496	0.686
Communication											
Fc	**0.692**	0.604	0.603	0.603	0.603	0.505	0.600	0.598	—	—	0.644
Features											
CN	0.956	0.910	0.966	0.980	0.980	0.759	**0.984**	0.979	0.965	0.965	0.836
EX	**0.690**	0.646	0.645	0.645	0.645	0.500	0.476	0.476	—	—	0.661
WC	0.675	0.649	0.651	0.651	0.651	0.500	**0.688**	**0.688**	—	—	0.658
FG	—	0.671	0.669	0.669	0.669	0.500	—	—	—	—	**0.678**
GE	**0.688**	0.547	0.552	0.555	0.555	0.500	0.685	0.685	0.624	0.624	0.524
PS	**0.949**	0.902	0.913	0.919	0.919	0.731	0.760	0.600	0.847	0.848	0.909
PW	0.697	0.732	0.736	0.735	0.735	0.597	**0.838**	0.788	0.792	0.793	0.809
RL	**0.962**	0.887	0.887	0.887	0.884	0.753	0.945	0.939	0.846	0.846	0.907
R2	**0.690**	0.655	0.656	0.658	0.658	0.501	0.575	0.566	—	—	0.676
RE	—	0.670	0.653	0.653	0.652	0.500	—	—	—	—	**0.679**
TM	0.693	0.679	0.682	0.679	0.680	0.510	0.697	**0.700**	0.681	0.681	0.691
YG	**0.687**	0.598	0.607	0.607	0.607	0.500	0.661	0.660	0.551	0.551	0.629
Folksonomy											
B_{ti}	0.923	0.950	0.950	0.950	0.950	0.758	0.923	0.923	—	—	**0.956**
B_{ui}	**0.862**	0.776	0.776	0.776	0.776	0.641	0.852	0.853	—	—	0.771
B_{ut}	**0.878**	0.819	0.819	0.820	0.819	0.778	**0.878**	**0.878**	—	—	0.791
C_{ti}	**0.698**	0.601	0.605	0.605	0.605	0.509	**0.698**	**0.698**	0.359	0.359	0.606
C_{ui}	0.824	**0.846**	**0.846**	**0.846**	**0.846**	0.720	0.837	0.835	0.705	0.705	0.835
C_{ut}	**0.882**	0.832	0.833	0.833	0.833	0.783	**0.882**	**0.882**	0.611	0.611	0.801
M_{ti}	0.868	0.865	0.865	0.865	0.865	0.742	**0.874**	0.872	0.786	0.786	0.864
M_{ui}	**0.772**	0.711	0.708	0.708	0.708	0.657	0.765	0.761	0.676	0.677	0.726
M_{ut}	**0.899**	0.801	0.803	0.803	0.803	0.656	0.853	0.837	0.627	0.626	0.815
W_{ti}	0.953	0.919	0.919	0.919	0.919	0.750	0.953	**0.954**	—	—	0.902
W_{ui}	—	**0.754**	**0.754**	**0.754**	**0.754**	0.690	—	—	—	—	0.753
W_{ut}	—	**0.786**	0.784	0.784	0.785	0.688	—	—	—	—	0.784
V_{ti}	**0.714**	0.650	0.630	0.630	0.629	0.528	0.713	0.713	0.319	0.319	0.654
V_{ui}	**0.758**	0.754	0.753	0.753	0.753	0.595	**0.758**	**0.758**	0.554	0.554	0.753
V_{ut}	**0.877**	0.806	0.806	0.806	0.806	0.773	0.876	0.875	0.491	0.491	0.807
Interaction											
Lb	—	0.941	**0.942**	**0.942**	**0.942**	0.929	—	—	—	—	0.939
Ls	—	0.843	**0.844**	**0.844**	**0.844**	0.747	—	—	—	—	0.839
Ratings											
BX	0.000	0.261	0.261	0.261	0.261	**0.275**	0.267	0.267	—	—	0.259
ER	—	**0.457**	**0.457**	**0.457**	**0.457**	0.312	—	—	—	—	0.456
Fr	—	0.652	0.652	0.652	0.652	0.544	—	—	—	—	**0.655**
JE	0.501	0.574	0.571	0.581	0.580	0.458	**0.588**	**0.588**	0.547	0.494	0.579
M1	0.000	0.626	0.626	0.626	0.626	0.616	0.642	0.639	0.641	0.657	**0.662**
M3	0.000	0.608	0.608	0.608	0.608	0.491	0.612	0.612	—	—	**0.614**
M2	0.000	0.688	0.688	0.688	0.688	0.608	**0.708**	**0.708**	0.635	0.648	0.707

Table B.4 gives the best performing graph kernel for each dataset. This is an extended version of Table 4.5.

Table B.4: Best fitting curve by dataset.

Dataset	Best transformation	Best method	MAP
Advogato (AD)	$\mathbf{A}_a + \mathbf{A}_a^T \to \mathbf{A}_b + \mathbf{A}_b^T$	A-EXP	0.932
Amazon (AM)	$\mathbf{A}_a + \mathbf{A}_a^T \to \mathbf{A}_a + \mathbf{A}_a^T + \mathbf{A}_b + \mathbf{A}_b^T$	A-EXP	0.673
U. Rovira i Virgili (A@)	$\mathbf{A}_a + \mathbf{A}_a^T \to \mathbf{A}_b + \mathbf{A}_b^T$	A-POLYN	0.972
Caenorhabditis elegans (PM)	$\mathbf{N}_a + \mathbf{N}_a^T \to \mathbf{A}_b + \mathbf{A}_b^T$	A-TRI	0.940
Pretty Good Privacy (PG)	$\mathbf{L}_a \to \mathbf{A}_b$	L-REG	0.990
Route Views (AS)	$\mathbf{L}_a \to \mathbf{A}_a + \mathbf{A}_b$	L-COM	0.954
CAIDA (IN)	$\mathbf{N}_a + \mathbf{N}_a^T \to \mathbf{A}_b + \mathbf{A}_b^T$	N-COM-O	0.985
Skitter (SK)	$\mathbf{A}_a + \mathbf{A}_a^T \to \mathbf{A}_b + \mathbf{A}_b^T$	A-POLY	0.685
BibSonomy (B_{ti})	$\mathbf{B}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	A-POLY-O	0.962
BibSonomy (B_{ui})	$\mathbf{M}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	N-POLY-O	0.854
BibSonomy (B_{ut})	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-POLY-O	0.878
BookCrossing (BX)	$\mathbf{M}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	N-POLYN-O	0.278
arXiv astro-ph (AP)	$\mathbf{A}_a + \mathbf{A}_a^T \to \mathbf{A}_b + \mathbf{A}_b^T$	A-POLY	0.690
Google.com internal (GC)	$\mathbf{A}_a + \mathbf{A}_a^T \to \mathbf{A}_b + \mathbf{A}_b^T$	A-POLY	0.687
CiteSeer (CS)	$\mathbf{A}_a + \mathbf{A}_a^T \to \mathbf{A}_a + \mathbf{A}_a^T + \mathbf{A}_b + \mathbf{A}_b^T$	A-EXP	0.684
CiteULike (C_{ti})	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.698
CiteULike (C_{ui})	$\mathbf{L}_a \to \mathbf{A}_a + \mathbf{A}_b$	L-HEAT	0.873
CiteULike (C_{ut})	$\mathbf{M}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.882
arXiv hep-ph (PH)	$\mathbf{N}_a + \alpha\mathbf{N}_a^T \to \mathbf{A}_b$	N-COM-O	0.758
arXiv hep-th (TH)	$\mathbf{N}_a + \mathbf{N}_a^T \to \mathbf{A}_a + \mathbf{A}_a^T + \mathbf{A}_b + \mathbf{A}_b^T$	N-COM-O	0.752
Haggle (CO)	$\mathbf{A}_a + \mathbf{A}_a^T \to \mathbf{A}_b + \mathbf{A}_b^T$	A-NEU	0.738
DBLP (Pa)	$\mathbf{M}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	A-SINH	0.679
DBLP (Pi)	$\mathbf{A}_a + \mathbf{A}_a^T \to \mathbf{A}_b + \mathbf{A}_b^T$	A-POLY	0.966
DBLP (Pc)	$\mathbf{A}_a + \mathbf{A}_a^T \to \mathbf{A}_a + \mathbf{A}_a^T + \mathbf{A}_b + \mathbf{A}_b^T$	A-EXP	0.759
Countries (CN)	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.984
Genre (GE)	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	A-SINH	0.686
Record labels (RL)	$\mathbf{B}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.963
Similarity (SI)	$\mathbf{L}_a \to \mathbf{A}_a + \mathbf{A}_b$	L-POLY	0.995
Movies (ST)	$\mathbf{M}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	A-SINH	0.682
Teams (TM)	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-NEU-O	0.700
Arabic Wikipedia (ar)	$\mathbf{M}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	N-POLYN-O	0.778
Bengali Wikipedia (bn)	$\mathbf{L}_a \to \mathbf{A}_a + \mathbf{A}_b$	L-REG	0.774
Breton Wikipedia (br)	$\mathbf{L}_a \to \mathbf{A}_a + \mathbf{A}_b$	L-COM	0.772
Catalan Wikipedia (ca)	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.768
Welsh Wikipedia (cy)	$\mathbf{B}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	A-POLY-O	0.779
German Wiktionary (mde)	$\mathbf{B}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.708
Greek Wikipedia (el)	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.785
English Wikibooks (ben)	$\mathbf{M}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	N-POLY-O	0.851
English Wikinews (nen)	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-POLY-O	0.869
English Wikiquote (qen)	$\mathbf{L}_a \to \mathbf{A}_b$	A-TRI	0.815
English Wiktionary (men)	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-POLY-O	0.718
Esperanto Wikipedia (eo)	$\mathbf{L}_a \to \mathbf{A}_a + \mathbf{A}_b$	L-HEAT	0.743
Spanish Wikipedia (es)	$\mathbf{B}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-ABS	0.784
Basque Wikipedia (eu)	$\mathbf{M}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	N-NEU	0.771
French Wikibooks (bfr)	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.807
French Wikinews (nfr)	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	A-SINH	0.738
French Wiktionary (mfr)	$\mathbf{B}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.755
Galician Wikipedia (gl)	$\mathbf{M}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	A-SINH	0.770
Haitian Wikipedia (ht)	$\mathbf{B}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.774
Italian Wikipedia (it)	$\mathbf{B}_a \to \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.760
Japanese Wikipedia (ja)	$\mathbf{B}_a \to \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-ABS	0.764

