This is to certify that the
dissertation entitled

EXPLOITING THE LINK STRUCTURE IN MINING NETWORK DATA

presented by

JERRY SCRIPPS

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Computer Science

Pan Payne

Major Professor's Signature

6/24/2009

Date

Exploiting the Link Structure in Mining Network Data

By

Jerry Scripps

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science

2009

ABSTRACT

Exploiting the Link Structure in Mining Network Data

By

Jerry Scripps

The study of networks in general and social networks in particular, has intensified in recent years due in part to the interest in on-line social networks and the availability of large data sets of related objects. An area called network mining has emerged from the larger area of data mining, whose purpose is to extract hidden knowledge from large, linked data sets.

It is the purpose of this dissertation to study the relationships that develop in networks involving links, specifically the relationships between links and communities and between links and attributes. Understanding the alignment between communities and the links offers valuable insights into the roles that nodes play with respect to communities. It will also be shown that learning the alignment between links and attributes leads to improvements in link prediction and collective classification. Finally, studying the changes in the relationship of attributes to links over time has revealed information helpful for decisions that are made in processing network data.

During the course of this investigation, a number of tangible new algorithms and metrics have been discovered. First, a new metric is introduced that provides information about the number of communities to which a node belongs without having the actual community information. Combining this rawComm metric with the relative degree of a node allows community-based roles to be assigned to nodes. Next, a new framework is proposed that uses weights to align the attributes to the link structure. Two formulations of the framework are used for improving link prediction and collective classification techniques. It is also shown to be valuable in studying the dynamics of temporal networks.

To Sally

ACKNOWLEDGEMENTS

If I have seen further it is only by standing on the shoulders of giants. – Sir Issac Newton

This has been a long, long process. It has laid bare my weaknesses and exposed my dependencies. There are so many without whom I would not have finished this thesis. As I sit and write this my eyes are filled with tears and my heart is filled with gratitude.

To Dr. Pang-Ning Tan, my advisor and friend. So many times, when I was discouraged, you appeared not to noticed but just simply helped me to press on and find the nugget of truth we were looking for. I am thankful for your quiet enthusiasm, patience and guidance. You have rekindled my love of learning and refocused my dedication to teaching. To my committee, Drs. Abdol-Hossein Esfahanian, Anil Jain and Lyudmila Sakhanenko, I thank you. As instructors you shaped my thoughts. And in my thoughts you stayed as I worked on my research, motivating me to measure up to your standards. I appreciate your comments and guidance. I will miss the weekly meetings with Pang-Ning, Abdol and Ron Nussbaum. I learned much and had fun. I would also like to thank Dr. Edward Rothwell for sitting on my committee for the first year and for his encouraging comments.

I feel privileged to have had the many instructors, mentors and friends that I have had at MSU. Drs. Torng, McKinley, and Jin, it has been an honor to study with you and my thanks are not enough. I am also grateful to Dr. Weinshank, my Master's project advisor and mentor. And to Carl Page, a wonderful teacher and a gentleman. Thanks also to the staff at CSE: Linda, Cathy and Norma for helping me to navigate the labyrinth of paper work and to Adam for his help with technical issues. Finally, I would also like to say thank you to Lora Mae Higbee, a secretary from my early days at MSU – I miss your kind words of encouragement. I am grateful for the many friendships that I have made at MSU. To the long-ago graduated friends from the 1990's you are still in my thoughts. I think that an animal-Olympics reunion is in order. To my more recent friends, Haibin, Hamid, Samah, Feilong, Zubin, Jing Gao, Sohraab, Kapila: there was not enough time. My biggest regret about finishing up at MSU is not having more time to talk with you. I feel richer for having known you.

I owe so much to my friends and coworkers at Grand Valley State University. First, I would like to thank our Dean, Paul Plotkowski for his generous support. Second, Paul Leidig, our school director who helped to ease my burden. Thanks to all faculty for your support. Thanks to Christian and Hugh for getting me started; to Christian and Anna for their constant, gentle words of encouragement; to Greg, Jamal and Roger for listening to me; to George for badgering me (in a good way). Thanks to Scott, Joe, Diane and Cheryl for making my life easier. Cheryl made everyone sing "happy birthday" to me at a time that she knew I needed some cheering up. To Theresa and Dave, for suffering with me, your time will come. Thanks to all for believing in me.

Thanks to all friends. To those from Rockford: you have been loyal friends throughout the years and over good times and bad. To my triathlon buddies: you have given me fellowship during times of struggle and growth. To our friends from Alexander St. and Wealthy school: your words of encouragement and friendship have helped to sustain me in difficult times. Thanks to all for the occasional distraction. Thanks to the people at Flees's Taekwando for boosting my confidence.

To my family, who have shaped and sustained me, my thanks are not enough. My father taught me right from wrong. My brother Pat taught me the things they do not cover in school. My sister Debbie taught me the importance of family. And Wendy taught me to enjoy life. Thanks to step-siblings Bennie, Dirk, Connie and Barb for your kindness and friendship. To all the in-laws, nieces, nephews, aunts, uncles, cousins: we are all interconnected and without any one of you my network is incomplete.

To my children: thank you for your patience with me during this process. I love you all and look forward to spending more time with you. To my wife Sally, you more than anyone share in this accomplishment. In the face of sacrifice and hardship you never complained or criticized but offered only encouragement and belief. Thank you for the life and happiness that you brought to me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

It's not the towering sail, but the unseen wind that moves the ship. - Proverb

In a social network, the links between people can often be, like wind, an invisible force that lies behind other perceptions. We talk about groups or cliques within a company or school but it is the links that drive the formation of these. We talk about the attractive qualities of a person but it is our friendships that allow us to identify these.

Consider a college campus where students have academic, social and athletic attributes, such as major area of study, musical tastes and intramural sports participation. They also establish friendships with other students. It is sometimes possible to look past the friendships, to put them in our subconscious, that allow us to see higher level concepts.

For example, the students are often associated with roles with respect to the groups with which they are associated. Without considering all of the specific friendships that a student has we can label them as loners or popular within a group. Some students might be known as good ambassadors because they have many friends and several diverse groups. It is possible to observe the role without consciously recalling all of the friendships.

Another example involves using friendship links to identify the characteristics that some people find attractive. A student can have friends that have some things in common and other things varied. Say a student, Risha, has friends that all enjoy sports. Her friends have come from many social and ethnic backgrounds and study many diverse subjects but most have strong feelings about sports. By knowing this we can infer many things. For example, if we meet a new student who enjoys sports we might introduce them to

Risha, assuming they could become friends. Or say that we do not know if a student likes sports, but if we know that they are friends with Risha, it would seem likely that they are interested in sports.

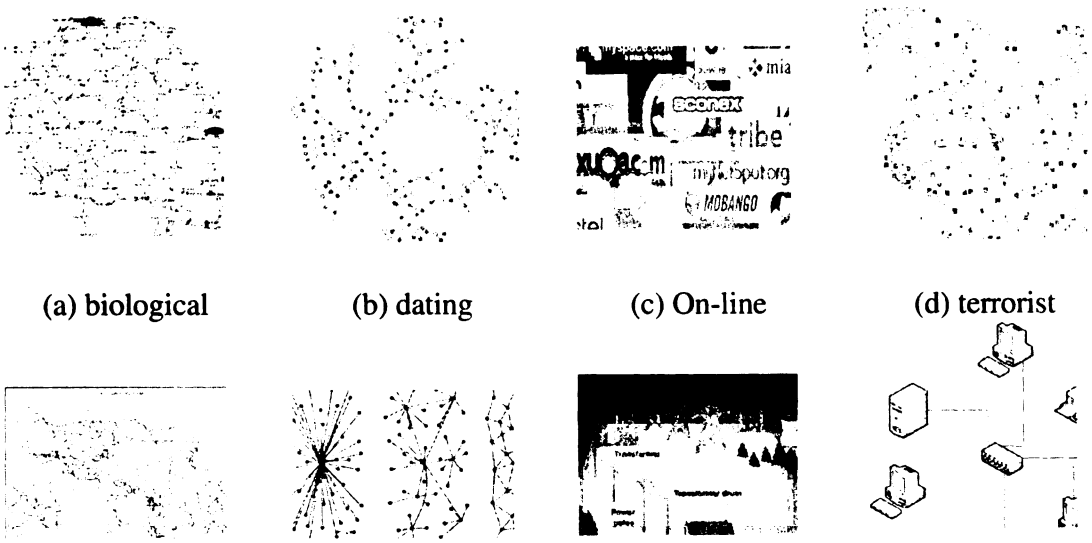
Our reasoning allows us to make these kinds of judgments for the relatively small group of friends that we have. Using computers and learning algorithms we can make more precise judgments on much larger networks of people (or of other things). This thesis is a compilation of my work in the area of network mining, in which links are used to discover other important and interesting bits of knowledge.

1.1 Social and Other Networks

To a social scientist, a social network is a finite set of actors (persons, companies, etc.) that have defined relationships. In the current work, actors are called nodes and can represent any object that is capable of having relationships with other objects. A formal definition will be given in the next section but it should be noted that while the analysis in this thesis can apply to many different types of networks, it is often intuitive to discuss their use with social networks. The rest of this section will offer evidence to support why studying networks is important, interesting and valuable.

Research in the area of social networking has increased dramatically in recent years. In a recent search in Google Scholar the terms "data mining" and "social networks" produced about 1.4 million and 2.75 million hits respectively. Even the more common term "politics" garnered only 2.2 million hits. As another measure, the number of conferences that are concerned with social networks is also significant. The Conference Service Mandl website (www.conference-service.com) maintains information about scientific academic conferences. Searching for "social network" returns 161 hits, while "knowledge discovery" and "data mining" return 218 and 272 hits respectively.

There are many reasons for the increased interest in social networking. First, it has a broad appeal to a wide range of disciplines. Social scientists have found that studying networks provides a new perspective for answering behavioral science questions [118]. While traditional economic theory assumes that economic agents interact with all others,



(a) biological (b) dating (c) On-line (d) terrorist

(e) highway (f) data comm. (g) power grid (h) computer

Figure 1.1: Examples of Networks

consideration of network relationships appears to be gaining attention [82]. Networks can be found in modeling genetic regulatory networks [28] and other biological areas, in cognitive science [71] and law enforcement [17]. This is not an exhaustive list but a few selected instances of disciplines that are starting to consider or already have considered the use of networks in their analysis.

Second, traditional analysis methods that rely on unrelated objects are often insufficient to explain the complex real life phenomenon that they attempt to model. For example, the orbit of the Moon around the Earth can be crudely approximated as a circle by considering only the relationship between the Earth and the Moon. However, to accurately describe it, one must consider the effect of other objects, such as the Sun, as well. Similarly, one cannot ignore the influence of friends and instructors when analyzing a student's choice of academic study. While considering all dependencies in a network scenario typically leads to intractable solutions, modeling the data using networks helps analysts to discover good approximate solutions.

Third, the recent popularity of social networking websites has raised the awareness and availability of such data. Early sites such as Classmates.com and SixDegrees.com, came on-line in 1995 but the phenomenon of social networking really began to gather

momentum with the advent of Friendster, MySpace, FaceBook and other sites which made linking and posting content fast and easy. As these sites grew, it opened up new opportunities for researchers and analysts. Social scientists who had to be content with smaller networks of 100 nodes or less can now perform their analysis on much larger and more statistically significant data sets. A Google search on *Business Networks* returns a long list of articles and books on using social networks within business and for personal career growth. Using social networks to improve the shopping experience for customers and increase sales for retailers is also a growing trend [43].

The behavior of network data is interesting because of the rich structures resulting from the links between the pairs of objects. Considering pairs (called dyads) or triads (triples) of linked objects is often used in the analysis of social networks [118] but can also be used for other higher level activities such as finding communities [6]. Communities, which result from the link placements, are themselves interesting subjects of study [98]. Hierarchies [21], trees [41] and other interesting structures have also been studied.

Techniques that have risen from studying networks have made vital contributions to the areas of web search [60, 87], advertising [30, 59] and law enforcement [108]. There are many more promising areas such as the transformation of power grids to smart grids and web 2.0 applications.

1.2 Network Mining

Descriptive statistics can easily be applied to social networks to find out such things as the average age of members, percentage of females, etc. As well, graph invariants can be used to find the average number of friends and distance – or degrees of separation – between two people. A deeper understanding of the network, though, can be achieved by some newer techniques.

With the increased interest in social networking there has been a corresponding increase in the research of mining this data. This area will be referred to as network mining in this thesis but has also been called link mining [38]. It is a convergence of the disciplines of machine learning, social network analysis, pattern recognition, graph theory and

data mining.

Theoretically, what sets network mining apart is that it considers objects to be related. Traditional data mining techniques, mostly make the assumption that objects consist of vectors of attributes and are i.i.d. (independently and identically distributed). In network mining the links provide evidence of a relationship between the nodes. In many cases this is more realistic than the i.i.d. assumption; however it also complicates things.

For example, in Figure 1.2, some probabilities can be easily collected if the i.i.d. assumption is made. Like, $p(smoker) = \frac{\sum smoker}{\sum smoker + \sum non-smoker}$. However, when we consider the links as conditional dependencies, it makes the analysis much more complex, since the probability of a node depends on its neighbors, which depend on their neighbors, etc.

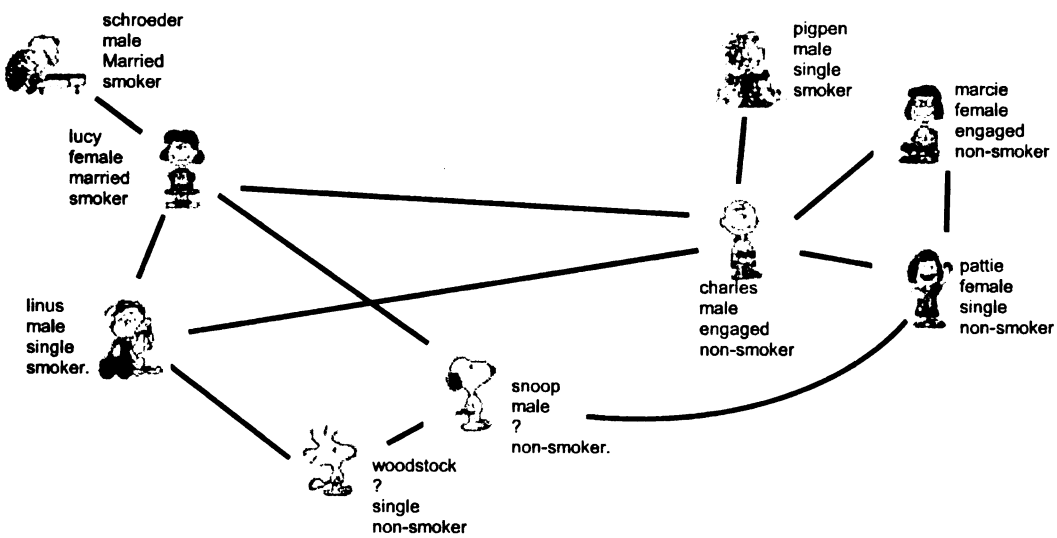


Figure 1.2: Sample Network

There have been a number of proposals for techniques to deal with the conditional dependencies. Some make a Markov type of assumption where a node is dependent only upon its immediate neighbors. Others use Monte Carlo sampling techniques to approxi-

mate the conditional probabilities. All of the works proposed involve some simplification. They are grouped into five techniques. *Link prediction* is where new or missing links are inferred from existing network data. In Figure 1.2 we may be interested in predicting who will be the next pair of people to become friends or perhaps some of the links are missing and we want to discover which ones. *Ranking* is the process of ranking nodes based on their authority within the network. *Collective classification* involves using the links between nodes has been shown to boost the performance of more traditional classification schemes. Using the network in Figure 1.2 we could predict whether Snoop's is a smoker or not based on his attributes and the status of his friends. *Community finding* is the technique that discovers groups in the network based on a combination of the the attributes of nodes and the links between them. Finally, *influence maximization* identifies highly influential nodes. These techniques will be described in more detail in Chapter 2. Table 1.1 lists a few applications that have been used with the different techniques.

Table 1.1: Applications of Network Mining Techniques

Technique	**Applications**
link prediction	protein interactions
	bibliographic collaborations
	locate missing links in hypertext
collective classification	political affiliations in networks
	counterterrorism detection
	stock fraud detection
influence maximization	advertising and promotion
	infectious disease spread
community finding	form clusters of legislators
	find demographic groups in network data
	grouping web search results by topic
node ranking	web search
	document retrieval

While great strides have been made in network mining there remains much more work

to be done. Improvements can be made to existing techniques. Techniques can also be expanded to adapt to circumstances not considered earlier, such as the dimension of time. Most proposals have considered only stationary networks, whereas now they are being reconsidered in the face of evolving networks. In the next section some of the challenges that relate to this thesis are discussed.