Continued on next page

Dataset	Best transformation	Best method	MAP
Latvian Wikipedia (lv)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.773
Low German Wikipedia (nds)	$\mathbf{M}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.790
Dutch Wikipedia (nl)	$\mathbf{B}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-ABS	0.741
Occitan Wikipedia (oc)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.787
Polish Wikipedia (pl)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-POLY-O	0.741
Portuguese Wikipedia (pt)	$\mathbf{B}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-ABS	0.732
Russian Wikipedia (ru)	$\mathbf{B}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-ABS	0.773
Slovak Wikipedia (sk)	$\mathbf{M}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.760
Serbian Wikipedia (sr)	$\mathbf{M}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-POLY-O	0.783
Swedish Wikipedia (sv)	$\mathbf{B}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-ABS	0.731
Vietnamese Wikipedia (vi)	$\mathbf{M}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.765
Chinese Wikipedia (zh)	$\mathbf{M}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-POLY-O	0.802
English Wikipedia (EL)	$\mathbf{A}_a + \mathbf{A}_a^T \rightarrow \mathbf{A}_b + \mathbf{A}_b^T$	A-EXP	0.828
EU institution (EU)	$\mathbf{N}_a + \mathbf{N}_a^T \rightarrow \mathbf{A}_a + \mathbf{A}_a^T + \mathbf{A}_b + \mathbf{A}_b^T$	N-COM-O	0.693
Enron (EN)	$\mathbf{N}_a + \alpha\mathbf{N}_a^T \rightarrow \mathbf{A}_a + \mathbf{A}_b$	N-COM-O	0.876
Epinions (EP)	$\mathbf{A}_a \rightarrow \mathbf{A}_a + \mathbf{A}_b$	X-ABS	0.310
Epinions (ER)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.461
Facebook New Orleans (Ol)	$\mathbf{A}_a + \mathbf{A}_a^T \rightarrow \mathbf{A}_a + \mathbf{A}_a^T + \mathbf{A}_b + \mathbf{A}_b^T$	X-ABS	0.669
Facebook New Orleans (Ow)	$\mathbf{N}_a + \alpha\mathbf{N}_a^T \rightarrow \mathbf{A}_b$	N-POLYN-O	0.802
Filmtipset (Fc)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-POLY-O	0.681
Filmtipset (Ff)	$\mathbf{L}_a \rightarrow \mathbf{A}_b$	L-REG	0.685
Filmtipset (Fr)	$\mathbf{B}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-SQU	0.655
Flickr (FG)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-POLY-O	0.691
Flickr (FL)	$\mathbf{A}_a + \mathbf{A}_a^T \rightarrow \mathbf{A}_b + \mathbf{A}_b^T$	A-POLY	0.655
Github (GH)	$\mathbf{M}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.688
English Wikipedia (EX)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-SINH	0.690
Reuters-21578 (R2)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-SINH	0.689
arXiv hep-th (th)	$\mathbf{A}_a + \mathbf{A}_a^T \rightarrow \mathbf{A}_b + \mathbf{A}_b^T$	A-POLY	0.686
Jester (JE)	$\mathbf{M}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-SQU	0.590
Last.fm (Lb)	$\mathbf{B}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-ABS	0.953
Last.fm (Ls)	$\mathbf{B}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-ABS	0.851
Líbímseti.cz (LI)	$\mathbf{A}_a \rightarrow \mathbf{A}_b$	A-NEU-O	0.639
Reality Mining (RM)	$\mathbf{A}_a \rightarrow \mathbf{A}_a + \mathbf{A}_b$	A-NEU-O	0.498
MovieLens 100k (M1)	$\mathbf{M}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	N-POLY-O	0.717
MovieLens 10M (M3)	$\mathbf{B}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-SQU	0.614
MovieLens (M_{ti})	$\mathbf{M}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.874
MovieLens (M_{ui})	$\mathbf{M}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.765
MovieLens (M_{ut})	$\mathbf{M}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.853
MovieLens 1M (M2)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.724
Digg (DG)	$\mathbf{N}_a + \mathbf{N}_a^T \rightarrow \mathbf{A}_b + \mathbf{A}_b^T$	A-TRI	0.845
Twitter (Wa)	$\mathbf{A}_a + \mathbf{A}_a^T \rightarrow \mathbf{A}_b + \mathbf{A}_b^T$	A-EXP	0.771
Twitter (W_{ti})	$\mathbf{M}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-POLY-O	0.955
Twitter (W_{ui})	$\mathbf{B}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.754
Twitter (W_{ut})	$\mathbf{B}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	X-ABS	0.794
Twitter (Ws)	$\mathbf{N}_a + \mathbf{N}_a^T \rightarrow \mathbf{A}_a + \mathbf{A}_a^T + \mathbf{A}_b + \mathbf{A}_b^T$	N-COM	0.691
Gnutella (GN)	$\mathbf{A}_a + \mathbf{A}_a^T \rightarrow \mathbf{A}_b + \mathbf{A}_b^T$	A-NEU	0.654
US patents (PC)	$\mathbf{A}_a \rightarrow \mathbf{A}_a + \mathbf{A}_b$	X-ABS	0.665
vi.sualize.us (V_{ti})	$\mathbf{M}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	N-NEU	0.714
vi.sualize.us (V_{ui})	$\mathbf{M}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.758
vi.sualize.us (V_{ut})	$\mathbf{M}_a \rightarrow \mathbf{B}_a + \mathbf{B}_a^T + \mathbf{B}_b + \mathbf{B}_b^T$	N-COM-O	0.876
Prosper.com (PS)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.944
Prosper.com (PW)	$\mathbf{L}_a \rightarrow \mathbf{A}_b$	L-HEAT	0.842
Reuters (RE)	$\mathbf{B}_a \rightarrow \mathbf{B}_b + \mathbf{B}_b^T$	A-POLYN-O	0.691
California (RO)	$\mathbf{A}_a + \mathbf{A}_a^T \rightarrow \mathbf{A}_b + \mathbf{A}_b^T$	A-TRI	0.615
Slashdot Zoo (SZ)	$\mathbf{A}_a + \mathbf{A}_a^T \rightarrow \mathbf{A}_b + \mathbf{A}_b^T$	A-POLYN	0.397

Continued on next page

Dataset	**Best transformation**	**Best method**	**MAP**
Internet topology (TO)	$\mathbf{A}_{\mathrm{a}} + \mathbf{A}_{\mathrm{a}}^{\mathrm{T}} \to \mathbf{A}_{\mathrm{b}} + \mathbf{A}_{\mathrm{b}}^{\mathrm{T}}$	A-EXP	0.726
TREC WT10g (WT)	$\mathbf{A}_{\mathrm{a}} \to \mathbf{A}_{\mathrm{a}} + \mathbf{A}_{\mathrm{b}}$	A-POLYN-O	0.636
Berkeley/Stanford (BS)	$\mathbf{A}_{\mathrm{a}} + \mathbf{A}_{\mathrm{a}}^{\mathrm{T}} \to \mathbf{A}_{\mathrm{b}} + \mathbf{A}_{\mathrm{b}}^{\mathrm{T}}$	A-POLY	0.688
Google (GO)	$\mathbf{N}_{\mathrm{a}} + \alpha\mathbf{N}_{\mathrm{a}}^{\mathrm{T}} \to \mathbf{A}_{\mathrm{a}} + \mathbf{A}_{\mathrm{b}}$	N-NEU-O	0.685
English Wikipedia (WC)	$\mathbf{M}_{\mathrm{a}} \to \mathbf{B}_{\mathrm{a}} + \mathbf{B}_{\mathrm{a}}^{\mathrm{T}} + \mathbf{B}_{\mathrm{b}} + \mathbf{B}_{\mathrm{b}}^{\mathrm{T}}$	N-NEU-O	0.688
English Wikipedia (WP)	$\mathbf{N}_{\mathrm{a}} + \mathbf{N}_{\mathrm{a}}^{\mathrm{T}} \to \mathbf{A}_{\mathrm{b}} + \mathbf{A}_{\mathrm{b}}^{\mathrm{T}}$	N-EXP	0.664
English Wikipedia (WK)	$\mathbf{A}_{\mathrm{a}} + \mathbf{A}_{\mathrm{a}}^{\mathrm{T}} \to \mathbf{A}_{\mathrm{b}} + \mathbf{A}_{\mathrm{b}}^{\mathrm{T}}$	A-NEU-O	0.693
World Wide Web (W3)	$\mathbf{N}_{\mathrm{a}} + \alpha\mathbf{N}_{\mathrm{a}}^{\mathrm{T}} \to \mathbf{A}_{\mathrm{b}}$	N-NEU-O	0.692
YouTube (YG)	$\mathbf{B}_{\mathrm{a}} \to \mathbf{B}_{\mathrm{a}} + \mathbf{B}_{\mathrm{a}}^{\mathrm{T}} + \mathbf{B}_{\mathrm{b}} + \mathbf{B}_{\mathrm{b}}^{\mathrm{T}}$	X-ABS	0.667
YouTube (YT)	$\mathbf{A}_{\mathrm{a}} + \mathbf{A}_{\mathrm{a}}^{\mathrm{T}} \to \mathbf{A}_{\mathrm{b}} + \mathbf{A}_{\mathrm{b}}^{\mathrm{T}}$	A-EXP	0.687

Appendix C

Decomposing Large, Sparse Matrices

To compute spectral transformations, the eigenvalue and singular decompositions of large, sparse matrices have to be computed. This section briefly reviews how to achieve this using GNU Octave[1]. The methods presented here work equally well in other programming languages using a similar syntax.

When performing a decomposition, there is a crucial difference between the adjacency and Laplacian matrix: We are interested in the largest eigenvalue of the adjacency matrix, but in the smallest eigenvalues of the Laplacian matrix. This difference leads to two distinct methods for decomposition. For the normalized adjacency matrix, both methods can be used, since it is equivalent to the normalized Laplacian up to a spectral transformation.

Adjacency Matrix The sparse symmetric n by n adjacency matrix $\mathbf{A}$ can be decomposed as $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$. The number of nonzero entries in $\mathbf{A}$ equals the number of edges in the network, and is usually on the order of $O(n)$. Since the full matrix $\mathbf{U}$ of size $n \times n$ cannot possibly be represented in memory, only the r eigenpairs with highest eigenvalue are computed, for a given r. The number r is called the *rank* of the decomposition.

Rank-reduced decompositions of large, sparse matrices can be computed in GNU Octave using the function `eigs`. By default, the largest eigenvalues and corresponding eigenvectors are found. This is what we want when decomposing $\mathbf{A}$:

```
[U, Lambda] = eigs(A, r);
```

Laplacian Matrix The Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is symmetric and positive-semidefinite. It has eigenvalue zero if one of the graph's connected components is balanced. We are interested in its eigenvectors of smallest nonzero eigenvalue.

The function `eigs()` has a mode for finding the eigenvalues nearest to a given value. However, this value must not be an eigenvalue itself. If the Laplacian is known to not have zero as eigenvalue, we can tell GNU Octave to find the eigenvalues nearest to zero:

```
[U, Lambda] = eigs(L, r, 0);
```

If zero is an eigenvalue, we must pass a small negative value ε:

```
[U, Lambda] = eigs(L, r, -1e-3);
```

[1] www.gnu.org/software/octave

Normalized Adjacency Matrix The normalized adjacency matrix $\mathbf{N} = \mathbf{D}^{1/2}\mathbf{A}\mathbf{D}^{1/2}$ can be decomposed in two ways: By the *adjacency* method or by the *Laplacian* method. The eigenvalues of $\mathbf{N}$ are contained in the interval $[-1, +1]$. The adjacency method is simply:

```
[U, D] = eigs(N, r);
```

The Laplacian method works in two steps. First we compute the eigenvalues nearest to $+1 + \varepsilon$ and then the eigenvalues nearest to $-1 - \varepsilon$:

```
[U1, D1] = eigs(N, r/2, +1+1e-3);
[U2, D2] = eigs(N, r/2, -1-1e-3);
```

The Laplacian method is faster but takes more memory. The normalized Laplacian $\mathbf{Z} = \mathbf{I} - \mathbf{N}$ can be decomposed analogously.

Bibliography

[AA01] Lada Adamic and Eytan Adar. Friends and neighbors on the Web. *Social Networks*, 25:211–230, 2001.

[ABK+08] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *Proc. Int. Semantic Web Conf.*, pages 722–735, 2008.

[AJB99] Réka Albert, Hawoong Jeong, and Albert-László Barabási. The diameter of the World Wide Web. *Nature*, 401:130, 1999.

[BA99] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[BBPSV04] Alain Barrat, Marc Barthélemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. The architecture of complex weighted networks. *Proc. Natl. Acad. Sci. USA*, 101(11):3747–3752, 2004.

[BCH03] Peter Bailey, Nick Craswell, and David Hawking. Engineering a multi-purpose test collection for Web retrieval experiments. *Information Processing and Management*, 39(6):853–871, 2003.