1.3 Challenges

With the addition of link information, networks are richer than attribute vectors but also more complex. A number of challenges have been identified below that must be dealt with so that the link structure can be better utilized.

1. With an attribute vector, objects can be clustered using their attributes alone but with the addition of links, an underlying community structure is assumed. While there are many algorithms [9, 18, 40, 49, 54, 79, 81, 93, 110, 113] to find communities there may be times where community finding is intractable or not undesirable for some other reason. Still, even without knowledge of communities, it is helpful to characterize nodes according to their community belongingness. For example, a person that has many friends in several different communities would likely be an influential person and thus be a good target for a promotional campaign. Graph theory and social network analysis have offered many useful metrics that allow nodes in a network to be characterized in some meaningful way. Until recently though, there have not been any characterizations of a node with respect to the community structure and these are only useful if the community structure is known. The first challenge is to provide a metric to measure the community belongingness without regard to any specific community finding algorithm.

2. Many methods have been proposed (see Section 2) to model network behavior. The proposals have been dominated by generative models. Generative models define a joint probability over some portion of the attributes, links and communities and then solve for the parameters using ML, MAP or some other estimation technique. Often

the models can then be used to predict new links or classify nodes. By contrast, discriminative methods attempt to find the optimal solution directly. Discriminative methods are thought to be more effective than generative solutions [83].

The advantage of generative models though is that because of the joint probabilities their solutions reflect the impact of the attributes and links jointly. Methods of discriminative link prediction [14, 19, 45, 70, 107, 109, 125] and classification [1, 47, 55, 63, 68, 89, 90, 92] have been proposed but they consider the effect of the links and attributes separately if at all. Several of the generative models are capable of modification so that they can be used for predicting links as well as classes. Having a framework that directly models the alignment between links and attributes in a discriminative way would be advantageous.

3. Prediction methods often assume a linear relationship between the data and the predicted values. While this makes the analysis simpler or, in some cases, feasible, in reality the relationship is often more complex than just linear. Kernel functions can be employed in some circumstances to work around this limitation. For instance, the classification technique of SVM [117] is designed to find a linear separation between two classes of objects but it can be transformed by a kernel function. Applying the kernel trick, the linear kernel can be replaced with a higher dimensional function to find a non-linear separation. In many network prediction tasks it is likely that boundaries are also non-linear. Thus far there have not been any techniques proposed that are adaptable to kernel functions.

4. Temporal networks, where changes can be seen over time, are just starting to become the object of study for network miners. When building temporal networks analysts must often make several preprocessing decisions that can affect the results of their analysis. For example, given transactions (changes to attributes and links) over time, a network can be built by either accumulating the data over time (once linked, always linked) or by using only the data for a particular period (links can appear or disappear). The final challenge, then is to study these decisions and the impact they have on the networks so that analysts can be confident of their decisions.

1.4 Contributions

It is the purpose of this thesis to study the relationships that develop in networks involving links, specifically the relationships between links and communities and between links and attributes. Understanding the alignment between communities and the links offers valuable insights into the roles that nodes play with respect to communities. It will also be shown that learning the alignment between links and attributes leads to improvements in link prediction and collective classification. Finally, studying the changes in the relationship of attributes to links over time has revealed information helpful for decisions that are made in processing network data.

In the course of studying these relationships the following contributions are offered:

- A community-belongingness metric is developed that provides an efficient way to approximate the number of communities to which a node belongs. This allows for a node to be assigned a role that it plays within the network according to its popularity (number of links) and community-belongingness. This information is shown to be helpful in the technique of influence maximization.

- A new matrix alignment framework for aligning the structure of the links to the attributes is derived. In this framework, attributes are weighted according to their importance in terms of linking. Matrix alignment is shown to be effective at both link prediction and collective classification. In addition, the framework is modified to make effective use a kernel function.

- The preprocessing decisions that analysts and researchers make when building networks from data can have a drastic impact on the analysis that is performed. The matrix alignment framework is used to study these decisions. The results of the experiments clearly show the effects that various decisions have.

1.5 Dissertation Structure

This introduction has informally presented the challenges that will be addressed in later chapters. In Chapter 2, the concepts used throughout the remaining chapters will be formally defined. A discussion of recent literature in the link mining area that pertains to this work will also be presented in that chapter. The remaining chapters (3 - 6) provide the details of the proposed solutions to the challenges listed above. The results from these chapters have been published in major conferences such as KDD [102, 105], ICDM [103], SDM [99] and ICPR [104]. Extended version of these work have also been submitted to IEEE Transactions on Knowledge and Data Engineering [100] and Data Mining and Knowledge Discovery Journal [106].

The challenge of finding a metric that accurately and objectively estimates the number of communities to which a node belongs is met with the definition of the rawComm statistic defined in Chapter 3. Chapter 4 presents a matrix alignment approach to link prediction. It is a discriminative approach that weights the attributes to align them to the links. The matrix alignment framework is applied to the problem of collective classification in Chapter 5. Here it modified so that it can directly learn the relative weights for links and attributes that are important for two nodes having the same class. These weights can then be used to infer the class of an unlabeled node. In Chapter 6 the matrix alignment framework is used to study the effects of preprocessing decisions (such as accumulating links or attributes over time) on the analysis of temporal networks. Also examined is the validity of the modeling assumptions (e.g., first-order Markovian assumption) and the empirical justification behind it. The chapter also introduces several metrics to measure the forces of selection and influence from dynamic network data. The matrix alignment framework is also extended to use kernel functions and is used to predict links in temporal networks.

Chapter 2

Preliminaries

In this chapter the groundwork will be laid for Chapters 3, 4, 5 and 6. First in Section 2.1, the network structures and their notations will be defined. Then in Section 2.2, is an overview of the network generators and models that have been proposed and used for network mining. Finally, Section 2.3 provides a review of the major techniques of network mining.

2.1 Definitions

A *network* is a collection of *nodes* which can be assigned attribute values. In addition to having attributes, the nodes can have relationships with other nodes. The relationships can be represented by a graph $G = (V, E)$ where where $V = \{1, 2, ..., |V|\}$ is the set of nodes and $E \subseteq V \times V$ is the set of links. Links can be directional to denote asymmetric relationships between objects (e.g., influence of an individual over another or hyperlink from one Web page to another). In addition to direction, links can be weighted or of different types. Cycles can be allowed or prohibited. In this work, links are generally non-directional, unweighted and of a single type.

A node in a network can have a vector descriptive attributes. In other disciplines, like pattern recognition or data mining, a node is referred to as an instance, object or a sample; attributes are often referred to as features. Although in general, attributes can be numeric or categorical, in this work they are generally conceived as binary. A selected attribute

can be used for the class in order to label nodes. Nodes can also be grouped into sets of communities.

Networks in this thesis will be encoded as matrices. For a network with n nodes, each having d attributes, let $A = [a_{ij}]_{n \times n}$ be the adjacency matrix of the graph where $a_{ij} = 1$ if there is a link between nodes i and j and zero otherwise. For the attributes, let $X = [x_{ik}]_{n \times d}$ be the data matrix where $x_{ik} \in \Re$ is the k^{th} attribute value for node i. Where communities are considered, a set of communities is defined as $\mathcal{C} = \{C_1, ...C_c\}$ where $C_i \in V, \forall i$ and $\cup C_i = V$ and $\cap C_i = \emptyset$. In Chapter 4, notation for communities is defined in Section 4.3.4.

Currently, in the network mining field there is a lot of excitement about networks that change over time. Since definitions vary we offer the taxonomy of Figure 2.1. A static network is a single snapshot – that is a set of nodes, related attributes and links. Most of the work until recently has been done using static networks. Temporal networks are collections of static networks. Given a temporal network, a single snapshot of the network is a static network. Temporal networks can be dynamic or stationary. Both dynamic and stationary networks change over time, according to some network parameters (which are discussed in the next section). The difference between stationary and dynamic networks is that with stationary networks the parameters do not change but with dynamic networks they do.

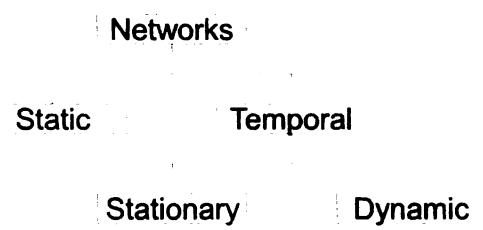


Figure 2.1: Network taxonomy

2.1.1 Network Types

There has been much work done in identifying the characteristics of networks. This is helpful for analysis of networks but also for model design. Over the years several network types have emerged. Random networks [33] appear to have links placed between randomly chosen nodes. The degree sequence follows a Poisson distribution $p(d(v_i) =$

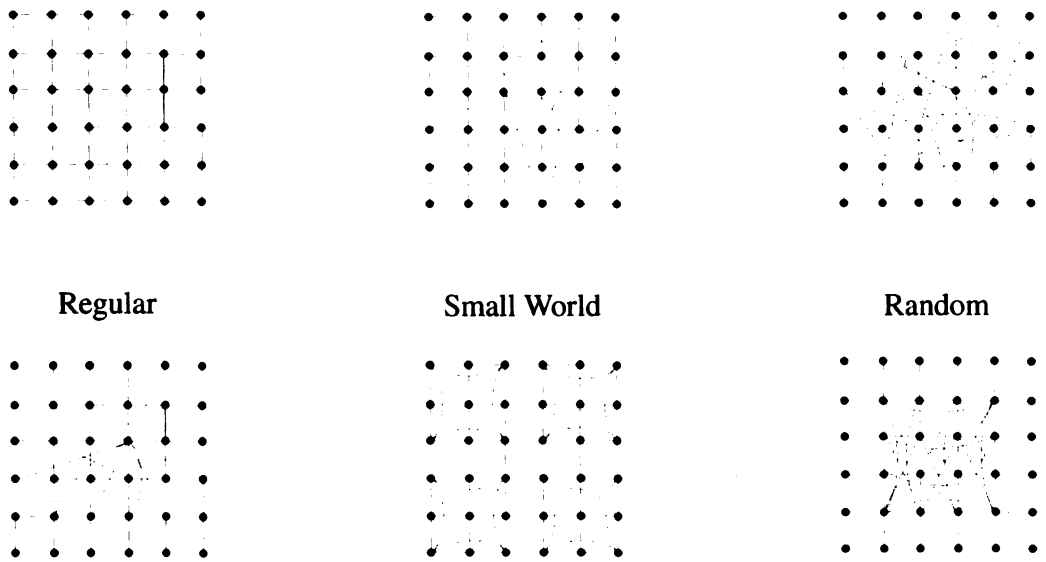


Figure 2.2: Network Types

$k|\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$, where $d(v_i)$ is the degree or number of links connected to node v_i. On the other hand, regular networks are typically lattices where all nodes have the same degree. Two metrics, *average path length* and *clustering coefficient* show how these two network types differ. The average path length $\frac{\sum_{ij \in V} path_{ij}}{n(n-1)/2}$ is the average of the shortest path ($path_{ij}$) between all pairs of nodes. Clustering coefficient for a node i, is defined as $c_i = \frac{2|\{e_{jk}\}|}{d(i)(d(i)-1)}$ for $v_j, v_k \in N_i$ and $e_{jk} \in E$ where E is the set of links and N_i are the neighbors (nodes that are directly connected) of i. Random graphs tend to have a low clustering coefficient and a low average path length, where regular graphs have high values for both.

A small world network is a cross between the previous two, having the short average path length of random networks but the high clustering coefficient of a regular network. Many networks exhibit a degree sequence that follows a power law distribution, $p(d(v_i) = k) = \frac{1}{k^n}$ where n is a constant. A network follows a power law degree sequence with the same constant at any time during its growth is considered a scale-free network. Two others, whose characteristics are not as well defined are core-periphery and cellular. A

core periphery is a network with a core set of nodes that are tightly connected to each other and those on the periphery, while the peripheral nodes are connected to the core but not each other. Cellular networks have several tightly connected cores.

Figure 2.2 provides a visual depiction of the different network types. The properties of these network types are summarized in Table 2.1 in terms of well-known metrics such as degree distribution, clustering coefficient and path length.

Table 2.1: Summary of Network Types

Network Type	Properties	References
Regular	Uniform degree distribution	
Random	poisson degree distribution	[33]
Small World	high clustering coefficient and low average path length	[120]
Scale-free	Power Law degree distribution	[7]
Cellular	tightly connected cells with sparse interconnections	[2, 35]
Core-periphery	single, tightly connected core of nodes with periphery points connected only to the core	[2, 16]

2.1.2 Forces of Change within Networks

Most networks are temporal in nature. Over time the state of the network changes as nodes are added and deleted, attributes are modified, links are formed and removed and communities undergo change. Although the changes appear to be random, some general forces have been observed. With scale-free networks nodes a links are added in such a way, so that the network continues to exhibit the scale-free property. The rather complex, forest fire model [66] has a number of guidelines that dictate changes in the network. Two other forces have come from the area of social network analysis (SNA).

SNA is an area within sociology that has some significant overlap in the network mining area. SNA researchers are primarily concerned with the structure and growth of social networks. Two interesting growth processes that have been discovered are selection and influence. Selection (also called homophily) is the force that drives people to make friends with others with similar attributes. For example, a smoker will tend to befriend other smokers. Influence (also called assimilation) is the force that coerces people to become more like their friends. So a person who has many friends that enjoy heavy metal

music will likely also become head bangers.

These forces operate in networks other than just social networks. Recently, a study [23] was made using a social network created from examining the editors of Wikipedia and their activity. The authors found that similarity between editors formed a sigmoid curve where it increased just before they made contact with each other (linked) and then continued to increase for a while before tapering off. This suggests that editors tend to link (selection) to others who are similar and then become more similar (influence) after the initial contact.

2.2 Network Generators and Models

This section describes network generators and models – tools that researchers use for studying networks. Network generators or sampling algorithms [2] are algorithms designed to create a network topology which exhibits the properties of a given network type. Parameters can be used to tune the resulting network graph.

Getoor, et al. [39] describe their probabilistic relational model as a "statistical framework for content and links". Extending this definition slightly, let network models refer to statistical frameworks for networks which could involve the activities of attributes, links and communities.

Network generators and models are useful for the study of networks for several reasons. Generators allow researchers to create synthetic data for experimenting and to help determine to which network type a sample network belongs. Network models are useful for learning the parameters that govern the network growth and for prediction tasks such as collective classification and link prediction.

2.2.1 Generators

For some of the network types described in Section 2.1.1 a number of generators have been proposed to explain their evolution. These generators are unlike the models described in the next section in that they help to explain the global properties but are not intended to predict values of specific node classes or links.

Here we recap the concepts pertinent to this thesis – the paper by Airoldi and Carley [2] provides a more thorough investigation. Random network generators simply place nodes between randomly selected nodes using a uniform probability distribution. The original small world generator starts with a regular graph and *rewires* some randomly selected edges. Scale-free networks can be generated using the principle of preferential attachment. Under this "rich get richer" process, nodes are chosen for new links with a probability proportional to their degree. They

Most generators either assume that nodes are added with time or that the network contains a fixed number of nodes. The model by Moore, et al. [75], uses preferential attachment but allows for nodes to be deleted. The networks generated with their model where the growth rates are much higher than shrinkage rates result in degree distributions that are the closest to real networks.

Leskovec et al. [66] developed a new model to account for certain properties that were revealed in their study of real evolving networks. Their model, which is based on a "forest fire" spreading process, exhibits characteristics such as shrinking network diameter, increasing densification, as well as power law degree distribution.

Another model that has been used to simulate social networks is agent-based modeling and simulation (ABMS) [72]. In an ABMS, agents – autonomous software objects that have goals and rules – interact with each other, causing them to modify their characteristics and even their behavior. ABMS's are not designed to generate a network of a specific type but have been shown to be helpful in understanding the dynamics of a social network using agents as nodes and setting the links based on the interactions of the agents[61].

2.2.2 Graphical Models

In the network mining literature there are a number of graphical models that use graphs to represent the relational domain where nodes are random variables and links between the nodes describe the inter-dependency. Learning the conditional probabilities for an exact joint probability distribution of even a small network is intractable, so these models use various methods to approximate the distribution and learn the parameters.

Statistical models are often divided into two groups – generative and discriminative.