[BFK01] Zhiqiang Bi, Christos Faloutsos, and Flip Korn. The 'DGX' distribution for mining massive, skewed data. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, pages 17–26, 2001.

[BFL06] Ulrik Brandes, Daniel Fleischer, and Jürgen Lerner. Summarizing dynamic bipolar conflict structures. *Trans. on Visualization and Computer Graphics*, 12(6):1486–1499, 2006.

[BHK98] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[BKLR09] Ulrik Brandes, Patrick Kenis, Jürgen Lerner, and Denise van Raaij. Network analysis of collaboration structure in Wikipedia. In *Proc. Int. World Wide Web Conf.*, pages 731–740, 2009.

[BL04] Phillip Bonacich and Paulette Lloyd. Calculating status with negative relations. *Social Networks*, (26):331–338, 2004.

[BL07] James Bennett and Stan Lanning. The Netflix Prize. In *Proc. KDD Cup*, pages 3–6, 2007.

[BLG98] Kurt Bollacker, Steve Lawrence, and C. Lee Giles. CiteSeer: An autonomous Web agent for automatic retrieval and identification of interesting publications. In *Proc. Int. Conf. on Autonomous Agents*, pages 116–123, 1998.

[BN02] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, pages 585–591, 2002.

[BNJL03] David Blei, Andrew Ng, Michael Jordan, and John Lafferty. Latent Dirichlet allocation. *J. Machine Learning Research*, 3:993–1022, 2003.

[Bol98] Béla Bollobás. *Modern Graph Theory*. Springer, 1998.

[Bon87] Phillip Bonacich. Power and centrality: A family of measures. *American J. of Sociology*, 92(5):1170–1182, 1987.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[BP07] Lukáš Brožovský and Václav Petříček. Recommender system for online dating service. In *Proc. Znalosti*, pages 29–40, 2007.

[BPSDGA04] Marián Boguñá, Romualdo Pastor-Satorras, Albert Díaz-Guilera, and Alex Arenas. Models of social networks based on social distance attachment. *Phys. Rev. E*, 70(5):056122, 2004.

[Bra97] Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.

[BRZ11] Daniel Boley, Gyan Ranjan, and Zhi-Li Zhang. Commute times for a directed graph using an asymmetric Laplacian. *Linear Algebra and Applications*, 435:224–242, 2011.

[Cel10] Òscar Celma. Music recommendation datasets for research. `http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html`, May 2010. Version 1.0.

[Cha09] Scott Chacon. The 2009 GitHub contest. `http://github.com/blog/466-the-2009-github-contest`, July 2009.

[CHC+07] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Trans. on Mobile Computing*, 6(6):606–620, 2007.

[Chu97] Fan Chung. *Spectral Graph Theory*. American Math. Society, 1997.

[Chu05] Fan Chung. Laplacians and the Cheeger inequality for directed graphs. *Ann. of Combinatorics*, 9(1):1–19, 2005.

[CJ93] Rama Chellappa and Anil Jain. *Markov Random Fields: Theory and Application*. Academic Press, 1993.

[Cli73] William K. Clifford. Preliminary sketch of quaternions. *Proc. London Math. Soc.*, 4(1):381–395, 1873.

[CLS+10] Munmun De Choudhury, Yu-Ru Lin, Hari Sundaram, K. Selçuk Candan, Lexing Xie, and Aisling Kelliher. How does the data sampling strategy impact the discovery of information diffusion in social media? In *Proc. Conf. on Weblogs and Social Media*, pages 34–41, 2010.

[CRS97] Dragoš Cvetković, Peter Rowlinson, and Slobodan Simić. *Eigenspaces of Graphs*. Cambridge University Press, 1997.

[CS97] Pavel Chebotarev and Elena Shamis. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, 58(9):1505–1514, 1997.

[CS98] Pavel Chebotarev and Elena Shamis. On proximity measures for graph vertices. *Automation and Remote Control*, 59(10):1443–1459, 1998.

[CSJS09] Munmun De Choudhury, Hari Sundaram, Ajita John, and Dorée Duncan Seligmann. Social synchrony: Predicting mimicry of user actions in online social media. In *Proc. Int. Conf. on Computational Science and Engineering*, pages 151–158, 2009.

[CSN09] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[CWS03] Olivier Chapelle, Jason Weston, and Berhard Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 585–592, 2003.

[DA05] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Phys. Rev. E*, 72(2):027104, 2005.

[Dav09] Ian Davidson. Knowledge driven dimension reduction for clustering. In *Proc. Int. Conf. on Research and Development in Information Retrieval*, pages 1034–1039, 2009.

[DGK04] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: Spectral clustering and normalized cuts. In *Proc. Int. Conf. Knowledge Discovery and Data Mining*, pages 551–556, 2004.

[DM58] Andrew Dulmage and Nathan Mendelsohn. Coverings of bipartite graphs. *Canadian J. Math.*, 10:517–534, 1958.

[DM96] Patrick Doreian and Andrej Mrvar. A partitioning approach to structural balance. *Social Networks*, 18:149–168, 1996.

[DR94] Madhav Desai and Vasant Rao. A characterization of the smallest eigenvalue of a graph. *Graph Theory*, 18(2):181–194, 1994.

[DS84] Peter G. Doyle and J. Laurie Snell. *Random Walks and Electric Networks*. Math. Ass. of America, 1984.

[EC07] Kevin Emamy and Richard Cameron. CiteULike: A researcher's social bookmarking service. *Ariadne*, 51, 2007.

[ER59] Paul Erdős and Alfréd Rényi. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[ERV05] Ernesto Estrada and Juan A. Rodríguez-Velázquez. Spectral measures of bipartivity in complex networks. *Phys. Rev. E*, 72(4):046105, 2005.

[ESP06] Nathan Eagle and Alex (Sandy) Pentland. Reality Mining: Sensing complex social systems. *Personal Ubiquitous Computing*, 10(4):255–268, 2006.

[Fie73] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23(98):298–305, 1973.

[FPRS04] François Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. A novel way of computing dissimilarities between nodes of a graph, with application to collaborative filtering and subspace projection of the graph nodes. In *Proc. European Conf. on Machine Learning*, pages 26–37, 2004.

[FYPS06] François Fouss, Luh Yen, Alain Pirotte, and Marco Saerens. An experimental investigation of graph kernels on a collaborative recommendation task. In *Proc. Int. Conf. on Data Mining*, pages 863–868, 2006.

[GDDG+03] Roger Guimerà, Leon Danon, Albert Díaz-Guilera, Francesc Giralt, and Alex Arenas. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68(6):065103, 2003.

[GGK03] Johannes Gehrke, Paul Ginsparg, and Jon Kleinberg. Overview of the 2003 KDD Cup. *SIGKDD Explorations*, 5(2):149–151, 2003.

[GHKZ11] K. A. Germina, Shahul Hameed K., and Thomas Zaslavsky. On products and line graphs of signed graphs, their eigenvalues and energy. *Linear Algebra and Its Applications*, In Press, Corrected Proof, 2011.

[GKK03] Anna Goldenberg, Jeremy Kubica, and Paul Komarek. A comparison of statistical and machine learning algorithms on the task of link completion. In *Proc. Workshop on Link Analysis for Detecting Complex Behavior*, 2003.

[GKRT04] Ramanathan Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proc. Int. World Wide Web Conf.*, pages 403–412, 2004.

[GM08] Michel Gondran and Michel Minoux. *Graphs, Dioids and Semirings*. Springer, 2008.

[GMZ03] Christos Gkantsidis, Milena Mihail, and Ellen Zegura. Spectral analysis of Internet topologies. In *Proc. Joint Conf. IEEE Computer and Communications Societies*, pages 364–374, 2003.

[GRGP01] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[Gro06] GroupLens Research. MovieLens data sets. `http://www.grouplens.org/node/73`, October 2006.

[GSPA04] Roger Guimerà, Marta Sales-Pardo, and Luís A. Nunes Amaral. Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E*, 70(2):025101, 2004.

[Ham97] Richard W. Hamming. *The Art of Doing Science and Engineering: Learning to Learn*. Taylor & Francis, 1997.

[Har53] Frank Harary. On the notion of balance of a signed graph. *Michigan Math.*, 2(2):143–146, 1953.

[HCC04] Zan Huang, Wingyan Chung, and Hsinchun Chen. A graph model for e-commerce recommender systems. *American Society for Information Science and Technology*, 55(3):259–274, 2004.

[HH83] Per Hage and Frank Harary. *Structural Models in Anthropology*. Cambridge University Press, 1983.

[HJ94] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1994.

[HJD+04] Bo Hu, Xin-Yu Jiang, Jun-Feng Ding, Yan-Bo Xie, and Bing-Hong Wang. A model of weighted network: the student relationships in a class. *ArXiv Condensed Matter e-prints*, 2004.

[HJSS06] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. BibSonomy: A social bookmark and publication sharing system. In *Proc. Workshop on Conceptual Structure Tool Interoperability*, pages 87–102, 2006.

[HJT01] Bronwyn H. Hall, Adam B. Jaffe, and Manuel Trajtenberg. The NBER patent citations data file: Lessons, insights and methodological tools. In *NBER Working Papers 8498, National Bureau of Economic Research, Inc*, 2001.

[HLEK03] Petter Holme, Fredrik Liljeros, Christofer R. Edling, and Beom Jun Kim. On network bipartivity. *Phys. Rev. E*, 68(5):056107, 2003.

[HLP03] Yaoping P. Hou, Jiongsheng S. Li, and Yongliang Pan. On the Laplacian eigenvalues of signed graphs. *Linear and Multilinear Algebra*, 1(51):21–30, 2003.

[Hou05] Yaoping P. Hou. Bounds for the least Laplacian eigenvalue of a signed graph. *Acta Math. Sinica*, 21(4):955–960, 2005.

[Huc93] Joseph Hucks. Hyperbolic complex structures in physics. *J. Math. Phys.*, 34(12):5986–6008, 1993.

[HWSB08] Tad Hogg, Dennis M. Wilkinson, Gabor Szabo, and Michael J. Brzozowski. Multiple relationship types in online communities and social networks. In *Proc. AAAI Spring Symposium on Social Information Processing*, 2008.

[HZC07] Zan Huang, Daniel Zeng, and Hsinchun Chen. A comparison of collaborative-filtering recommendation algorithms for e-commerce. *IEEE Intelligent Systems*, 22(5):68–78, 2007.

[ISKM05] Takahiko Ito, Masashi Shimbo, Taku Kudo, and Yuji Matsumoto. Application of kernels to link analysis. In *Proc. Int. Conf. on Knowledge Discovery in Data Mining*, pages 586–592, 2005.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[KA07] Jérôme Kunegis and Sahin Albayrak. Adapting ratings in memory-based collaborative filtering using linear regression. In *Proc. Int. Conf. on Information Reuse and Integration*, 2007.

[Kat53] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[KCH02] Yehuda Koren, Liran Carmel, and David Harel. ACE: A fast multiscale eigenvectors computation for drawing huge graphs. In *Symposium on Information Visualization*, pages 137–144, 2002.

[KD08] Cristobald De Kerchove and Paul Van Dooren. The PageTrust algorithm: How to rank Web pages when negative links are allowed? In *Proc. SIAM Int. Conf. on Data Mining*, pages 346–352, 2008.

[KDLA10] Jérôme Kunegis, Ernesto W. De Luca, and Sahin Albayrak. The link prediction problem in bipartite networks. In *Proc. Int. Conf. in Information Processing and Management of Uncertainty in Knowledge-based Systems*, pages 380–389, 2010.

[KFB10] Jérôme Kunegis, Damien Fay, and Christian Bauckhage. Network growth and the spectral evolution model. In *Proc. Int. Conf. on Information and Knowledge Management*, pages 739–748, 2010.

[KL02] Risi Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proc. Int. Conf. on Machine Learning*, pages 315–322, 2002.

[KL09] Jérôme Kunegis and Andreas Lommatzsch. Learning spectral graph transformations for link prediction. In *Proc. Int. Conf. on Machine Learning*, pages 561–568, 2009.

[KLB08] Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. Alternative similarity functions for graph kernels. In *Proc. Int. Conf. on Pattern Recognition*, 2008.

[KLB09] Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. The Slashdot Zoo: Mining a social network with negative edges. In *Proc. Int. World Wide Web Conf.*, pages 741–750, 2009.

[KLBA08] Jérôme Kunegis, Andreas Lommatzsch, Christian Bauckhage, and Sahin Albayrak. On the scalability of graph kernels applied to collaborative recommenders. In *Proc. ECAI Workshop on Recommender Systems*, pages 35–38, 2008.

[KLMA07] Jérôme Kunegis, Andreas Lommatzsch, Martin Mehlitz, and Sahin Albayrak. Assessing the value of unrated items in collaborative filtering. In *Proc. Int. Conf. on Digital Information Management*, pages 212–216, 2007.

[KLPM10] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *Proc. Int. World Wide Web Conf.*, pages 591–600, 2010.

[KR93] Douglas J. Klein and Milan Randić. Resistance distance. *J. Math. Chemistry*, 12(1):81–95, 1993.

[KS07] Jérôme Kunegis and Stephan Schmidt. Collaborative filtering using electrical resistance network models with negative edges. In *Proc. Industrial Conf. on Data Mining*, pages 269–282, 2007.

[KSB+08] Jérôme Kunegis, Stephan Schmidt, Christian Bauckhage, Martin Mehlitz, and Sahin Albayrak. Modeling collaborative similarity with the signed resistance distance kernel. In *Proc. European Conf. on Artificial Intelligence*, pages 261–265, 2008.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proc. Int. World Wide Web Conf.*, pages 640–651, 2003.

[KSLL10] Jérôme Kunegis, Stephan Schmidt, Andreas Lommatzsch, and Jürgen Lerner. Spectral analysis of signed graphs for clustering, prediction and visualization. In *Proc. SIAM Int. Conf. on Data Mining*, pages 559–570, 2010.

[KSN+10] Jérôme Kunegis, Steffen Staab, Nicolas Neubauer, Klaus Obermayer, and Christian Bauckhage. Empirical verification of four complex network models on seventy-seven large network datasets. Unpublished, 2010.

[KSTC02] Jaz Kandola, John Shawe-Taylor, and Nello Cristianini. Learning semantic similarity. In *Advances in Neural Information Processing Systems*, pages 657–664, 2002.

[KY04] Bryan Klimt and Yiming Yang. The Enron corpus: A new dataset for email classification research. In *Proc. European Conf. on Machine Learning*, pages 217–226, 2004.

[LAH07] Jure Leskovec, Lada Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. on the Web*, 1(1), 2007.

[Lax84] Peter D. Lax. *Linear Algebra and Its Applications.* John Wiley & Sons, 1984.

[LBKT08] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, pages 462–470, 2008.

[Ley02] Michael Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *Proc. Int. Symposium on String Processing and Information Retrieval*, pages 1–10, 2002.

[LH08] Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. In *Proc. Int. World Wide Web Conf.*, pages 915–924, 2008.

[LHK10a] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Governance in social media: A case study of the Wikipedia promotion process. In *Proc. Int. Conf. on Weblogs and Social Media*, 2010.

[LHK10b] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proc. Int. Conf. on Human Factors in Computing Systems*, pages 1361–1370, 2010.

[LJD09] Zhengdong Lu, Prateek Jain, and Inderjit S. Dhillon. Geometry-aware metric learning. In *Proc. Int. Conf. on Machine Learning*, pages 673–680, 2009.

[LJZ09] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. Similarity index based on local paths for link prediction of complex networks. *Phys. Rev. E*, 80(4):046122, 2009.

[LKF07] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowledge Discovery from Data*, 1(1):1–40, 2007.

[LLDM08] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proc. Int. World Wide Web Conf.*, pages 695–704, 2008.