According to Ng and Jordan [84]:

> Generative classifers learn a model of the joint probability, $p(x, y)$, of the inputs x and the label y, and make their predictions by using Bayes rules to calculate $p(y|x)$, and then picking the most likely label y. Discriminative classifers model the posterior $p(y|x)$ directly, or learn a direct map from inputs x to the class labels.

Applying these categories to graphical models has not been straightforward. As recently as 2006 at an ICML workshop, researchers struggled to agree to a common definition. Blei [15] claims that the difference is whether a model permits the addition of nodes and links. Handcock [15] suggests other definitions such as "the ability to simulate network structures with given structural properties" and "dynamic changing edges and structures". Although we will apply the definition used by Ng and Jordan above to network models, some properties of the two types have emerged. Generative models can be used to "generate" a network and compare the probabilities of two different network states. On the other hand, discriminative models are designed around a particular task and so can be more finely tuned.

Generative models

A family of generative models that have been used extensively, particularly in social network analysis, is the family of exponential random graph models (EGRM). Given a set of random variables $X = \{x_1 ... x_n\}$ a *dependency graph* $G = (V, E)$ can be created can be created to represent the dependencies by assigning each variable x_i to a node v_i and putting in an edge (v_i, v_j) if there is a dependency between x_i and x_j. A dependency graph where a variable x_i is dependent only upon its immediate neighbors is called a Markov random field. By the Hammersley-Clifford theorem [11], a Markov random field behaves according to the Gibbs distribution:

$$p(x) = \prod_{c \in C} V_c(x) \tag{2.1}$$

where C is the set of cliques in G and V_c is a clique potential function that depends only upon the nodes in c. Equation 2.1 is a general form of EGRM. In practice the form is simplified by using functions that involve pairs (dyads) or triplets of nodes such as

$$p(G) = \frac{1}{Z(\theta)} e^{\theta^T f(G)} \tag{2.2}$$

where $f(G) : \Re_{n \times n} \rightarrow \Re$ is a set of network sufficient statistics (density, transitivity, etc.), $\theta \in \Re^d$ is a vector of weights and $Z(\theta)$ is a normalizing term. Where $f(G)$ is a function of node pairs, the length of θ becomes $n(n-1)/2$ – an impractically large number of parameters. To make Equation 2.2 more tractable, various assumptions can be made. The p_1 [48] model assumes that pairs of nodes are independent; p_2 [65] assumes node pair independence conditional on attributes of the nodes. Generalizing these models, Frank and Strauss [34] introduced the p^* that makes the Markov assumption that two links are conditionally dependent if they share a common node (end-point).

Different functions $f(G)$ can be used to express density, reciprocity, transitivity and other network effects. Given the functions, the parameters need to be estimated. Exact computation of the maximum likelihood is normally intractable because of the normalization term Z, so Markov chain Monte Carlo (MCMC) estimation [96] is often used. Often analysts are interested only in the parameter values as measures of the various effects.

Hanneke and Xing [46] extend the model to use temporal data. Their model takes the Markov assumption that all of the historical information necessary for prediction at time t is captured in the previous time period $t-1$:

$$p(A^t | A^{t-1}, \theta) = \frac{1}{Z(\theta, A^{t-1})} e^{\theta^T \Psi(A^t, A^{t-1})} \tag{2.3}$$

where A^t is the adjacency matrix at time t, $\Psi(A^t, A^{t-1})$ is a set of k functions $\Psi : \Re_{n \times n} \times \Re_{n \times n} \rightarrow \Re^k$ and $\theta \in \Re^k$. Because the definition of the model contains a function of A^t it cannot be used to directly predict A^t given only A^{t-1} and θ but the authors show that it can be used for classification using an EM type of approach.

Dependency networks are another graphical model that *approximate* the joint distribution of random variables over an undirected graph:

$$p(X) = \Pi_{i=1}^{n} p(X_i|N(X_i))$$

where $N(X_i)$ is the neighbors of X_i. An extension of this model, relational dependency networks, define a general dependency around attributes of nodes of the same type.

The latent group model [78] (LGM) describes a joint distribution over nodes, links, attributes and groups. This model assumes a generative process where groups are assigned types from a multinomial probability, objects are assigned to groups uniformly, given a class and choose attributes (both using multinomial distributions and that objects are linked to each other according to a Bernoulli probability based on their group type.

The original p* model was designed to be used only with topological metrics like density and reciprocity. To measure selection [94] and influence [95], the model was modified to use attributes. These new models can measure the occurrences of network structures involving two and three nodes with common attributes that suggest the presence of selection and influence. A similar approach [88] is to use the same sort of metrics with a continuous time Markov chain model.

Discriminative models

Progress is often incremental. Later models are typically more expressive than earlier ones but it is enlightening to review the progression. One early model that is still relevant is the conditional random fields (CRF) [62]. Like hidden Markov models (HMM) [64], CRFs are used to label (Y) a sequence of random variables (X):

$$p(y|x) \propto \exp\left(\sum_{e\in E,k} \lambda_k f_k(e, y|_e, x) \sum_{v\in V,k} \mu_k g_k(v, y|_v, x)\right)$$

where x is a data sequence, y a label sequence, $y|_S$ is the set of components of y associated with the vertices in subgraph S, and V and E are the nodes and edges of the chain. The generalized functions f and g are used instead of the constant transition probabilities of HMMs. The parameters λ and μ are learned from the data. Used primarily for sequential data like natural language processing or biological sequences, these models are unfortunately not suitable for network data that contains cycles.

Probabilistic relational models (PRM) [36] are a step towards more expressive models. These are designed around the concept of the relational data of a typical database where a node, x, can be related to other nodes and have attributes, $x.A$. The model uses the conditional probability $P(x.A|Pa(x.A))$, where the parents of an attribute ($Pa(x.A)$) are defined by the relationships of the database. A relational skeleton σ (or graph) is a set of nodes with some missing attributes. Given a relational skeleton σ, instance I of the network and a set of attributes for each class ($A(X)$), the distribution becomes:

$$P(I|\sigma) = \Pi_{x \in \sigma(X)} \Pi_{A \in A(X)} P(x.A|Pa(x.A))$$

For this distribution to be coherent the requirement that no variable $x.A$ can depend, directly or indirectly on itself means that like CRFs, PRMs cannot be used with cyclical graphs.

Extending the PRM approach, relational Markov networks (RMNs) [114, 115], are also described in a database setting. To model networks with cycles, RMNs make use of clique potentials in a way similar to their use in the Gibbs distribution. An instance I of a schema refers to a specific network with I.y, I.x, and I.r being the labels, attributes and links of the instance. The conditional distribution then is:

$$P(I.y|I.x, I.r) = \frac{1}{Z} \Pi_{C \in \mathcal{C}} \Pi_{c \in C(I)} \phi_C(I.x_c, I.y_c)$$

where $\mathcal{C}$ is the set of all cliques, ϕ_C is the clique potential for C and Z is a normalizing constant.

An alternative method of parameter estimation for the p^* model[4] converts the model using a logit function in terms of a specific link between nodes i and j:

$$logit(A_{ij}|G) = exp\left(\frac{p(A_{ij} = 1|G)}{p(A_{ij} = 0|G)}\right) = \theta f(G)$$

which eliminates the normalization term. Additionally, since this model directly learns the likelihood it becomes a discriminative model.

There have been several papers that use conventional flat (non-relational) classifiers [19, 47, 70, 89]. This approach do not necessarily introduce new models but instead

Figure 2.3: General Network Mining Process

describe unique ways to convert the data from a network structure into the typical table structure that most classifiers use for input.

2.3 Network Mining Tasks

Network mining techniques, in general, use the link (A), attribute (X) and, sometimes, community (X) information to make a prediction, select nodes or form groups (Figure 2.3). The techniques vary in the way they utilize the input. Generally, the goal is to learn $f(A, X, C)$ such that a specific function is optimized. This thesis is primarily concerned with the tasks of predicting the class of a node (collective classification) and links between objects (link prediction), however, for completeness, an overview of all the major network mining tasks is provided.

2.3.1 Collective Classification

Traditional classifiers make a simplifying assumption that the objects are independent of each other. Researchers have recently begun to take advantage of the clearly defined relationships (links) within networks to improve classification [14, 19, 70, 107, 109, 125]. In the social network example above, the smoking status of a person can potentially be inferred from the smoking status of his/her friends. The challenge for collective classification is integrating the attribute and link data. Using the attributes of neighbors has been shown to actually be detrimental in some cases [19], however, using the class of the neighbor has been shown to be helpful [19, 70]. A related challenge is to recognize and utilize the structures inherent in the network. The study by Yang et al. [123] identified the existence of certain regularities in networks. For example, some networks exhibit

encyclopedia regularity where nodes of one class link to nodes of the same class.

Formally, collective classification is the task of assigning a node $v \in \{1, ..., n\}$ to one its predefined classes in Y. Learning in collective classification is the process of estimating a function f that maps a node in the network to its corresponding class, i.e., $f : \aleph \times \{1, ..., n\} \rightarrow Y$. The inputs are the network N, which is characterized by the 3-tuple (X, A, C), and the index of the node to classify. In other words, $f(N, i)$ can be used to infer the class for a node i. To estimate the function f, the input data is divided into a training set consisting of nodes whose class is known, and a test set consisting of nodes whose class is unknown. The function f is trained to optimize a loss function, $L(f, y)$, using only the training data. The form of the loss function depends on the choice of learning algorithm (e.g., some algorithms are designed to minimize empirical risk, while others are designed to maximize the margin of the decision boundary).

A number of solutions use an iterative approach where an initial *bootstrap* prediction is made and then the predictions are iteratively refined until convergence is reached. These algorithms use traditional flat classifiers for local predictions. The iterative process then propagates the predictions throughout the network. The first work [19] classified web pages using the text of the page and the class of the pages neighbors. In the bootstrap step, the class is predicted using only the attributes. Then they iteratively update the classes using only the neighbor classes. In the iterative classification algorithm (ICA) of Lu and Getoor [70], two classifiers were trained, one on the attribute data and the other using neighborhood class statistics of neighbors. In the bootstrap, only the first classifier is used. During iteration, they use a linear combination of both classifiers. The Gibbs sampling (GS) [73] approach is similar to the previous approach except that it samples the local classifier for the best label and after many iterations, it assigns to the node the label to that was sampled most often.

Many of the models discussed in Section 2.2.2 have been applied to the problem of classification. Because of the cyclical nature of these graphical models, exact solutions are normally intractable, therefore approximate methods are used for parameter estimation and inference. Gibbs sampling [77], EM algorithm [46, 78], belief propagation [114] Markov chain Monte Carlo [46] and relaxation labeling [122] are methods that are used.

The problem of propagating the class of known nodes through a network to the nodes with unknown classes is analogous to the label propagation problem [125] in graph-based semi-supervised learning literature. Unlike link-based classification, the graph used for label propagation is constructed based on the attribute similarity between objects. However, like some of the other collective classification methods it is an iterative process.

Collective classication can be extended to make use of the temporal information of an evolving network. In the work by Sharan and Neville [109], a framework is proposed for predicting attributes in evolving networks. In the first, graph summarization, step, weights are assigned to edges based on age of the link using a given kernel (under the assumption that newer links are more predictive than older ones). In the second step, attributes are predicted using relational classifiers that are extended to account for weighted edges.

2.3.2 Link Prediction

The link prediction problem [1, 47, 55, 68, 89, 90, 92] can be stated as follows: Given a network, can we infer the node pairs that are likely to be linked together? Link mining techniques are applicable to static networks (to infer missing links in an incomplete network) or temporal networks (to predict new interactions that will occur in the near future). Examples of link prediction problems include detecting covert ties between criminal suspects or identifying future collaboration between researchers.

Formally, link prediction is the task of assigning a node-pair (v_i, v_j) to a binary class $\{0, 1\}$, where 1 means there is an edge between the node-pair and 0 means otherwise. Learning in link prediction is the process of estimating a function f that maps the node pair into a numeric valued output score, i.e., $f : \aleph \times \{1, ..., n\}^2 \rightarrow [0, 1]$. The output scores can be sorted in decreasing order to rank the node-pairs based on how likely they are to become linked. The function f is trained to optimize a loss function, $\sum_{i,j} L(f(N, i, j), a_{ij})$, where a_{ij} is the true class of the node-pair obtained from the complete adjacency matrix of an incomplete network or a future network.

Link prediction is a challenging problem due to the sparsity of many networks. Predicting which non-linked node pairs will become linked has yielded very low accuracies [68]. Rattigan and Jensen [92] have shown that this is due to the skewed class

distribution—as networks grow and evolve, the number of non-linked pairs increases quadratically while the number of linked pairs often grows only linearly. Research in social sciences has suggested the tendency of individuals to establish friendship ties with others who have similar interests (attributes) [51]. In addition, individuals may also become friends because they share common friends (link structure) or belong to similar groups (communities). Integrating these different sources of information to improve link prediction is a challenge.

Liben-Nowell and Kleinberg [68] compared a large number of graph metrics as link predictors. They tested the metrics on bibliographic data sets using only the link structure and ignoring the node attributes. This work has been expanded to include both link and attribute data [47] by using binary classifiers.

The graphical models can be tuned to predict links, where the goal is to learn the joint probability density of the nodes, links, subgroups, etc., and to predict the missing links by applying Bayes theorem [78, 115]. Because of the sparsity of networks, Rattigan and Jensen [92] proposed a variation to the problem known as anomalous link discovery, where the goal is to find links that are anomalous. Recent works have also considered the changes in the network over time. Potgieter et al. [90] combined the metrics from the Liben-Nowell study with temporal metrics such as return, moving average and recency. In another work by O'Madadhain et al. [85], a time evolving probabilistic classifier is constructed from training data sampled over many time periods. Hanneke and Xing's [46] extension of the Exponential Random Graph Model accounts for the evolution of networks over time. A recent study by Backstrom et al. [5] on the evolution of communities in large social networks suggested that community structures and link formation are closely related.

Another approach considers each node pair as a sample with a binary class: 1 for linked and 0 for unlinked. Any existing classifier can then be trained and used to predict unlabeled samples [47, 89]. Features can be created by aggregating the paired attributes [47], using topological metrics [47] or by using the results of database queries [89]. This approach works for missing links, where the training and test sets are clearly defined and separated. For predicting future links, however, there is the problem that all the pairs are

in both the training and test sets.

Recent advances make use of temporal data. Kashima and Abe [55] proposed an approach that uses historical, topological data. Their model uses and EM-like algorithm to find a stationary state of the network using edge label functions. During the iterative process, as edge labels are changed, they also influence the edge labels of shared nodes. In another direction, Lahiri and Berger-Wolf [63] predict links by learning the probability distribution functions for pairs of subgraphs. Their algorithm considers the evolving network to be a data stream and extracts only frequent closed subgraphs.

In another interesting extension, by Bilgic, et al. [14] the authors propose combining the tasks of link prediction and collective classification. Given a training network, their algorithm first learns the parameters for a given link predictor and given collective classifier. Then, looping until convergence is reached, it alternates between predicting the missing labels of nodes and missing links.

2.3.3 Influence Maximization

The technique of influence maximization (also known as *diffusion of innovation*), is important in the areas of epidemic spread and viral marketing. The goal is to find influential nodes – nodes that will spread their influence quickly through the network. Influence is assumed to spread using a particular model of *diffusion*. In these models, nodes become activated (contracted a virus or bought a product) and can, in turn, activate their neighbors.

To formally define influence maximization it is necessary to first define a function h that maps a set of nodes $S = \{v_1, ..., v_k\}$ to a real number, i.e. $h(S) \rightarrow \Re$, such that $h(S)$ is the expected number of nodes activated by first activating the nodes in S. Influence maximization, then, maps a network and a number to a set of nodes, i.e. $f : \aleph \times \{1, ..., n\} \rightarrow \{1, ..., n\}^k$. The input is the network $N = \{X, A, C\}$ and the size of the initial set of nodes so that $f(N, k)$ finds the best k nodes to initially activate in order to maximize the influence. The function f is learned such such that $h(f(N, k))$ is maximized.

The underlying diffusion models include the families of threshold and cascade models. In the threshold models [44] a node becomes activated when a certain percentage of its

neighbors become activated. Newly activated nodes under the cascade models [42] have a one-time chance to activate neighbors with a given probability. Most of the models are probabilistic in nature. Without probability (e.g. if nodes are activated with certainty) every graph component with an activated node would end up with all nodes activated. Using appropriate probabilities ensures that activated nodes will only activate some of their neighbors and that the spread will stop before the entire network is activated.

The problem then is to choose nodes that will maximize a particular utility function. The most apparent utility function is the spread of activation to as many nodes as possible. For example, in viral marketing, a company may want to offer a small number of free or discounted products to influential people in the hopes that they will inspire others to purchase the product.

One might first consider activating only the highest degree nodes to obtain the optimal solution. However, one can quickly imagine that if the high degree nodes are all neighbors, the spread of influence will be less than if lower degree nodes, more spread out, were chosen. For example, in Figure 2.4, to maximize the number of nodes activated, the selected node is likely the best choice even though it is not the highest degree. Another challenge is that the link information may not be reliable – for example, in an online network, links between users are easy to add but do not always reflect genuine friendship. Furthermore, given the size of many real-world networks, simulating the activation process repeatedly to find the optimal solution is computationally expensive.

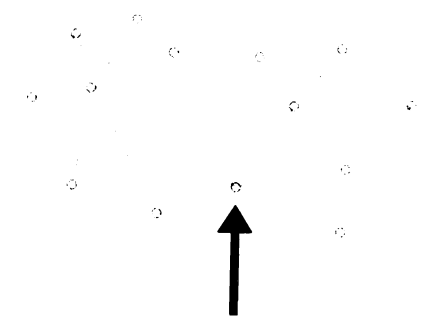


Figure 2.4: Choosing a node to maximize influence

Kempe et al. [59] showed that the problem is NP-complete under the specific diffu-

sion models of Independent Cascade and Linear Thresholds. They then propose a greedy strategy based on submodular functions [76], which guarantees a solution that is provably within 63% of optimal for these same models. In their experiments, the greedy strategy always performs better than the alternative strategies of selecting the nodes with the highest degree or lowest closeness scores.

Domingos and Richardson [30] proposed a cost/benefit approach to the influence maximization problem. They assume there is a cost for activating nodes and a revenue associated with activated nodes. The problem then becomes choosing a subset of nodes to activate that will maximize the expected lift in profit (i.e., revenue minus cost). A solution to the influence maximization problem in the face of competition was proposed by Bharathi et al. [12]. For example, multiple companies with similar products may attempt to influence the buying decisions of a targeted group of customers. Extending influence maximization to dynamic networks, where nodes may join or leave the network, is another open research problem. Variations of the problem in dynamic networks include finding the nodes that are most influential for new nodes or identifying the nodes whose influence spread is increasing or decreasing.

2.3.4 Community Finding

The technique of community finding[18, 40, 41, 79–81, 91, 93, 113], also called group detection [38], positional analysis [31, 118] or blockmodelling [118], is the process of placing nodes into groups based on a common set of properties. Similarly, clustering [112] and graph partitioning [52] form groups in such a way that the nodes within a group are "similar" to each other and "dissimilar" to nodes in other groups. From the graph theory perspective, the problem of community finding is to remove links from the graph so that the remaining graph has the "desired" components.

Let the set of all nodes be $V = \{v_1, ..., v_n\}$ and the set of communities be $C = \{C_1, ..., C_m\}$. Community finding can be formally defined as the task of assigning the nodes from V to one of the subsets $\{C_1, ..., C_m\}$ where $C_1 \cup C_2 \cup ... \cup C_m = V$. Learning is the process of estimating the function f that maps the network to a set of communities, i.e. $f : \aleph \rightarrow 2^{nm}$. For disjoint communities, $C_i \cap C_j = \emptyset$ for $1 < i < n, 1 < j < n, i \neq j$,

however for overlapping communities this restriction does not apply. Thus $f(N)$, where C is not specified, will return the matrix C. The loss function $L(f, C)$ is optimized in learning f.

Community finding is ill-posed; there is no agreed-upon metric for evaluation. Metrics that are commonly used can be separated by into supervised, where the original community assignments are known, and unsupervised, where they are not known. Supervised metrics, such as purity, entropy and normalized mutual information, measure – in different ways – how well the "found" communities reflect the original communities. Unsupervised metrics generally measure the cohesion (the similarity of the nodes within communities) and/or the separation (the distance between nodes from different communities). A recently proposed unsupervised graph-based metric from Newman and Girvan [81], called modularity, is based on the fraction of links within a community to those between communities. An additional challenge to community finding is scalability. Networks such as the World Wide Web or online social networks can have millions, even billions, of nodes. Using an agglomerative hierarchical algorithm with a complexity of $O(n^3)$ – where n is the number of nodes – can be infeasible.

The most common approach to community finding is to segment the entire network into disjoint groups, where each node is assigned to exactly one community. Traditional clustering algorithms such as k-means, DBScan, Chameleon, etc. [112] can be applied to generate such communities. Graph partitioning algorithms are also applicable. For example, spectral clustering [110] divides a network into balanced components based on the eigenvalues of its Laplacian matrix. This approach is equivalent to finding a partitioning that minimizes the normalized cut criterion [29]. Karypis and Kumar [53] developed a multi-level graph partitioning approach that can accommodate different heuristic functions for coarsening, partitioning and refining the clusters. While these algorithms were not specifically designed for networks, their application is straightforward. An approach that was specifically designed for networks, from Girvan and Newman [41], uses the edge betweenness metric to remove edges iteratively. It is intuitively appealing since high betweenness edges would appear to be bottlenecks between communities; however, it is slow [50, 91].

A variant of finding disjoint communities is to discover a hierarchy of communities. This approach allows the communities to be nested and organized as a tree structure called a dendrogram. Agglomerative hierarchical clustering methods such as single-link and complete-link can be used to find hierarchies in networks. More recently, Clauset et al. [22] proposed a method of extracting hierarchies based on maximum likelihood methods and Markov chain Monte Carlo sampling.

More recently, progress in community finding has focused along several directions. Semi-supervised learning methods have become popular in the clustering literature, where side information is available in the form of constraints on pairs of nodes that should or should not be grouped together. The side information can be obtained from the similarity between node attributes, partial knowledge of the class labels, etc. The side information may improve community finding in many ways—to aid in the cluster initialization, to guide the clustering process toward finding better partitions, and to learn the appropriate distance metric consistent with the domains expectations [9, 20, 37]. Another trend is finding communities in dynamic networks, where the nodes, links, and attributes change over time. Backstrom et al. [5] studied how the structural features of communities affect how nodes join and leave communities. A paper by Tantipathananadth et al. [113] proposed a new framework for tracking community changes in dynamic networks by modeling it as a graph coloring problem. Communities are identified by approximately solving a combinatorial optimization problem using dynamic programming.

2.3.5 Node Ranking

Ranking is the process of creating a total ordering of the nodes in a network. The rank of a node reflects the measurement of some particular structural property of the network, with respect to the node, which conveys a semantic meaning such as importance, popularity, authority, etc. As an end in itself, rankings can also be used to look for well-connected or *central* nodes in a network.

Formally, the task of ranking assigns a real number to a node $v \in \{1, ..., n\}$. Learning in ranking is the processing of estimating the function f that maps a node in the network to its corresponding score: $f : \aleph \rightarrow \Re$. The input is the network N represented as a

3-tuple (X, A, C) and the index of the node to be ranked. Thus, $f(N, i)$ will infer the relative rank for node i. A common approach for learning f is to use a random walk on the network. Given a transition matrix M where M_{ij} is the probability of moving from state i to j and given a vector A of scores, then the characteristic equation $A' = MA'$ can be solved using the power method where A' will be the dominant eigenvector. By substituting the normalized adjacency matrix for M then the values of the eigenvector A' can be thought of as a vector of authority rank representing the fraction of time that the random surfer would spend on each node.

In network mining, ranking can be done using *centrality* measures [118]. The first, *degree* centrality, is simply the degree of the node – in directional graphs it can be indegree or outdegree. *Closeness* centrality is the average shortest distance between a node and all other (reachable) nodes in the network:

$$closeness(v) = \frac{1}{|V| - 1} \sum_{u \in V \setminus v} d(v, u)$$

where $d(v, u)$ is the shortest distance from v to u. Lower values of closeness indicate a more centrally located node. Another centrality measure is *betweenness*, which is the number of shortest paths between all pairs of nodes that go through it:

$$betweenness(v) = \sum_{s \in V, t \in V, s \neq t \neq v} \frac{g_{st}(v)}{g_{st}}$$

where g_{st} is the number of shortest paths from s to t and $g_{st}(v)$ is the number of shortest paths from s to t that go through v.

A popular ranking method for large directed networks like the World Wide Web is the *eigenvector* method [60, 87]. In this method a node's rank is the sum of the ranks of its incoming neighbors. Given a network with n nodes and an adjacency matrix A where A_{ij} is 1 if there is a directed edge from i to j and zero otherwise, for the node v_i, the rank r_i is defined as:

$$r_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} r_j$$

where λ is a constant. Written in matrix form it becomes the eigenvector equation $\lambda r = Ar$. The dominant eigenvector of A provides an effective measure of authority rank. Google's PageRank is an example of this ranking method.

2.4 Summary

Networks are ubiquitous structures in the physical world. To understand them, it is necessary to identify their components (nodes, attributes, links and communities), to analyze their types (random, small world, etc.) and to examine their forces (e.g. influence and selection). Models can also be used to provide a probabilistic framework over the components of a network. The area of network mining uses a number of techniques to find hidden knowledge within these complex structures. Link prediction and collective classification are used to predict missing or evolving data. Hidden communities can be found using the technique of community finding. And influence maximization and ranking are used to assign a value to a nodes ability to influence or their authority within the network.

Chapter 3

Relationship between Link and Community Structure

3.1 Introduction

This chapter begins with an exploration into the relationship between links and communities. Next, roles are defined for nodes according to their relationship to communities. Finally, a new metric is proposed for nodes that captures their community belongingness information without knowing the actual communities.

Many existing community finding algorithms implicitly assume that the communities are compatible with the link structure of a network. Intuitively, this is supported by our familiarity with social networks where groups of people who are interconnected by friendship can be considered to be a community. The communities that are aligned with the link structure are considered to be *natural communities* of the network. In practice, however, a network can have communities imposed by other factors beyond its link structure. For example, with social networking sites, members can join special interest groups independent of the friends they have established. If the communities and link structure are incompatible, it would be meaningless to apply network mining on the network to learn characteristics of the non-natural communities. To overcome these limitations, a pair of new statistics for measuring the alignment between communities and link structure is proposed.

For the natural communities of a network, the notion of community-based node roles is introduced. Roles, which measure the function that nodes play with respect to the network, can convey meanings such as authority, popularity, and influence. The node role definition in this work is the first that we are aware of that makes use of community knowledge. An example is offered to illustrate how community-based roles can extend current methods of influence maximization.

Community-based roles can only be assigned if the community information is known, which it is often not. Therefore a new metric called rawComm is presented to estimate the number of communities to which a node is attached. It is shown that, if communities are aligned with the link structure, our estimate will be quite accurate.

The remainder of this chapter is organized as follows. After a preliminary discussion in Section 3.2, Section 3.3 describes the proposed metrics for measuring the level of alignment between link structure and communities. This will be followed by the descriptions of our rawComm metric and node role definition in Section 3.4. It is then illustrated how the community information can be applied to the influence maximization problem in Section 3.5. Finally, experiments to are presented to support the findings and conclusions to summarize our work in Sections 3.6 and 3.7.

3.2 Preliminaries

Before proceeding with the main contributions of this chapter, it is helpful to review some background information on communities and community finding algorithms. The concept of a community refers to nodes that are grouped together based on a common set of properties. In social networks we think of friends, family, and colleagues as the basis of communities.

If the communities are not given, they can often be created. Many clustering and community finding algorithms have been proposed in recent years (see Section 2.3.4). The goal of these algorithms is to group together the nodes so that they appear to have a common set of properties. With networks, there can be several sources of information for guiding the formation.

Some clustering algorithms use the attribute similarity only to find clusters that maximize the intra-community similarity and minimized the inter-community similarity. Others, such as graph partitioning and spectral clustering algorithms, use only the links. These separate the network into groups where the intra-group links are maximized and the inter-group links are minimized. There are some methods that use both the links and the attributes, as with non-negative matrix factorization [124]. There are clustering algorithms [99] that use other information such as must-link and cannot-link edges to form pure and complete clusters.

For the methods that use links, if the algorithms are successful, the communities that are found will be aligned with the link structure. However, the degree of alignment may vary depending on the algorithm used. And for those that do not use links, there is certainly no guarantee that the communities will be well aligned with the links.

Furthermore, in many applications, the communities could be defined based on other external criterion. As an example imagine a social networking site where members can join special interest groups. Let us further assume that the links between members are based on friendship whereas members join groups based on other criteria (such as political leaning, age group, favorite music, etc). It will be shown in Section 3.3 that often these externally imposed communities are not in alignment with the link structure. Therefore it is useful to measure the alignment between the network and the community structure one has in mind.

3.3 Measuring Community and Link Structure Alignment

This section will present a means to measure the alignment and thus provide a basis for judging the effectiveness of the community finding algorithm.

3.3.1 Modularity Measure

There have been very few attempts at measuring the compatibility between the communities and the link structure. Newman and Girvan [81] proposed a modularity function (Q) for estimating this compatibility. Q compares the difference between the fraction of

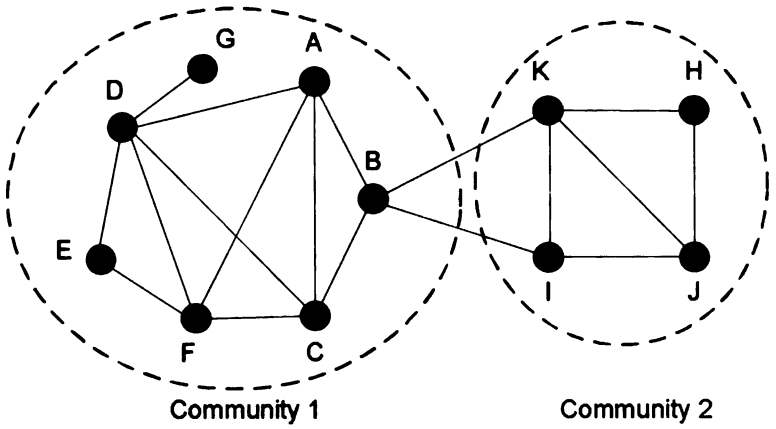


Figure 3.1: Communities within a network

links that occur within the communities to the fraction that would be expected to occur if the links were randomly distributed. Although the measure seems intuitive, it has several drawbacks. First it is sensitive to the number and size of communities. Thus Q is not an appropriate metric for comparing the compatibility of communities in two networks. Another limitation is that Q does not explicitly consider the non-links between nodes in the same community. As a result, it may underestimate the degree of compatibility in some networks and overestimate the compatibility in others.

Given a network of k communities, modularity is defined as $Q = \sum_{i=1}^{k} (e_{ii} - a_i^2)$ where e is a $k \times k$ matrix whose element e_{ij} is the percentage of links that connect nodes in community i with community j. a_i is the sum of row i for e, which is the fraction of edges with at least one end in community i. In a network where links are placed randomly $e_{ij} = a_i a_j$. So, a_i^2 is the expected fraction of edges within community i if the edges were distributed randomly. Values of Q close to one indicate strong alignment between link and community structure whereas values close to zero indicate no alignment.

The modularity measure has several drawbacks. First, the values for Q are sensitive to the number of communities and number of nodes in each community. Q would be an adequate relative measure when comparing algorithms on the same network with the same number of communities but not for comparing one network to another or comparing different number of communities for the same network.

Another problem with Q is that it does not explicitly take non-links into consideration.

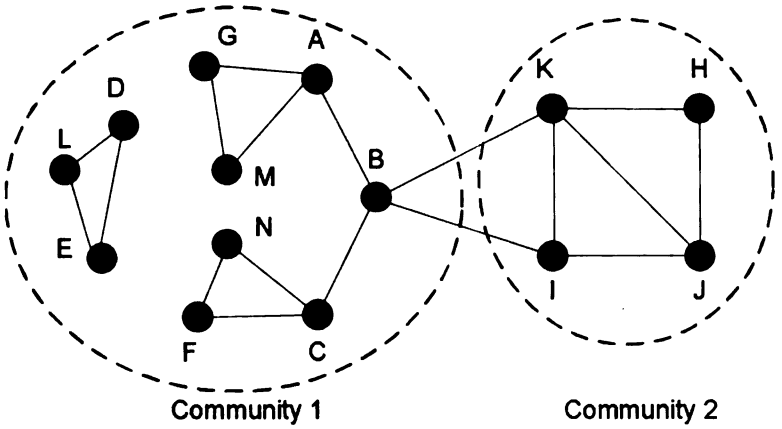


Figure 3.2: Communities with more non-links

For example, compare the network in Figure 3.1 to the one in Figure 3.2. They both have the same number of links and both have the same value for Q of 0.216 but the second has more non-links and clearly is not as cohesive as the first — thus Q does not reflect the reduction in alignment.