[LNK03] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proc. Int. Conf. on Information and Knowledge Management*, pages 556–559, 2003.

[LQCL09] Wei Liu, Buyue Qian, Jingyu Cui, and Jianzhuang Liu. Spectral kernel learning for semi-supervised classification. In *Proc. Int. Joint Conf. on Artifical Intelligence*, pages 1150–1155, 2009.

[LS00] Daniel D. Lee and Sebastian H. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2000.

[Lux07] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[LW00] Magnhild Lien and William Watkins. Dual graphs and knot invariants. *Linear Algebra and its Applications*, 306(1):123–130, 2000.

[LYRL04] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *J. Machine Learning Research*, 5:361–397, 2004.

[MA05] Paolo Massa and Paolo Avesani. Controversial users demand local trust metrics: an experimental study on epinions.com community. In *Proc. American Association for Artificial Intelligence Conf.*, pages 121–126, 2005.

[Maj06] Pawel Majewski. Directed Laplacian kernels for link analysis. In *Proc. Int. Conf. on Intelligent Data Engineering and Automated Learning*, pages 314–321, 2006.

[Mey00] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. Society for Industrial and Applied Mathematics, 2000.

[MF10] Andri Mirzal and Masashi Furukawa. Eigenvectors for clustering: Unipartite, bipartite, and directed graph cases. In *Proc. Int. Conf. on Electronics and Information Engineering*, pages 303–309, 2010.

[MH05] Paolo Massa and Conor Hayes. Page-reRank: Using trusted links to re-rank authority. In *Proc. Int. Conf. on Web Intelligence*, pages 614–617, 2005.

[Mil67] Stanley Milgram. The small-world problem. *Psychology Today*, 1(1):61–67, 1967.

[Mis09] Alan Mislove. *Online Social Networks: Measurement, Analysis, and Applications to Distributed Information Systems*. PhD thesis, Rice University, 2009.

[MKG+08] Alan Mislove, Hema Swetha Koppula, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Growth of the Flickr social network. In *Proc. Workshop on Online Social Networks*, pages 25–30, 2008.

[MMG+07] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proc. Internet Measurement Conf.*, 2007.

[Moh91] Bojan Mohar. The Laplacian spectrum of graphs. *Graph Theory, Combinatorics, Applications*, 2:871–898, 1991.

[MP07] Marina Meilă and William Pentney. Clustering by weighted cuts in directed graphs. In *Proc. SIAM Int. Conf. on Data Mining*, 2007.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[MS01] Marina Meilă and Jianbo Shi. A random walks view of spectral segmentation. In *Proc. Int. Conf. on Artificial Intelligence and Statistics*, 2001.

[MYC+10] Amin Mantrach, Luh Yen, Jérôme Callut, Kevin Françoisse, Masashi Shimbo, and Marco Saerens. The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(6):1112–1126, 2010.

[Nat10] National Institute of Standards and Technology. Text REtrieval Conference (TREC) English documents. `http://trec.nist.gov/data/docs_eng.html`, August 2010. Volume 4 & 5.

[New06a] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74(3):036104, 2006.

[New06b] M. E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Phys.*, 46(5):323–351, 2006.

[NJW01] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2001.

[NO10] Nicolas Neubauer and Klaus Obermayer. Analysis of the Visualize.us folksonomy. Unpublished, 2010.

[NZT07] Marc A. Najork, Hugo Zaragoza, and Michael J. Taylor. Hits on the Web: How does it compare? In *Proc. Int. Conf. on Research and Development in Information Retrieval*, pages 471–478, 2007.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[Per07] Oskar Perron. Zur Theorie der Matrices. *Math. Ann.*, 64(2):248–263, 1907.

[PFP+07] Gergely Palla, Illés J. Farkas, Péter Pollner, Imre Derényi, and Tamás Vicsek. Directed network modules. *New J. Phys.*, 9(6):186, 2007.

[Pro10] Prosper Marketplace, Inc. Prosper data export. `http://www.prosper.com/tools/DataExport.aspx`, October 2010. v1.2.6.

[Rea54] Kenneth E. Read. Cultures of the Central Highlands, New Guinea. *Southwestern J. of Anthropology*, 10(1):1–43, 1954.

[RFI02] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing J.*, 6, 2002.

[RLH09] Agnes Radl, Ulrike von Luxburg, and Matthias Hein. The resistance distance is meaningless for large random geometric graphs. In *Proc. Workshop on Analyzing Networks and Learning with Graphs*, 2009.

[Ros97] Boris Rosenfeld. *Geometry of Lie Groups*. Kluwer Academic Publishers, 1997.

[Roz83] Yurii A. Rozanov. *Markov Random Fields*. Springer, 1983.

[SCAK11] Stephan Spiegel, Jan Clausen, Sahin Albayrak, and Jérôme Kunegis. Link prediction on evolving data using tensor factorization. In *Proc. Workshop on Behavior Informatics*, 2011.

[SDLA10] Alan Said, Ernesto W. De Luca, and Sahin Albayrak. How social relationships affect user similarities. In *Proc. IUI Workshop on Social Recommender Systems*, 2010.

[SK03] Alex Smola and R. Kondor. Kernels and regularization on graphs. In *Proc. Conf. on Learning Theory and Kernel Machines*, pages 144–158, 2003.

[SKKR00] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender systems–a case study. In *Proc. ACM WebKDD Workshop*, 2000.

[SKKR01] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. Int. World Wide Web Conf.*, pages 285–295, 2001.

[SLZZ10] Ming-Sheng Shang, Linyuan Lü, Yi-Cheng Zhang, and Tao Zhou. Empirical analysis of Web-based user-object bipartite networks. *Europhys. Lett.*, 90(4):48006, 2010.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[Sob95] Garret Sobczyk. The hyperbolic number plane. *The College Math. J.*, 26:268–280, 1995.

[Ste90] Gilbert W. Stewart. Perturbation theory for the singular value decomposition. Technical report, Univ. of Maryland, College Park, 1990.

[Ste05] Daniel Stewart. Social status in an open-source community. *American Sociological Review*, 70(5):823–842, 2005.

[STF06] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, pages 374–383, 2006.

[TB06] George Theodorakopoulos and John S. Baras. Linear iterations on ordered semirings for trust metric computation and attack resiliency evaluation. In *Proc. Int. Symposium on Math. Theory of Networks and Systems*, pages 509–514, 2006.

[TM69] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[VMCG09] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in Facebook. In *Proc. Workshop on Online Social Networks*, pages 37–42, 2009.

[WC04] Yi Wu and Edward Y. Chang. Distance-function design and fusion for sequence data. In *Proc. Int. Conf. on Information and Knowledge Management*, pages 324–333, 2004.

[Wed72] Per-Åke Wedin. Perturbation bounds in connection with singular value decomposition. *BIT Numerical Math.*, 12(1), 1972.

[Wey12] Hermann Weyl. Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differenzialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung). *Math. Ann.*, 71(4):441–479, 1912.

[Wik10] Wikimedia Foundation. Wikimedia downloads. `http://download.wikimedia.org/`, January 2010.

[Wil65] James H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, 1965.

[WS97] M. Wax and J. Sheinvald. A least-squares approach to joint diagonalization. *IEEE Signal Processing Letters*, 4(2):52–53, 1997.

[WS98] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[WZB08] Robert Wetzker, Carsten Zimmermann, and Christian Bauckhage. Analyzing social bookmarking systems: A del.icio.us cookbook. In *Proc. Mining Social Data Workshop*, pages 26–30, 2008.

[YCL07] Bo Yang, William Cheung, and Jiming Liu. Community mining from signed social networks. *Trans. on Knowledge and Data Engineering*, 19(10):1333–1348, 2007.

[Zas82] Thomas Zaslavsky. Signed graphs. *Discrete Applied Math.*, 4:47–74, 1982.

[Zas08] Thomas Zaslavsky. Matrices in the theory of signed simple graphs. In *Proc. Int. Conf. Discrete Math.*, 2008.