3.3.2 Alignment Measures based on Completeness and Purity of Node Pairs

To measure the disparity between community membership and the groups implied by links, the well known concepts of must-link and cannot-link edges used in the semi-supervised and constraint-based clustering literature [9, 13, 26, 27, 99] were borrowed. A must-linked node pair is defined as either complete or incomplete and a cannot-linked node pair as pure or impure. A complete node pair is one where the linked nodes belong to the same community. Pure node pairs are non-linked nodes that do not appear together in any community.

For example, in Figure 3.1, it can be seen that the linked node pair (A,B) is within a community and therefore is complete, whereas the node pair (B,K) represents an incomplete node pair. The node pair (A,K) is non-linked and in separate communities making it pure, whereas the node pair (A,G) is impure. A disconnected network with communities that correspond to each disjoint clique will not have any impure nor incomplete node pairs.

It follows that a community structure with a minimum number of incomplete and impure nodes is compatible with the link structure. To measure the compatibility of link and community structure, we define the ratios p and q as:

$$\begin{aligned} p &= \frac{\text{Complete node pairs}}{\text{Total linked node pairs}} \\ q &= \frac{\text{Pure node pairs}}{\text{Total non-linked node pairs}} \end{aligned} \tag{3.1}$$

High values of p and q mean that the link structure provides solid support for the community structure. For example a p of .9 means that 90% of the linked edges are within the same community.

There are a number of differences between the p and q statistics and the modularity metric Q presented in the previous section. First the meaning of p and q are clear—p is the fraction of links within communities and can therefore be thought of as the probability of an edge being within a community. Similarly q is the fraction of non-edges between communities. Also the ranges are well defined; both can take values between 0 and 1.

This makes p and q appropriate statistics for comparing different algorithms, different networks or different community sets for a single network. Additionally, unlike the modularity metric Q, the q statistic gives us a measure of how pure the communities are.

Another advantage to p and q is that they more accurately capture the non-link information. Again, referring to Figures 3.1 and 3.2, as stated previously the Q value is the same for both as is the value of p (0.89). However, q=0.70 in the first network and 0.52 in the second. Thus, though Q by itself does not give an complete measure of the alignment, considering both p and q will.

One last difference between Q and p and q is the placement of the between-community (or incomplete) links. Referring to Figure 3.3 notice two networks with three communities. For both networks, $p = 0.33$ and $q = 1$ but for (a), $Q = -0.67$ and for (b), $Q = -0.59$. It appears that as the incomplete links are less uniformly distributed the farther Q will be from zero but the interpretation is not clear. Finally, Q has an advantage over p and q in that it is a single metric.

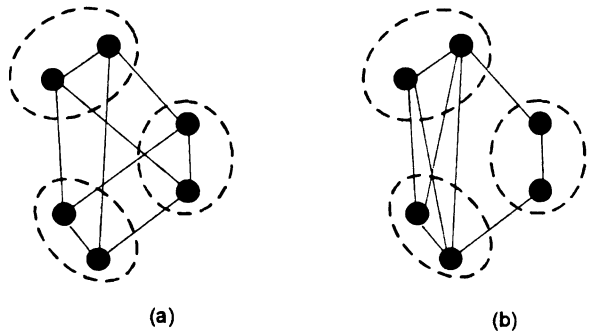


Figure 3.3: Placement of incomplete links

3.3.3 Application of Community-Based Alignment Measures

As discussed above, many network analysis applications yield community information that is in disagreement with analysts' pre-conceived notions of community The alignment measures p and q described above provide a powerful method for understanding the extent of this disagreement.

Another application would be to measure p and q after a community finding algorithm is applied to a network to judge the effectiveness of the algorithm. If the statistics are high this suggests that it would be consistent with a user's expectations to perform network analysis using the link structure.

In a later section p and q will be used to define a metric to estimate community attachment when the communities are not available.

3.4 Community-Based Node Roles

Network mining techniques have largely ignored the potentially helpful knowledge of community and community attachment in their analysis. In this section the concept of community-based roles is introduced and a new metric called rawComm for estimating the number of communities connected to a node is presented.

3.4.1 Node Roles

In social network analysis, role is used to describe the behavior of a node in relationship to its neighbors and to the network at large. Examples of node roles include those that are based on their popularity, centrality and authority. Metrics are often used to determine the roles of nodes in a network. Of these the most prominent are degree, closeness and betweenness [118].

Degree is the sum of the links attached to a node. Closeness is the reciprocal of the sum of all the geodesic (shortest) distances from a given node to all others: $C_C(n_i) = \left[\sum_{j=1}^{N} d(n_i, n_j)\right]^{-1}$ where $d(u, v)$ is the geodesic distance from u to v. Nodes with a small C_C score are closer to the center of the network while those with higher scores are closer to the edge. Betweenness is defined as: $C_B(n_i) = \sum_{j<k} \frac{g_{jk}(n_i)}{g_{jk}}$ where g_{jk} is the number of geodesic paths from j to k and $g_{jk}(n_i)$ is the number of geodesic paths from j to k that go through i. A higher betweenness value for a node means that it is on more shortest-paths between nodes, which is an indication of the node's importance.

Hubs and authorities are two other important roles, particularly in web search. These roles are defined in terms of rank-based metrics provided by algorithms such as PageRank [87] or HITS [60]. They are applicable to directional networks where a node's authority is based on a summation of authority on its incoming links.

3.4.2 Community-Based Node Roles

Community-based roles are useful in a number of ways. First, they provide useful information to analysts in areas as such as anti-terrorism and law enforcement. In searching for potential terrorist threats, for example, analysts may find it useful to identify suspects with certain roles (mastermind, financier, facilitators, military commander, etc). If they were looking for persons with diverse contacts, they could focus on nodes whose community-based roles are designated as bridges or ambassadors (see Figure 3.4). Second, community-based roles could also be utilized in existing link mining applications such as ranking, link prediction, influence maximization, and node classification. An example illustrating the application of community-based roles to influence maximization is

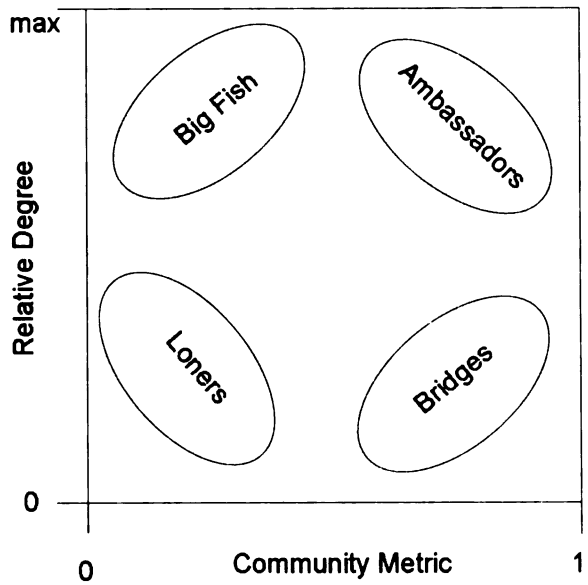


Figure 3.4: Community-based roles

given in Section 3.5.

We define the community-based role of a node according to the number of communities and links incident to it. Figure 3.4 shows a community-degree chart that is divided into four quadrants for the four different roles. The vertical axis represents the degree while the horizontal axis represents the community metric.

The community-based node role is identified based on which of the four quadrants a node falls into. Nodes in the upper right quadrant are those with a high degree and a high community score. They act as *ambassadors*, providing connections to many different communities. The upper left quadrant contains what we call *big fish* from the cliche "big fish in a small pond" meaning that they are very important only within a community. This is due to their having a high degree but a relatively small community score. In the lower right quadrant are those with a low degree but a high community score. These we call *bridges* because they serve as bridges between a small number of communities. Finally, in the lower left are the *loners*—those with a low relative degree and low community score.

The metrics shown in the community-degree chart have been normalized to values between 0 and 1. For the community metric, the minimum is subtracted and the result divided by the range between maximum and minimum. The degree is divided by the highest

degree node in the network, giving a relative degree score between 0 and 1. In the experiments, a threshold of .5 was chosen to classify the node roles; however, depending on the distributions of degree and community metric scores, other thresholds can be chosen.

In order to define the community-based node role it is necessary to measure the number of communities linked to each node. If the community membership information is available, this can easily be done. However, often it is not available, in which case a method is needed to infer it from the network. The next section presents our proposed community metric known as rawComm.

3.4.3 rawComm Metric

To understand the intuition for the rawComm metric, consider the diagram shown in Figure 3.5. There are 9 nodes from three communities in the neighborhood of node A. Nodes B through E are attached to one community, F and G are attached to another, and H through J are attached to a third community. Another way of computing the number of communities connected to A is to add up the *community membership contributions* from each of its neighboring nodes. For example, since B is in a community of four nodes (excluding A), its community membership contribution to A is $\frac{1}{4}$. Similarly, F is in a community with one other node, so it is assigned a community membership contribution of $\frac{1}{2}$, and so on. When all of the community membership contributions from nodes B to J are added, the total is 3, which is exactly the number of communities to which node A is attached.

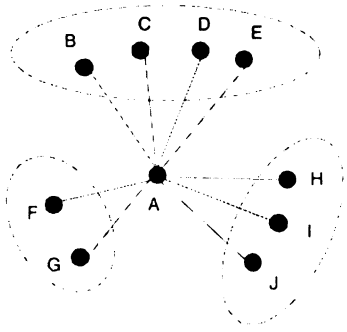


Figure 3.5: Community membership contribution of neighbors to rawComm of node A.

The method for computing the community metric, called rawComm can now be formalized. The community metric for the node i, is defined as follows:

$$\text{rawComm}(i) = \sum_{j \in N(i)} \tau_i(j) \tag{3.2}$$

where $N(i)$ is the set of nodes in the neighborhood of node i, and $\tau_i(j)$ is the community membership contribution of node j to the rawComm of node i. Following the above discussion, the community membership contribution $\tau_i(j)$ is defined as follows:

$$\tau_i(j) = \frac{1}{1 + C_{ij}} \tag{3.3}$$

where C_{ij} is the number of nodes (other than j) that are neighbors of i and are in the same community with j. For example, since B is in a community with three other nodes, $C_{AB} = 3$ and $\tau_A(B) = 1/4$.

Note that Equations 3.2 and 3.3 give the exact formula for computing the community metric for node i even though it is computed using only nodes that are adjacent to i. If the community membership of every node is known, then it is possible to compute the community metric precisely. Otherwise, it is necessary to estimate the parameters $\tau_i(j)$ and C_{ij} from the topology of the network. A probabilistic approach for estimating the expected values of these parameters are given in the next section. It will also be shown that the approximations become quite reliable when the notion of community is well-aligned with the network topology.

3.4.4 Estimating Community Membership Contribution

Let $\mathcal{G} = (V, E)$ be a graph and $v \in V$ is one of its nodes. The neighborhood of v, denoted as $N(v) = (V_v, E_v)$, is an induced subgraph of $\mathcal{G}$, where $V_v = \{u \in V | (u, v) \in E\}$ and $E_v = \{(u, w) | u \in V_v, w \in V_v, (u, w) \in E\}$. We use the following terms in our definitions and theorems:

G_v^*	set of all possible community assignments for the nodes in V_v
G_v	a specific community assignment, $G_v \in G_v^*$
$\|G_v\|$	the number of communities in G_v
Ω_v	set of all possible communities
g	a specific community
$\|g\|$	the number of nodes in community g
$P(G_v)$	probability of a specific community assignment G_v
$P(g)$	probability of a community g

As an example, let $V_v = \{a, b, c, d, e\}$ be the set of nodes adjacent to v in $\mathcal{G}$. The set of all possible communities is $\Omega_v = \{\{a\}, \{b\}, \cdots, \{a, b, c, d, e\}\}$. $G_v = \{\{a, b, e\}, \{c, d\}\}$ is a specific community assignment and $g = \{a, b, e\}$ is a community in G_v. Therefore, $|G_v| = 2$ and $|g| = 3$.

The method for estimating the community membership contribution τ is based on the following two assumptions. First, it is assumed that the community structure and network topology are well aligned. Based on our discussion in Section 3.3, this assumption implies that p and q are at least larger than 0.5. The second assumption is that the neighborhood of a node contains all of the information necessary to estimate the community membership contribution. Based on these assumptions, the next two theorems give the formula for computing the expected value of $\tau_v(u)$ using the neighborhood information $N(v)$.

Theorem 1 *Given a node v and its neighborhood $N(v) = (V_v, E_v)$, the expected value of the community membership contribution for node $u \in V_v$ is*

$$E[\tau_v(u)] = \sum_{g \in \Omega_v} \frac{P(g)}{|g|} \delta(u \in g) \tag{3.4}$$

where δ is an indicator function whose value is 1 if its argument is true and 0 otherwise.

Proof The expected value of rawComm for a node v is:

$$\begin{aligned} E[\text{rawComm}(v)] &= \sum_{G_v \in G_v^*} |G_v| \cdot P(G_v) \\ &= \sum_{G_v \in G_v^*} \sum_{g \in G_v} P(G_v) \\ &= \sum_{G_v \in G_v^*} \sum_{g \in \Omega_v} P(G_v)\delta(g \in G_v) \end{aligned}$$

where $|G_v| = \sum_{g \in G_v} 1$ is replaced on the second line. Note that the two summations can be interchanged because they are independent. Furthermore, a community g can appear in more than one community assignment G_v. Therefore, $P(g)$ can be computed from $P(G_v)$ as follows:

$$P(g) = \sum_{G_v \in G_v^*} P(G_v)\delta(g \in G_v)$$

So the expected value for rawComm is

$$\begin{aligned} E[\text{rawComm}(v)] &= \sum_{g \in \Omega_v} P(g) = \sum_{g \in \Omega_v} \sum_{u \in g} \frac{P(g)}{|g|} \\ &= \sum_{g \in \Omega_v} \sum_{u \in V_v} \frac{P(g)}{|g|}\delta(u \in g) \end{aligned}$$

since $|g| = \sum_{u \in g} 1$. Next the summations are exchanged so that the outer summation is over all the nodes instead of over all possible communities:

$$E[\text{rawComm}(v)] = \sum_{u \in V_v} \sum_{g \in \Omega_v} \frac{P(g)}{|g|}\delta(u \in g)$$

Since $E[\text{rawComm}(v)] = \sum_{u \in V_v} E[\tau_v(u)]$, the theorem is thus proven.

Theorem 1 states that the expected value of the community membership contribution of node u can be computed by the sum of probabilities of all communities containing node u weighted by the community size. To compute the right hand side of Equation 3.4, we need enumerate every g that contains u and compute $P(g)$ — a very expensive procedure. A more efficient approach is proposed by using the nodes in g that are linked to u. To this end a probability model is built that uses the links between u and the other nodes in V_v.

Let n_1 be the number of nodes in V_v that are linked to u and n_2 be the number of nodes in V_v that are not linked to u. Note that $n_1 + n_2 + 1 = V_v$. Also given a community g, let $X \sim BIN(n_1, p)$ be a random variable for the number of nodes in g linked to u and $Y \sim BIN(n_2, 1-q)$ be a random variable for the number of nodes in g not linked to u. Note that X and Y are independent of each other. With these definitions and the probabilities p and q from Equation 3.1 the first and second order approximations for $\tau_v(u)$ can be presented.

Theorem 2 *Given a node v and its neighborhood, $N(v) = (V_v, E_v)$, the estimated contribution of node u to v's community count is*

$$E[\tau_v(u)] = \frac{1}{1 + n_1 p + n_2(1-q)} \tag{3.5}$$

where n_1 is the number of nodes in V_v that are linked to u and n_2 is the number of nodes in V_v that are not linked to u.

Proof $\forall g \in \Omega_v$ where $u \in g$ there are $0 \leq k_1 \leq n1$ nodes in g linked to u and $0 \leq k_2 \leq n_2$ nodes in g not linked to u. Then it can stated that:

$$\sum_{g \in \Omega_v} \frac{P(g)}{|g|} \delta(u \in g) = \sum_{k_1=0}^{n_1} \sum_{k_2=0}^{n_2} \sum_{g \in \Omega_v} \frac{P(g)}{|g|} \delta(g, u, k_1, k_2)$$

where $\delta(g, u, k_1, k_2)$ is 1 for $u \in g$ and the number of nodes in g linked to u is k_1 and the number of nodes in g not linked to u is k_2; otherwise $\delta(g, u, k_1, k_2)$ is zero. Clearly $|g| = 1 + k_1 + k_2$ where the 1 is for the node u itself. The above equation is valid because the right side will sum $P(g)$ for every g that contains u since each g has a fixed k_1 and k_2 and the δ function insures that each g is selected exactly once. Now,

$$\sum_{g \in \omega_v} \frac{P(g)}{|g|} \delta(g, u, k_1, k_2) = P(X = k_1, Y = k_2)$$
$$= \binom{n_1}{k_1} \binom{n_2}{k_2} \cdot p^{k_1} (1-p)^{n_1 - k_1} (1-q)^{k_2} q^{n_2 - k_2}$$

As can be seen, this allows $P(g)$ to be expressed as a function of the number of linked and non-linked nodes. The expected value of the community membership contribution then is:

$$\sum_{k_1=0}^{n_1} \sum_{k_2=0}^{n_2} \binom{n_1}{k_1} \binom{n_2}{k_2} \cdot \frac{p^{k_1} (1-p)^{n_1 - k_1} (1-q)^{k_2} q^{n_2 - k_2}}{1 + k_1 + k_2}$$

This expression cannot be easily reduced to a formula but notice that it is $E(f(X,Y))$ where $f(X,Y) = \frac{1}{1+X+Y}$ so it can be estimated by taking the Taylor expansion of $f(X,Y)$ around μ_X, μ_Y:

$$\begin{aligned} f(X,Y) &= f(\mu_X,\mu_Y) + \frac{\partial f(\mu_X,\mu_Y)}{\partial x}(X-\mu_X) \\ &+ \frac{\partial f(\mu_Y,\mu_Y)}{\partial y}(Y-\mu_Y) \\ &+ \frac{1}{2}\frac{\partial^2 f(\mu_X,\mu_Y)}{\partial x^2}(X-\mu_X)^2 \\ &+ \frac{1}{2}\frac{\partial^2 f(\mu_X,\mu_Y)}{\partial y^2}(Y-\mu_Y)^2 \\ &+ \frac{\partial^2 f(\mu_X,\mu_Y)}{\partial x \partial y}(X-\mu_X)(Y-\mu_Y) \\ &\ldots \end{aligned}$$

Taking the expectation of the above expansion will be $\tau_u(v)$. Below the expected value of the first six terms is shown:

$$\begin{aligned} &E(f(\mu_X,\mu_Y) = \frac{1}{1+\mu_X+\mu_Y} \\ &E\left(\frac{\partial f(\mu_X,\mu_Y)}{\partial x}(X-\mu_X)\right) = \frac{\partial f(\mu_X,\mu_Y)}{\partial x}E(X-\mu_X) = 0 \\ &E\left(\frac{\partial f(\mu_X,\mu_Y)}{\partial y}(Y-\mu_Y)\right) = \frac{\partial f(\mu_X,\mu_Y)}{\partial y}E(Y-\mu_Y) = 0 \\ &E\left(\frac{1}{2}\frac{\partial^2 f(\mu_X,\mu_Y)}{\partial x^2}(X-\mu_X)^2\right) = \frac{\sigma_X^2}{(1+\mu_X+\mu_Y)^3} \\ &E\left(\frac{1}{2}\frac{\partial^2 f(\mu_X,\mu_Y)}{\partial y^2}(Y-\mu_Y)^2\right) = \frac{\sigma_Y^2}{(1+\mu_X+\mu_Y)^3} \\ &E\left(\frac{1}{2}\frac{\partial^2 f(\mu_X,\mu_Y)}{\partial x \partial x}(X-\mu_X)(Y-\mu_Y)\right) = \frac{\sigma_{XY}}{(1+\mu_X+\mu_Y)^3} \end{aligned}$$

Since X and Y are both binomial we have $\mu_X = n_1 p$, $\mu_Y = n_2(1-q)$, $\sigma_X^2 = n_1 p(1-p)$, $\sigma_Y^2 = n_2 q(1-q)$ and $\sigma_{XY} = 0$ because X and Y are independent. Taking the first three terms our first order approximation becomes

$$\tau_u(v) \approx \frac{1}{1+n_1 p + n_2(1-q)} \tag{3.6}$$

and taking the first six gives the second order approximation:

$$\tau_u(v) \approx \frac{1}{1 + n_1 p + n_2(1-q)} + \frac{n_1 p(1-p) + n_2 q(1-q)}{(1 + n_1 p + n_2(1-q))^3} \tag{3.7}$$

The second term on the right hand side of Equation 3.7 will often be small but not negligible. In tests the contribution was calculated using the two formulas against the true expected value. The probabilities p and q were varied from .5 to 1 and the number of linked and non-linked nodes were varied from 0 to 10. In the worst case using both terms the approximation was within 2% of the actual where just using the first term was within 20%.

The justification for using just the first few terms of the series follows: The variances of X and Y are $n_1 p(1-p)$ and $n_2 q(1-q)$ respectively. As stated earlier, it is assumed that p and q would have high values which means the variances would be low and so X would be close to μ_X and Y would be close to μ_Y. If high accuracy is demanded the approximation should use both terms but just using the first formula gives a close approximation that is adequate for relative comparisons. In the tests of rawComm for influence maximization the shorter version was used and in comparisons using both terms had no noticeable effect.

In order to apply Equation 3.6 or 3.7, we need to know the values of p and q for a given network. Unless otherwise mentioned, our experiments were conducted using $p = q = 1$. As will be shown in Section 5.2, even if the approximation is not very good we can still achieve good results for two reasons. First, we are mainly interested in communities that are consistent with the network links, which means that their p and q values should be reasonably high (i.e., no less than 0.5). Otherwise, we should not expect the metric, nor any community finding algorithms, to produce a view that agrees with the community concept we have in mind. Second, to define the type of community-based node roles (Figure 3.4), it is sufficient to know the relative ordering of their community scores rather than their absolute counts. As long as the metric does better than random guessing, we expect to see some improvements in the experimental results.

3.5 Application of Roles to Influence Maximization

It is assumed that the nodes in the network are capable of adopting an idea, purchasing a product or something similar. This process is referred to as activating. It is also assume that nodes that are activated have the ability to influence their immediate neighbors who themselves may choose to activate. The problem becomes choosing the best nodes to initially activate in order to maximize the number of activated nodes at the end of the process.

The problem of influence maximization can be thought of as finding the best k nodes to activate in order to maximize the number of nodes that will eventually be activated. Several algorithms [30, 59] have been developed in recent years to identify the most promising set of nodes to activate. These algorithms however focus only on maximizing the number of activated nodes at the end of the influence diffusion process. In some cases, it may be more useful to maximize the number of communities that are influenced. As an example, a marketer might be interested in not only informing as many people as possible about their product but might also wish to maximize their reach to different demographic groups.

Figure 3.1 shows a small network of eleven nodes from two communities—nodes A-G are attached to community 1 and nodes H-K are attached to community 2. Suppose that we wish to find the one best node in this network to maximize the spread of influence. Current algorithms would choose to activate node D (mostly due to its high degree). Depending on the influence diffusion model, activating node D often yields the largest number of activated nodes but the influence may not propagate to any of the nodes in community 2. Choosing node B, on the other hand, would elevate the chances that nodes in both communities are influenced.

In the paper by Kempe, et al [59] several models are introduced that describe the behavior of the node activation. In the experiments the Independent Cascade model was chosen. Under this model, influence is spread from node to node in discrete steps. A node i that becomes active in step t has one chance to make his inactive neighbors active in step $t+1$. The probability that node i will activate node j in their paper will be called the edge

weight.

The work in this area is exclusively concerned with maximizing only the raw number of nodes activated. However, it is proposed here to extend the problem to focus on the number of communities covered. A community is covered if one of the nodes in the community is activated.

The approach here will be to choose the initial set of nodes using the community-based node roles using two methods in order to maximize the communities covered. The first method selects those nodes with the highest rawComm score which focuses on ambassadors and bridges. The second method focuses exclusively on ambassadors by choosing nodes with a combination of high rawComm and high degree. The results of our experiments will show that using roles to maximize community coverage shows improvement over the other influence maximization methods.

3.6 Supporting Analysis and Experiments

This section presents the empirical evidence that demonstrates the usefulness of our proposed approaches. Section 3.6.1 provides a description of the data sets used. In Section 3.6.2 we examine the properties of community alignment measures (p, q, and Q) and the rawComm metric. Finally, Section 3.6.3 illustrates the advantages of using community-based node roles in influence maximization problems.

3.6.1 Data Sets

The experiments were conducted on three data sets—movie data from UCI KDD repository, Enron email data, and the Facebook social networking data. For the movie data set, a link was created between actors who have co-starred in at least one movie together. The resulting network had 3,725 nodes and 58,123 links. The Enron data set is a collection of email messages exchanged among 149 executives at Enron. A network was formed by establishing links between executives who had more than five email exchanges between them. The FaceBook data set was created from crawling the FaceBook web site for a medium sized university in Michigan. FaceBook is a social networking site that allows

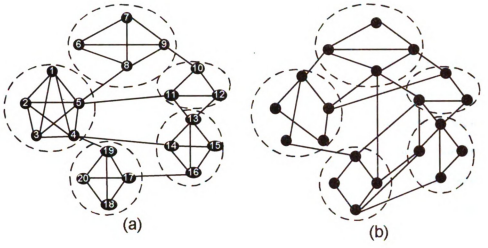


Figure 3.6: Effect of p on τ

students to join, post pictures, text and other descriptive information. Most importantly they can create links to friends and join groups. The resulting network contains 2,550 nodes.

In addition to the three real data sets experiments were also run on the synthetic networks shown in Figure 3.6. Both networks contain the same node set and communities but different link structures. The communities were purposely fixed so that one could clearly see the difference in the alignment between the link structure in (a) versus (b).

3.6.2 Community and Link Structure Alignment

The goals of the experiments in this section are to show that:

- p and q provide an effective measure of the extent to which communities are aligned with the link structure
- p and q often provide more meaningful information about the alignment than modularity (Q)
- rawComm provides a reliable estimate for community metric when p and q are sufficiently high.

For each of the three real data sets, the values for p, q and Q were calculated for two types of community definitions: (1) based on some externally imposed criterion and (2)

based on the results obtained using the normalized cut (nCut) spectral clustering algorithm [110].

For the FaceBook data three types of communities were created, each containing 20 communities. The first community type groups the students by a combination of gender, age and political party preference. Next, the students are grouped by their hometown (19 most popular and other) and after that, by their concentration (or major). For Enron, the employees were grouped by title (director, vice-president, etc). The movie data set was grouped by genre (actors were placed in genres based on the movies they were in).

Table 3.1: Alignment of Communities with Link Structure

Grouping method	Nbr of comm.	p	q	Q	SSE	Avg error	Avg comm.
FaceBook							
age, etc.	20	0.12	0.91	-.19	53k	3.70	10.95
hometown	20	0.58	0.46	-.31	36k	2.89	7.47
major	20	0.59	0.49	-.25	32k	2.67	7.06
nCut	20	0.44	0.93	.23	17k	2.01	9.15
Enron executives							
title	12	0.16	0.87	-.28	1124	2.25	4.55
nCut	12	0.61	0.94	.36	256	0.99	3.07
Movie actors							
genre	14	1.00	0.16	1.11	183k	6.64	11.01
nCut	14	0.75	0.70	.23	35k	2.13	2.73
Synthetic							
net (a)	5	.79	1.00	0.48	1.31	0.25	1.75
net (b)	5	0.59	0.93	0.19	11.78	0.53	2.20

Comparison between Community Alignment Measures

To see how p and q can be used to characterize the alignment of link and community structure, consider the results shown in Table 3.1. This discussion will concentrate on the p, q and Q columns of this table. Observe that communities defined using nCut are often more compatible with the link structure than communities defined using external criterion. This is not surprising since nCut uses the link structure to create its communities.

For FaceBook, observe that the community formed using a combination of age, gender, and political leaning has a very low value of p (almost 90% of the links are between

communities) but a high value of q (more than 90% of non-links are between communities). This means that two friends are almost certain to be in different communities based on age/gender/politics but also that two non-friends are also very likely to be in different communities. It can be concluded that these communities are not aligned well with the links but the non-links are aligned. For the next two community definitions—hometown and major—the alignment is better for p but for q it is about 50/50. All three externally defined community types have very poor alignment but the p and q values are very different for age/gender/politics compared to hometown and major. Yet, the value of Q is about the same for all three. Thus, even if the alignment is poor, p and q allows us to distinguish between data sets where Q does not. As another example, note that the nCut communities for both FaceBook and movie had a Q of 0.23. This would imply that the alignment between the communities and the link structure was of a similar nature but as we can see by looking the the p and q values this is not the case. As mentioned in Section 3.3, the magnitude of Q may not be meaningful since it is sensitive to the number of communities and number of nodes in each community. Instead it is more suitable as a relative measure for comparing different alignments on the same network with the same number of communities.

The Enron data has a high q and low p for communities defined based on the executive's title. From this it can be surmised that executives are very likely to email other executives outside of their community but also that executives who do not email each other are most likely from different communities. For movies, the links between actors appear to be good predictors for communities defined according to movie genre. This is partly because the communities are overlapping (in the technical report [101] it is explained how p and q are modified to handle overlap). Since the q value is low, this implies that even if two actors did not appear together in a movie they will still likely work in the same genre.

For the synthetic data, observe that both networks have fairly high values for both p and q but that (a) appears to be better aligned than (b). Visually inspecting the networks confirms this. The Q value also supports this but does not let us know why. We can tell from p and q that it is the links (p) that cause the misalignment and not the non-links (q).

Effect of alignment on rawComm

The last three columns of Table 3.1 correspond to the following:

- **SSE**. This represents the sum-squared error (SSE) between the number of actual communities linked to each node and the number predicted by rawComm.
- **Avg error**. This represents the average error (average of the absolute value of the difference between the actual number of communities and rawComm) of all nodes.
- **Avg Comm**. This represents the average number of actual communities assigned to the nodes by each of the community definition methods.

For each data set, the average SSE found by each grouping method is compared. The smaller the average SSE is, the more accurate our estimation of rawComm is. For FaceBook, nCut has the best SSE which is not surprising given that it has a high q value and an average p. Looking at the rest of the table it is clear that with better alignment (higher p and q values), the accuracy of rawComm improves.

The last two columns reveal another view of the accuracy of rawComm. For FaceBook, using nCut, rawComm, on average, is off by about two where the average is about nine communities per node. This seems quite a reasonable estimate, considering that the p value is less than 0.5.

3.6.3 Influence Maximization

In the implementation of the influence maximization problem used here the independent cascade model described in Section 3.5. All the edge weights are set to .01. Due to the stochastic nature of the model, for each method, the simulation was repeated 5,000 times and then the results averaged.

For evaluation purposes, the total number of nodes that are activated are compared as well as the number of communities reached using five different algorithms. The baseline *random* approach selects k nodes randomly. *degree* selects the k nodes with highest degree. These correspond to nodes whose community-based roles are designated as either big fish or ambassador. The algorithm proposed by Kempe, et al. [59], labeled *greedy*,

Table 3.2: Comparisons using Movie Data

		Average Group Coverage	
algorithm	**nodes**	**Director**	**Genre**
random	11.136	60	9.7
degree	18.996	83	12
greedy	22.084	259	12
comm	17.578	261	12
ambass	20.052	288	13

chooses one new node each iteration, selecting the node that will result in the greatest increase of activated nodes according to the Independent Cascade model. The last two methods use the metrics that have been proposed in this chapter. The method *comm* selects the k nodes with the highest rawComm score and *ambass* selects the k nodes with the highest sum of normalized degree and rawComm. The nodes selected by *ambass* are mostly ambassadors while those selected by *comm* is a combination of ambassadors and bridges.

The results for the movie data are summarized in Table 3.2. The column labeled *nodes* is the average number of nodes activated by the initial 10 nodes. The columns under *comm. coverage* indicates the number of communities that had at least one node activated. The greedy method, not surprisingly, was able to activate the greatest number of nodes. However, even though *ambass* activated fewer nodes, it was able to reach more communities. *comm* also was able to spread to a large number of communities even though it selected fewer nodes than *degree*, *greedy* or *ambass*. That is not too surprising given that nodes connected to many communities are not necessarily high degree nodes.