[ZKGL05] Xiaojin Zhu, Jaz Kandola, Zoubin Ghahramani, and John Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2005.

[ZKLG06] Xiaojin Zhu, Jaz Kandola, John Lafferty, and Zoubin Ghahramani. *Semi-supervised Learning*, chapter Graph Kernels by Spectral Transforms. MIT Press, 2006.

[ZLMZ05] Beichuan Zhang, Raymond Liu, Daniel Massey, and Lixia Zhang. Collecting the Internet AS-level topology. *SIGCOMM Computer Communication Review*, 35(1):53–61, 2005.

[ZM08] Dell Zhang and Robert Mao. Classifying networked entities with modularity kernels. In *Proc. Conf. on Information and Knowledge Management*, pages 113–122, 2008.

[ZMKL05] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proc. Int. World Wide Web Conf.*, pages 22–32, 2005.

Index

Glossary of Symbols

a	arc, 12
c	clustering coefficient, 26
d	density, 22
$d(i)$	degree function, 15
e	edge, 12
$f(\lambda)$	spectral transformation function, 39
i, j	vertices, 12
m, n	vertex counts, 12
$m(i, j)$	edge multiplicity function, 12
$p(\lambda)$	power series, 40
$p(i, j)$	link prediction function, 54
r	rank, 18
t	time, 34
w	edge weight, 12
$w(i, j)$	edge weight function, 12
α, β, γ	parameters, 40
γ	power law exponent, 23
$\delta_{0.9}$	90-percentile effective diameter, 26
λ	eigenvalue, 18
ξ	algebraic conflict, 89
σ	singular value, 18
$\sigma(i, j)$	sign function, 80
$\mathbf{u}, \mathbf{v}$	eigenvectors, 17
A	arc set, 12
D	directed graph, 12
E	edge set, 12
$F(\mathbf{\Lambda})$	matrix spectral transformation function, 33
G	graph, 12
$K(\mathbf{A})$	graph kernel, 40
$R(\mathbf{X})$	rank reduction function, 41
T	end time, 34
V	vertex set, 12
$\mathcal{G}$	graph set, 22
$\mathbf{A}$	adjacency matrix, 14
$\mathbf{B}$	biadjacency matrix, 15
$\mathbf{D}$	degree matrix, 15
$\mathbf{L}$	Laplacian matrix, 16
$\mathbf{M}$	normalized biadjacency matrix, 16
$\mathbf{N}$	normalized adjacency matrix, 16
$\mathbf{U}, \mathbf{V}$	eigenvector matrices, 18
$\mathbf{Z}$	normalized Laplacian matrix, 16
$\mathbf{\Delta}$	spectral diagonality test matrix, 38
$\mathbf{\Lambda}$	eigenvalue matrix, 18
$\mathbf{\Sigma}$	singular value matrix, 18
$\mathbf{X}^{\mathrm{T}}$	matrix transpose, 14
$\|\mathbf{X}\|_{\mathrm{F}}$	Frobenius norm, 14
$\mathbf{X}^{+}$	Moore–Penrose pseudoinverse, 63

Dipl.-Inform. Jérôme Kunegis (Complete CV)

University of Koblenz–Landau
Institute for Web Science and Technologies
Universitätsstraße 1
56070 Koblenz
Germany

Tel: +49 261 287-2775
Fax: +49 261 287-100-2775
kunegis@uni-koblenz.de
uni-koblenz.de/~kunegis
twitter.com/kunegis
linkedin.com/in/kunegis

Education

Dr. rer. nat. (ongoing) est. 2011
University of Koblenz–Landau
Thesis: *On the Spectral Evolution of Large Networks*

Dipl.-Inform. 2006
Technische Universität Berlin
Thesis: *Using Integer Linear Programming for Search Results Optimization*

Baccalauréat 1999
Lycée Français de Berlin
Série scientifique

Abitur 1999
Lycée Français de Berlin
Leistungskurse: Mathematik, Physik

Diplôme national du brevet 1996
Lycée Français de Berlin
Série collège

Experience

Research Assistant 2010–present
Institute for Web Science and Technologies, University of Koblenz–Landau
WeKnowIt (EU FP7), ROBUST (EU FP7), MULTIPLA (DFG).

Research Assistant 2006–2010
DAI Lab, Technische Universität Berlin
Projects PIA (Personalized Information Agent), Universal Recommender (Semantic Recommendation Engine), Connected Living (Innovationszentrum "Vernetztes Leben"), Smart Senior (Intelligent Services for Elderly People), UCPM (User Centric Profile Management), WebTV (Semantic TV Recommendations), SERUM (Semantic Recommenders based on Large, Unstructured Data).

Undergrad Research Assistant 2002–2006
DAI Lab, Technische Universität Berlin
Projects Corinna und Cornelius (Speech-controlled Personal Digital Assistant), PIA (Personalized Information Agent), Synergie (Collaborative e-Science Platform).

Teaching

- Grundlagen der Datenbanken (Introduction to Database Systems), University of Koblenz–Landau, WS 2011/2012.

- Grundlagen der Datenbanken (Introduction to Database Systems), University of Koblenz–Landau, WS 2010/2011.

Conference Chairs

- Web Science Conf. (WebSci), 2011, Publicity Chair.
- Special Session on Uncertainty in Network Mining (UNM) at the Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU), 2010, Technical Chair.

Program Committees

- European Conf. on Information Retrieval (ECIR), poster track, 2012.
- Conf. on Information and Knowledge Management (CIKM), 2011.
- Web Science Conf. (WebSci), 2011.
- Int. Conf. on Knowledge Discovery and Data Mining (KDD), research track, 2011.
- Challenge on Context-aware Movie Recommendation (CAMRa) at the Conf. on Recommender Systems (RecSys), 2010.
- Special Session on Uncertainty in Network Mining (UNM) at the Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU), 2010.

Review Committees

- IEEE Trans. on Neural Networks (TNN), 2011.
- J. on Autonomous Agents and Multi-agent Systems (AAMAS), Special Issue on Agent Mining, 2011.
- ACM Trans. on Knowledge Discovery from Data (TKDD), 2011.
- TV Content Analysis (TVCA), 2011.
- Workshop on Ontologies and Lexical Resources (OntoLex) at the Int. Conf. on Computational Linguistics (COLING), 2010.

Supervised Theses

- Julia Preusse. Analysis of the WebUni Online Student Community, *Dipl.-Inform.*, Otto-von-Guericke-Universität Magdeburg, 2010.
- Stephan Spiegel. A Hybrid Approach to Recommender Systems based on Matrix Factorization, *Dipl.-Inform.*, Technische Universität Berlin, 2009.
- Christian Banik. Recommending Wiki Articles using Collaborative Filtering, *Dipl.-Inform.*, Technische Universität Berlin, 2009.
- Iris Breddin. Untersuchung der Klassifizierbarkeit und Klassifikation von Schweißnahtdaten, *Dipl.-Inform.*, Technische Universität Berlin, 2008.

Publications

[1] Searching microblogs: Coping with sparsity and document quality. Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. In *Proc. Int. Conf. on Information and Knowledge Management*, pages 183–188, 2011.

[2] Discriminating graphs through spectral projections. Damien Fay, Hamed Haddadi, Steve Uhlig, Liam Kilmartin, Andrew W. Moore, Jérôme Kunegis, and Marios Iliofotou. *Computer Networks*, 55(15):3458–3468, 2011.

[3] Bad news travels fast: A content-based analysis of interestingness on Twitter. Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. In *Proc. Web Science Conf.*, 2011.

[4] One community does not rule them all. Thomas Gottron, Jérôme Kunegis, Ansgar Scherp, and Steffen Staab. In *Proc. Web Science Conf.*, 2011.

[5] Link prediction on evolving data using tensor factorization. Stephan Spiegel, Jan Clausen, Sahin Albayrak, and Jérôme Kunegis. In *Proc. Workshop on Behavior Informatics*, 2011.