The p values were 1 for both types of community definitions. The q values were .947 and .164 for director and genre communities respectively. This means that for both types of communities if there is a link between two actors there is a 100% chance that the two actors will be in the same community. If there is no link between them then the q tells us that, for the director set, it is almost certain that they will not be in the same community, whereas for genre, there is only a 16% chance that they will be in different communities.

Table 3.3: Comparisons using Enron Data

algorithm	**nodes**	**Group coverage**
random	13	5.3
degree	13	6.9
greedy	15	6.8
comm	14	8.4
ambass	13	7.5

Table 3.4: Comparisons using FaceBook

algorithm	**nodes**	**Group Cov.**	**time**
random	12	3.5	10ms
degree	21	3.7	60ms
greedy	23	4.1	82min.
comm	18	5.6	371ms
ambass	20	4.3	411ms

For Enron data, because of the sparsity of the network, an edge weight of .05 was used. The results are summarized in Table 3.3. Again, even though *comm* and *ambass* did not activate the most nodes they provided the widest coverage.

The influence maximization results for FaceBook is given in Table 3.4. For this experiment communities based on concentration (students major) was used. Once again the algorithms *comm* and *ambass* outperform the others on spreading to a larger number of communities. For this data set the execution time for the different algorithms was also recorded and is listed in the last column. It can be seen that all the algorithms are quite fast with the exception of the *greedy* algorithm with is approximately 12,000 times slower than the slowest of the others. With large data sets or when time is critical, *degree* or *ambass* would be worthy alternatives.

3.7 Conclusions

This chapter explored the advantages of utilizing community knowledge in the analysis of networked data. Towards this end, several community-based roles were introduced according to the number of communities and the degree of each node. It was shown that nodes with roles called ambassadors are useful to maximize the number of communities activated during influence maximization. Metrics for estimating the number of communities and measuring the compatibility between communities and link structure are also proposed in this study.

Chapter 4

Relationship between attributes and links: Link Prediction

4.1 Introduction

Link prediction is a technique used to predict the formation of ties within a network. It can be used to recommend new relationships such as friends in social networks or collaborators in a bibliographic database. It may also uncover missing or previously unknown links such as regulatory interactions among genes or covert ties between criminal suspects.

More generally, the link prediction problem can be formulated as a binary classification problem [55, 68]—given a node pair, we seek to accurately predict whether there is an edge between them based on their node features, neighborhood structure, or other properties of the network topology. Such a problem has been approached in two ways: (1) using a generative approach, where the focus is on learning a model of the joint probability density of the nodes, links, subgroups, etc., and then make a prediction by using Bayes rule [38, 77, 78, 114], or (2) using a discriminative approach, which directly learns a target function that will map an input node pair to its class [47, 55, 68]. In this chapter a new discriminative learning technique for link prediction based on the matrix alignment approach is introduced.

It is assumed that the node attributes contain information needed to make the prediction but that the links would help us to prioritize the attributes. For example, in social

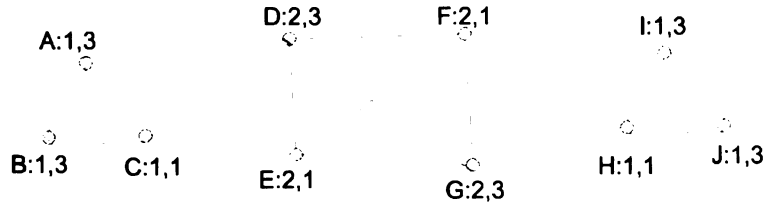


Figure 4.1: Clustering a small network

networks, people may become linked (friends, relatives, coworkers, accomplices, etc.) because they have shared some common characteristics or interest. While many attributes about a person may be known (e.g., eye color, height, books they like to read, school they attend, etc), it is a small set of them that are important when befriending. The challenge is to determine which subset of attributes are important to establish the links observed in a network. Another way to determine whether two persons should be linked is by examining their existing ties (e.g., do they have common friends or are they popular or influential figures?) It is not the purpose of this chapter to debate the merits of using attributes versus neighborhood or topological features [68]. Indeed, what may be appropriate for one data set may not be for another. The objective of this chapter is to present a flexible framework based on matrix alignment that allows us to identify the relevant attributes or topological features that are most well-aligned with the link structure.

To further illustrate the motivation behind this approach, consider the network shown in Figure 4.1. The figure shows a network of ten nodes, their identifiers, and attribute values (e.g., node A has the attributes $var1 = 1$ and $var2 = 3$). The question we would like to answer is: would it be more likely that node C would link to E or J? Using just the attributes it would appear that they would be equally likely since C has exactly one attribute value in common with both of the others. Using the topological features, it is probably more likely that C would link to E since they have a shorter path length, more common neighbors, etc. However, by examining the network we can tell that nodes that are linked are more likely to have identical first attributes than second. So we should assign a higher likelihood to C linking to J, than to E. For the network in Figure 4.1 attributes are more predictive than the topological features but in other networks it could

be otherwise. The proposed approach will automatically determine the most predictive attributes and topological features by aligning the adjacency matrix with weighted similarity matrices computed from the attributes and topological features. The weights of the similarity matrices can be easily determined by solving a system of linear equations.

Subgroups (also known as communities or clusters) often form in networks where groups of nodes have some common unifying properties. For example, imagining students forming study groups or joining social clubs can easily be imagined. One would expect the linking behavior to be different between the groups.

This chapter presents a matrix alignment framework that uses weights to align the attributes to the links. It is flexible so that it can utilize link, attribute and community knowledge. The framework is designed, in this chapter, to predict links. Table 4.1 lists the link prediction algorithms discussed in Chapter 2 as well as the data in a network that they utilize. As can be seen, the matrix alignment approach is the only one that uses attributes, links and community knowledge. It will be shown in Chapter 6 that it is also capable of using kernel functions.

The remainder of this chapter is organized as follows: Section 4.2 presents a preliminary discussion of terms. Section 4.3 presents the formulation of the matrix alignment approach. Experimental results to support the effectiveness of the proposed algorithm are given in Section 4.4 Finally, the conclusions are presented in Section 4.5.

4.2 Preliminaries

Consider a network represented as a graph $G = (V, E)$, where $V = \{1, 2, ..., n\}$ is the set of nodes to the objects and $E \subset V \times V$ is the set of edges. The link structure of the network can be represented by an $n \times n$ adjacency matrix $A = \left[a_{ij}\right]_{n \times n}$ where $a_{ij} = 1$ if there is a link between nodes i and j and zero otherwise. Each node $v \in V$ references an object (e.g. person, gene or web page) and is associated with a set of attributes. The attribute information for each node can be encoded in an $n \times d$ data matrix $X = [x_{ik}]_{n \times d}$, where each row of the matrix corresponds to a vector of attribute values for a node in the network.

Table 4.1: Summary of Link Prediction Algorithms

Name	attributes	links	communities
Al Hasan, et al. [47]	X	X	
Bilgic, et al. [14]	X	X	
Hanneke and Xing [46]	X	X	
Kashima and Abe [47]		X	
Lahiri and Berger-Wolf [63]		X	
Liben-Nowell and Kleinberg [68]		X	
O'Madadhain and Smyth [85]		X	
Popescul and Unger [89]	X		
Potgieter, et al. [90]	X	X	
Rattigan and Jensen [92]	X	X	
Taskar, et al. [115]	X	X	
Scripps, et al. [104]	X	X	X

If X is a binary-valued (0/1) matrix, each row, $\vec{x_i}$ can be normalized to have unit length, i.e., $< \vec{x_i}, \vec{x_i} >= 1$. By normalizing the rows, the matrix product XX^T would correspond to computing the cosine similarity between the nodes. On the other hand, if X is continuous-valued, each row vector can be standardized by subtracting their means and dividing by its standard deviation. Computing the matrix product XX^T then would be equivalent to calculating the attribute correlation between two nodes in the network. For brevity, it can be assumed the node attributes for all the networks considered in this study have been properly normalized or standardized. As a result, we may represent the attribute similarity between nodes using the matrix product XX^T. Tables 4.2 and 4.3 show the adjacency and data tables for the network in Figure 4.1. Note that the numbers for $var1$ and $var2$ from the figure have been binarized.

Let $\vec{w}$ be a vector of weights of length d. Let the weight matrix $W = [w_{hk}]_{d \times d}$ where $w_{hk} = 0$ for $h \neq k$ and $w_{kk} = \vec{w_k}$. Communities will be shown with superscripts like $X^{(h)}$ for the attribute data for community h. Using the adjacency and data matrices in Tables 4.2 and 4.3 from Figure 4.1 the weights were calculated using the method in

Table 4.2: Example adjacency matrix A

	A	B	C	D	E	F	G	H	I	J
A	0	1	1	0	0	0	0	0	0	0
B	1	0	1	0	0	0	0	0	0	0
C	1	1	0	1	0	0	0	0	0	0
D	0	0	1	0	1	1	1	0	0	0
E	0	0	0	1	0	1	1	0	0	0
F	0	0	0	1	1	0	1	0	0	0
G	0	0	0	1	1	1	0	1	0	0
H	0	0	0	0	0	0	1	0	1	1
I	0	0	0	0	0	0	0	1	0	1
J	0	0	0	0	0	0	0	1	1	0

Section 4.3 and are displayed in the diagonal matrix in Table 4.4. Note that there are higher values for $var1$ (columns 1 and 2) than for $var2$, which appears to be intuitive.

Table 4.3: Example data matrix X

	var1=1	var1=2	var2=1	var2=2	var2=3
A	1	0	0	0	1
B	1	0	0	0	1
C	1	0	1	0	0
D	0	1	0	0	1
E	0	1	1	0	0
F	0	1	1	0	0
G	0	1	0	0	1
H	1	0	1	0	0
I	1	0	0	0	1
J	1	0	0	0	1

Topological data will be represented by matrices $Y^{(k)} = \left[y_{ij}^{(k)}\right]_{n\times n}$, for each topological metric. For example, given that $Y^{(k)}$ is the matrix for common neighbors, then $y_{ij}^{(k)}$ represents the number of common neighbors between nodes i and j.

In this chapter, the *missing link* problem of link prediction is considered. In this problem the attributes and the link structure is given but some of the node-pairs have missing

Table 4.4: Example weight matrix W

	1	2	3	4	5
1	0.4046	0	0	0	0
2	0	1.0116	0	0	0
3	0	0	-0.0694	0	0
4	0	0	0	0.0000	0
5	0	0	0	0	0.0000

values and the goal is to correctly predict the link value. The matrix alignment framework could be used for other problems such as predicting the initial links of new nodes or predicting the new links in a dynamic network but this work focuses solely on the missing link problem.

4.3 Methodology

The matrix alignment framework is presented in this section starting with a description of the method of aligning the attributes to the links. After the initial formulation it is extended it to make it more powerful. The first extension introduces a regularization term for data sets where overfitting can be a problem. Next topological data is added to the framework providing an additional source of knowledge for predictions. Finally a novel method for simultaneously learning communities and the attribute alignment weights for them is presented.

4.3.1 Alignment of Links and Attributes

In a network, the similarity between nodes can be measured by their links. For now, assume the simplest case where two nodes are *similar* if linked and *dissimilar* if not linked. Other link-based similarity metrics (e.g. common neighbors, shortest path, etc.) will be discussed later. Similarity can also be measured using the attributes and any of the traditional proximity metrics (e.g. cosine, jaccard, etc.) In an ideal network, one can imagine perfect alignment between the links and the attributes - that is where $sim(x_i, x_j) = a_{ij}$ for all i, j. However, in most networks such perfect alignment will not exist.

Let X be a normalized data matrix and XX^T be the similarity matrix for each pair of nodes in the network. Given a set of weights $\vec{w} = \{w_1, ..., w_d\}$ (one for each of the attributes), XWX^T would correspond to the weighted similarity matrix. The formulation for the matrix alignment approach is to minimize the following objective function:

$$L = \|A - XWX^T\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij} - \sum_{k=1}^{d} x_{ik}x_{jk}w_k\right)^2$$

Intuitively, the objective function aims to learn a set of weights that will maximize the degree of alignment between the link structure and attribute similarity. The first step is to take the derivative of L with respect to each $w_m, 1 < m < d$:

$$\frac{\partial L}{\partial w_m} = -2 \cdot \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij} - \sum_{k=1}^{d} x_{ik}x_{jk}w_k\right) \cdot x_{im}x_{jm}$$

and set them to zero. Rearranging terms we have:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \cdot x_{im}x_{jm} = \sum_{k=1}^{d}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ik}x_{jk} \cdot x_{im}x_{jm}\right) w_k$$

These d equations can then be arranged into a matrix-vector multiplication problem $b = Z\vec{w}$, by letting

$$b_m = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \cdot x_{im}x_{jm}$$

and

$$Z_{mk} = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ik}x_{jk} \cdot x_{im}x_{jm}$$

The weight vector $\vec{w}$ can be computed using Gaussian elimination. The complexity for assembling the b vector and Z matrix is approximately $O(n^2d^2)$ (although it can be sped up if the matrix X is sparse) and $O(d^3)$ for the Gaussian elimination.

Once the vector $\vec{w}$ has been learned the expression $x_i W x_j^T$ provides a relative measure of the likelihood of objects, i and j forming a link. Unlike some approaches to link

prediction, the weights have the flexibility to be applied to a number of different settings. Another advantage to this approach is its extendability. Now that a matrix alignment solution to link prediction has been shown, it will be modified it in a number of ways to make it more flexible and powerful.

4.3.2 Regularization

To avoid overfitting the weights to the training data, a regularization technique can be employed. Typically in problems that are overdetermined, a regularization penalty term $\lambda\|W\|_F^2$ can be added to the objective function. In this case, this will coerce the weights to zeros for high values of λ. By using a penalty term of $\lambda\|W - I\|_F^2$:

$$\begin{aligned} L &= \|A - XWX^T\|_F^2 + \lambda\|W - I\|_F^2 \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij} - \sum_{k=1}^{d} x_{ik}x_{jk}w_k\right)^2 + \lambda\sum_{k=1}^{d}(w_k - 1)^2 \end{aligned}$$

will reduce to the original matrix alignment $\|A - XWX^T\|_F^2$ when $\lambda = 0$ and when λ is large, $w_k \rightarrow 1$ for all k. Taking the partial derivatives with respect to w_m:

$$\begin{aligned} \frac{\partial L}{\partial w_m} &= -2 \cdot \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij} - \sum_{k=1}^{d} x_{ik}x_{jk}w_k\right) \cdot x_{im}x_{jm} \\ &\quad +2\lambda w_m - 2\lambda \end{aligned}$$

After setting to zero and rearranging terms we get the matrix equation $b = (Z + \lambda I)\vec{w}$, where b and Z are defined the same as in the previous section. This has the same complexity as the previous formulation.

4.3.3 Incorporating Topological Data

Since most link prediction algorithms make use of knowledge of the link structure, it will be shown how this topological data can be incorporated into the matrix alignment framework as well. Basically, the topological metrics (path length, common neighbors,

etc.) are treated like attributes and are provided a weight for each. For this subsection, let d_1 be the number of attributes and d_2 to be the number of topological metrics. Recall that for the k^{th} topological metric, $Y^{(k)} = \left[y_{ij}^{(k)}\right]_{n\times n}$ contains the metric values for each pair of nodes. The new objective function for aligning the adjacency matrix with the similarity matrices for attributes and topological features becomes:

$$L = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij} - \sum_{k=1}^{d_1} x_{ik}x_{jk}w_k - \sum_{k=1}^{d_2} y_{ij}^{(k)}v_k\right)^2$$

where $V = \{v_v, ..., v_{d_2}\}$ is the vector of weights for the topological metrics.