[6] On joint diagonalisation for dynamic network analysis. Damien Fay, Jérôme Kunegis, and Eiko Yoneki. Technical report, University of Cambridge, 2011.

[7] Exploiting hierarchical tags for context-awareness. Alan Said, Jérôme Kunegis, Ernesto William De Luca, and Sahin Albayrak. In *Proc. Workshop on Exploiting Semantic Annotations for Information Retrieval*, pages 35–36, 2010.

[8] Network growth and the spectral evolution model. Jérôme Kunegis, Damien Fay, and Christian Bauckhage. In *Proc. Int. Conf. on Information and Knowledge Management*, pages 739–748, 2010.

[9] The link prediction problem in bipartite networks. Jérôme Kunegis, Ernesto W. De Luca, and Sahin Albayrak. In *Proc. Int. Conf. in Information Processing and Management of Uncertainty in Knowledge-based Systems*, pages 380–389, 2010.

[10] Spectral analysis of signed graphs for clustering, prediction and visualization. Jérôme Kunegis, Stephan Schmidt, Andreas Lommatzsch, and Jürgen Lerner. In *Proc. SIAM Int. Conf. on Data Mining*, pages 559–570, 2010.

[11] Multilingual ontology-based user profile enrichment. Ernesto W. De Luca, Till Plumbaum, Jérôme Kunegis, and Sahin Albayrak. In *Proc. Workshop on the Multilingual Semantic Web*, pages 41–42, 2010.

[12] Connecting senior citizens through interactive TV. Barış Karataş, Torsten Schmidt, Jérôme Kunegis, and Till Plumbaum. In *Proc. Ambient Assisted Living*, 2010.

[13] The Universal Recommender, Jérôme Kunegis, Alan Said, and Winfried Umbrath. White paper, 2009.

[14] Learning spectral graph transformations for link prediction. Jérôme Kunegis and Andreas Lommatzsch. In *Proc. Int. Conf. on Machine Learning*, pages 561–568, 2009.

[15] The Slashdot Zoo: Mining a social network with negative edges. Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. In *Proc. Int. World Wide Web Conf.*, pages 741–750, 2009.

[16] Hydra: A hybrid recommender system [cross-linked rating and content information]. Stephan Spiegel, Jérôme Kunegis, and Fang Li. In *Proc. Workshop on Complex Networks in Information and Knowledge Management*, pages 75–80, 2009.

[17] Modeling collaborative similarity with the signed resistance distance kernel. Jérôme Kunegis, Stephan Schmidt, Christian Bauckhage, Martin Mehlitz, and Sahin Albayrak. In *Proc. European Conf. on Artificial Intelligence*, pages 261–265, 2008.

[18] Alternative similarity functions for graph kernels. Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. In *Proc. Int. Conf. on Pattern Recognition*, 2008.

[19] On the scalability of graph kernels applied to collaborative recommenders. Jérôme Kunegis, Andreas Lommatzsch, Christian Bauckhage, and Sahin Albayrak. In *Proc. ECAI Workshop on Recommender Systems*, pages 35–38, 2008.

[20] Assessing the value of unrated items in collaborative filtering. Jérôme Kunegis, Andreas Lommatzsch, Martin Mehlitz, and Sahin Albayrak. In *Proc. Int. Conf. on Digital Information Management*, pages 212–216, 2007.

[21] Adapting ratings in memory-based collaborative filtering using linear regression. Jérôme Kunegis and Sahin Albayrak. In *Proc. Int. Conf. on Information Reuse and Integration*, 2007.

[22] Collaborative filtering using electrical resistance network models with negative edges. Jérôme Kunegis and Stephan Schmidt. In *Proc. Industrial Conf. on Data Mining*, pages 269–282, 2007.

[23] Scalable bandwidth optimization in advance reservation networks. Stephan Schmidt and Jérôme Kunegis. In *Proc. Int. Conf. on Networks*, pages 95–100, 2007.

[24] Resource-aware update policy for highly dynamic P2P networks. Dragan Milošević, Jérôme Kunegis, and Sahin Albayrak. In *Proc. WI Workshops*, 2007.

[25] Using novel IR measures to learn optimal cluster structures for web information retrieval. Martin Mehlitz, Jérôme Kunegis, and Sahin Albayrak. In *Proc. Int. Conf. Web Intelligence*, 2007.

[26] A new evaluation measure for information retrieval systems. Martin Mehlitz, Christian Bauckhage, Jérôme Kunegis, and Sahin Albayrak. In *Proc. Int. Conf. on Systems, Man and Cybernetics*, 2007.

[27] A multi-agent framework for personalized information filtering. Andreas Lommatzsch, Martin Mehlitz, and Jérôme Kunegis. In *Proc. German e-Science*, 2007.

[28] Ein gradualisiertes Community-Modell zur Bildung wissenschaftlicher Gemeinschaften. Michael Hahne, Corinna Jung, Jérôme Kunegis, Andreas Lommatzsch, and André Paus. In *Analyse sozialer Netzwerke und Social Software – Grundlagen und Anwendungsbeispiele*, 2007.

Invited Talks

- On the Spectral Evolution of Large Networks. University College Cork, 2011.
- Time-aware Centrality in Contact Network Analysis. Eiko Yoneki, Damien Fay, Jérôme Kunegis. European Conference on Complex Systems, 2011.
- On the Spectral Evolution of Large Networks. Fraunhofer IAIS, St. Augustin, 2011.
- On the Spectral Evolution of Large Networks. Berlin Institute of Technology, 2011.
- The Slashdot Zoo: Mining a Social Network with Negative Edges. Data and Knowledge Engineering Research Colloquium, Otto-von-Guericke University Magdeburg, 2010.
- The Slashdot Zoo: Mining a Social Network with Negative Edges. University of Koblenz–Landau, 2010.
- The Slashdot Zoo: Mining a Social Network with Negative Edges. University of Hannover, 2010.
- PIA+COMM – an Intelligent Search Engine. Michael Hahne, Corinna Jung, Jérôme Kunegis, Andreas Lommatzsch and André Paus. Workshop on the Formation of Social Networks in Social Software Applications, INFORMATIK, 2006.

Posters without Publication

- Need Networks? KONECT – The Koblenz Network Collection. European Summer School on Information Retrieval, 2011.
- KONECT – The Koblenz Network Collection. Web Science Conf., 2011.
- Uncovering Multi-modal Spread Modes using Joint Diagonalization in Dynamic Human Contact Networks. Damien Fay, Jérôme Kunegis, Eiko Yoneki. Interdisciplinary Workshop on Information and Decision in Social Networks, 2011.
- On the Spectral Evolution of Large Networks. Postersession für Nachwuchswissenschaftler/innen, University of Koblenz–Landau, 2011.
- On the Spectral Evolution of Large Networks. SIAM Conf. on Data Mining Doctoral Forum, 2010.

Demonstrations and Presentations

- LiveTweet: Monitoring and Predicting Interesting Microblog Posts, European Conference on Information Retrieval (ECIR), 2012.
- Schülerinformationstage, Institute for Web Science and Technologies, University of Koblenz–Landau, 2011.
- PIA, Spree, CeBIT, 2010.
- PIA+COMM, CeBIT, 2007.
- PIA, Lange Nacht der Wissenschaften Berlin/Potsdam, 2006.

Awards

- Student Travel Award, Int. Conf. on Information and Knowledge Management, 2010.
- Student Travel Fellowship Award, SIAM Conf. on Data Mining, 2010.
- Best Paper Runner-up, Industrial Conference on Data Mining, 2007.

Miscellaneous

- Birthday: 1981-03-27.
- Languages: French (first language), German (first language), English (advanced).
- Main programming languages: C, C++, Java, Matlab, Bourne Shell.
- Secondary programming languages: Perl, BASIC, Logo, Turbo Pascal, PHP.
- Technologies: SED, Makefile, Latex, HTML.