The weights will be solved for as before. Taking the partial derivative $\frac{\partial L}{\partial w_m}$, setting it to zero and rearranging as before:

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{im}x_{jm} &= \sum_{k=1}^{d_1}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ik}x_{jk}x_{im}x_{jm}\right)w_k \\ &+ \sum_{k=1}^{d_2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} x_{im}x_{jm}y_{ij}^{(m)}\right)v_k\end{aligned}$$

Next the partial derivative $\frac{\partial L}{\partial v_m}$ is taken, set it to zero and terms rearranged to get:

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}y_{ij}^{(m)} &= \sum_{k=1}^{d_1}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ik}x_{jk}y_{ij}^{(m)}\right)w_k \\ &+ \sum_{k=1}^{d_2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij}^{(k)}y_{ij}^{(m)}\right)v_k\end{aligned}$$

The same method is used to build the system of equations

$$\begin{bmatrix} b \\ c \end{bmatrix} = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}\begin{bmatrix} \vec{w} \\ \vec{v} \end{bmatrix}$$

where,

$$
\begin{aligned}
b_m &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \cdot x_{im}x_{jm} \\
c_m &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \cdot y_{ij}^{(m)} \\
P_{mk} &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ik}x_{jk} \cdot x_{im}x_{jm} \\
Q_{mk} &= \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij}^{(k)} \cdot x_{in}x_{jm} \\
R_{mk} &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ik}x_{jk} \cdot y_{ij}^{(m)} \\
S_{mk} &= \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij}^{(k)} \cdot y_{ij}^{(m)}
\end{aligned}
$$

Again Gaussian elimination is used to find $\begin{bmatrix} \vec{w} \\ \vec{v} \end{bmatrix}$ which is a $d_1 + d_2$ sized vector. After learning the weights,

$$\sum_{k=1}^{d_1} x_{ik}x_{jk}w_k + \sum_{k=1}^{d_2} y_{ij}^{(k)}v_k$$

provides a relative measure of the likelihood of i and j forming a link according to the similarity of their attribute values and topological features.

4.3.4 Communities

Recall that the motivation for the matrix alignment approach is that links are formed between nodes based on attributes but that not all attributes contribute the same. Consider typical social networks where groups form around specific areas of interests. For example, on a college campus, one group may form because the members have an interest in poetry while another might form around the topic of sports. It would be reasonable to assume that the weights that align the attributes to links would be different from one group to another.

To make use of the community information it is proposed to learn a separate set of weights for each community. Let $C : V \rightarrow \{1, ..., p\}$ be a clustering function that maps every node to a group ID. For community c let the adjacency matrix $A^{(c)} = \left[a_{ij}\right]_{n_c \times n_c}$ where $a_{ij}^{(c)} = a_{ij} \forall i, j$, where $C(i) = C(j)$ and n_c is the number of nodes in the community c. The data matrix for community c, $X^{(c)} = [x_{ik}]_{n_c \times d}$ is simply the rows from X corresponding to the nodes in community c. Then a separate set of weights $W^{(c)}$ can be learned from each community.

Before describing the approach to learning both the communities and the weights, a simpler approach is examined and its weaknesses exposed. The simple, straightforward approach is to, first find the communities through a community finding or clustering algorithm and then to calculate the weights for each community separately as described in Section 4.3.1.

Consider, first, clustering the nodes using the links, as a graph partitioning algorithm would. The clusters will be tightly connected groups. However the groups will not necessarily reflect a unified linking behavior. The goal is to find groups that have shared interests which means that not only are the groups tightly connected but that they also have similar attributes.

Another approach would be to cluster based on the attributes. The clusters would then be nodes with similar interests but recall that the contention is that not all attributes are important to linking. Thus the communities would not necessarily represent clusters of nodes with similar linking behavior. This leads to a dilemma: the linking behavior of the clusters cannot be determined without knowing the clusters but the clusters cannot be created without knowing the linking behavior.

The goal here is to simultaneously learn communities of similar linking behavior and the weights associated with those communities. Now the interest is in learning a set of weights $\vec{w}^{(h)}$ for each community h. The objective function changes slightly to

$$L = \sum_{h=1}^{c} \|A^{(h)} - X^{(h)} W^{(h)} X^{(h)T}\|_F^2$$

where $\vec{w}^{(h)}$ forms the diagonal of $W^{(h)}$ with zeros everywhere else. The matrices $A^{(h)}$

and $X^{(h)}$ are the adjacency and data matrices for the h^{th} community, formed by using the graph induced by the nodes in community h. To learn the weights we propose an iterative type of algorithm (see Algorithm 1). First, it recalculates the weights using the matrix alignment approach for each community. Then, it determines the maximum gain for each node. The *gain* formula for moving node i from community h to community c is

$$\begin{aligned} gain(i,h,c) &= \sum_{j \in A^{(c)}} \left(a_{ij}^{(c)} - \sum_{k=1}^{d} x_{ik} x_{jk} w_k^{(c)} \right)^2 \\ &- \sum_{j \in A^{(h)}} \left(a_{ij}^{(h)} - \sum_{k=1}^{d} x_{ik} x_{jk} w_k^{(h)} \right)^2 \end{aligned}$$

and given that h is the current community for i the maximum gain is

$$maxGain(i) = \underset{c}{\operatorname{argmin}}\, gain(i,h,c)$$

Reassigning every node to the community of maximum gain leads to unstable communities making convergence difficult. A more stable approach is to select the q nodes with the highest maximum gain and reassign those.

Once the communities and weights have been learned they can be applied to the link prediction problem. However, it is not implied that nodes will not link exclusively to others in the same community. This presents a problem of how to evaluate the potential of a link between nodes from different communities. The approach presented here is to use different weights for intra-community pairs versus inter-community pairs. In addition to the weights that are learned for each community a set of global weights are also learned for the entire set. Then for node pairs that belong to the same community the weights for that community are used and for pairs where the nodes belong to different communities the global weights are used.

4.3.5 MatAlign

Iterative algorithms such as the well known EM algorithm have been used successfully to find optimal solutions when there are missing values in the data. Thus for the problem of

predicting missing links we propose using an iterative approach to simultaneously solve for the weights and the missing values. First, weights are calculated using only the known links. Then, the algorithm alternates between assigning new values to the missing links and recalculating the weights using the newly assigned link values.

This approach is also proposed for community assignment. The main loop alternates between calculating the weights for all of the communities and reassigning the nodes to communities. For the initial community assignment, any clustering algorithm can be used.

The *matAlign* algorithm that is presented in Algorithm 1 combines both of the iterative approaches proposed above. Note that it can be easily modified (e.g., to use without communities, simply remove the code related to communities). For conciseness, let $W = \{\vec{w_1}, ..., \vec{w_p}\}$ and $W^{(c)}$ be the matrix of zeros with the diagonal equal to $\vec{w_c}$. The $calcWgts$ function calculates the weights for the given adjacency and data matrix using the matrix alignment approach described earlier in this section with any of the extensions for regularization or topology. The $getHighestGain$ function returns the q nodes with the largest gain. The algorithm then reassigns those nodes to the communities that results in the largest gain in alignment.

The algorithm proceeds as follows: First the initial communities are assigned using Kmeans. Inside the main loop, first the weights are recalculated, then the missing links are predicted and finally the communities are reassigned. Convergence is achieved when the weights and the community assignments have stabilized. A maximum number (100) of iterations was also imposed. Typically convergence occurred within a few iterations.

4.4 Experimental Evaluation

In this section the utility of the matrix alignment approach is demonstrated by showing the results of experiments on several data sets for the missing links problem. In this problem, a test set of node-pairs are given for which the link value – zero for non-link, one for link – is unknown. For the training set, which are the remaining node-pairs, the link values are known. The problem is to predict the values for the test set given the training set.

Input: adjacency matrix A, data matrix X, number of communities p, number of nodes to reassign

Output: adjacency matrix A, weights W, community assignments C

$C \leftarrow Kmeans(X, p)$;

repeat

  // calculate weights

  for $c \leftarrow 1$ **to** p **do**

    $W^{(c)} \leftarrow calcWgts(A^{(c)}, X^{(c)})$;

  end

  // missing link assignment

  foreach *missing* $a_{ij} \in A$ **do**

    $a_{ij} \leftarrow assignLink(i, j, A, X, W)$;

  end

  // community assignment

  $V_{best} = getHighestGain(W, A, X, C, q)$;

  foreach $v \in V_{best}$ **do**

    $C(v) \leftarrow \text{argmin}\, c \sum_{j=1}^{|V|} (a_{vj} - \sum_{k=1}^{d} x_{vk} x_{jk} w_k^{(c)})^2$;

  end

until *converge* ;

return *W,A,C*;

Algorithm 1: Simultaneously learning weights and predicting links

4.4.1 Experimental Setup

Data for the experiments was taken from DBLP data base [67] of computer science publications, data crawled from the website TakingItGlobal.org and the WebKB data [121] set from the Linqs [69] website. Table 4.5 summarizes the sets and their properties.

The conference proceedings from the DBLP dataset was extracted resulting in over 500,000 papers over a 30 year period from over 8,000 conferences. For nodes the authors of the papers were used, linking them based on co-authorship. Because of the limited information available we used the words in the title as the attributes, selecting the 613 words

Table 4.5: Data Sets

Data set	**nodes**	**links**	**attributes**
DBLP	10709	22315	580
0: database	3445	8547	542
1: art. intel.	3492	5797	556
2: networks	2855	5586	524
3: soft. eng.	1238	2042	410
TakingItGlobal.org	5852	29776	123
0: Africa	1128	2387	123
1: Asia	855	1778	123
2: Europe	368	593	123
3: Australasia	137	302	123
4: North America	1258	4221	123
5: South America	334	1095	123
6: caribean	72	181	123
7: Middle East	304	674	123
Webkb			
0: Cornell	195	304	1703
1: Texas	187	328	1703
2: Washington	230	446	1703
3: Wisconsin	265	530	1703

– after stemming – that appear between 1,000 and 100,000 times in all the titles. Training data was selected from papers published in 1998 through 2003 and the test data were randomly selected node pairs. To get a variety of test sets papers from four sets of conferences representing the database, artificial intelligence, network and software engineering communities were selected.

TakingItGlobal.org (TIG) is a social networking website for young people to become involved in activities and share their ideas about global issues like poverty, social justice and health. Members (nodes) can establish friendship links (links) and select the activities and events of interest, post general, textual information about themselves and post journals. The problem with the textual information on this site is the multilingual nature. Therefore we used the activities and events and other demographic data as attributes. The members were grouped by region.

The WebKb data set is a collection of web pages collected from four university web sites in January of 1997. The data was preprocessed by Linqs into a network based on selecting a number of words from the web pages that were deemed to be significant. The data is organized into a binary adjacency matrix and a binary data matrix.

Experiments were primarily done on data sets that have been used in other works on link prediction. The main publications in this area have done experiments using DBLP and WebKb as well as the bibliographic sets high energy physics [68], CiteSeer [89] and biobase [47].

Predictions are made using a pair of quadratic discriminators [32] g_0 and g_1. g_0 is trained on the similarity scores (XX^T, XWX^T, etc.) for each pair of nodes in the training set that are not linked, and g_1 is trained on the scores for linked training set pairs. Each pair in the test set is predicted to be a link/non-link depending on which discriminator g_1/g_0 is higher using the score calculated for the pair.

The results report the values for precision, recall, accuracy, and F-measure. Accuracy was calculated as the number of correct predictions divided by the total sample size. Predictions are made by calculating the similarity score (XX^T , XWX^T , etc.) and applying a threshold to assign a link (1) or non-link (0). To find the threshold the average similarity score was calculated for the labeled links and for the labeled non-links and the average of

these averages used.

For all experiments the results of 10 tests runs were averaged. For each run, a test set of 10% of the links plus an equivalent number of non-links were randomly selected. A regularizer value of 1 was used (except where noted, explained below).

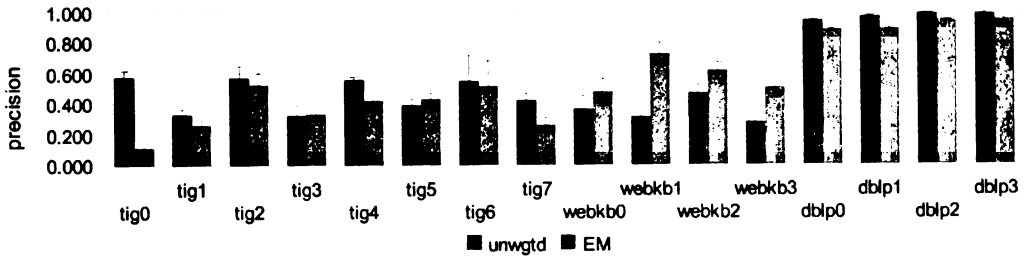
4.4.2 Experimental Results

The results that are shown here support the framework and the extensions that were presented in Section 4.3. In the first two subsections the effectiveness of matrix alignment, first with and then without topological data will be demonstrated. Then it will be shown how the regularizer can be used to compensate for overfitting. Next the results for the community finding extension will be shown.

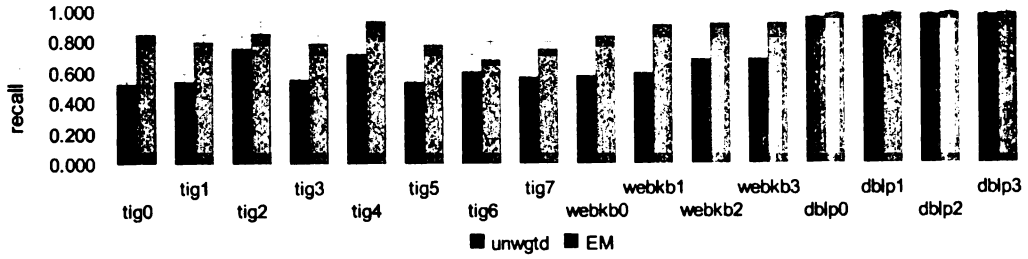
Matrix Alignment

The experiments conducted in this section calculated the accuracy of predicting links using the *unweighted* similarity measure XX^T and our EM method of alternately calculating the weights and then reassign the missing links. The results are summarized in Figure 4.2 where (a) shows the precision, (b) shows the recall, (c) shows the accuracy and (d) shows the F measure. For each data set there are two bars, the first for unweighted and the second for EM. For all data sets except for three of the DBLP sets using weights was more accurate for predicting links than the unweighted measure. The error bars at the top show the standard deviation of the test runs.

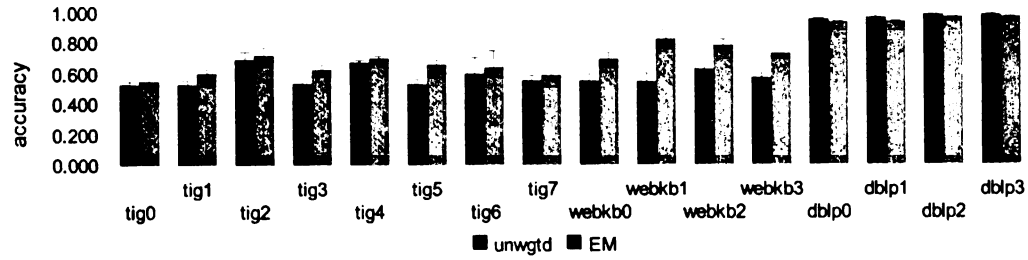
By comparing the four charts in Figure 4.2, it can be seen that while the accuracy is generally better for the EM approach, the precision and recall vary, especially for TIG and DBLP. For these two sets it can be seen that compared to the unweighted similarity, the precision is lower for EM but for recall it is higher. From this it appears that EM finds more links than using unweighted attributes but at the expense of classifying too may non-links as links. Overall, though the accuracy is better for EM. The error bars indicate that the results appear to be significant except for the smaller sets like TIG2 and TIG6.



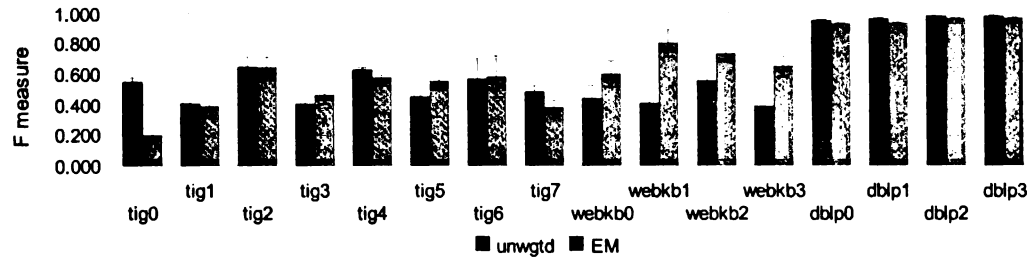
(a)



(b)



(c)



(d)

Figure 4.2: Comparison of unweighted similarity vs. EM for predicting missing links

From Section 4.1 recall that the motivation is that not all attributes are equally helpful when predicting links. The weights that are learned should and do reflect this. The

network generator that is used to create the synthetic data sets can be set to give a higher priority to some attributes. When the results are examined the higher priority attributes had much higher weights than the others.

When looking at the TIG data some attributes such as *gender* were low weighted but others such as *member groups* were highly weighted. This can be interpreted to mean that when people befriend others the groups they belong to are more important than their gender. Also, age groups in the 20's and 30's have low weights while the 40's age group had higher weights. Since TIG is a site promoted towards younger people it may be that the fewer older members tend to stick together.

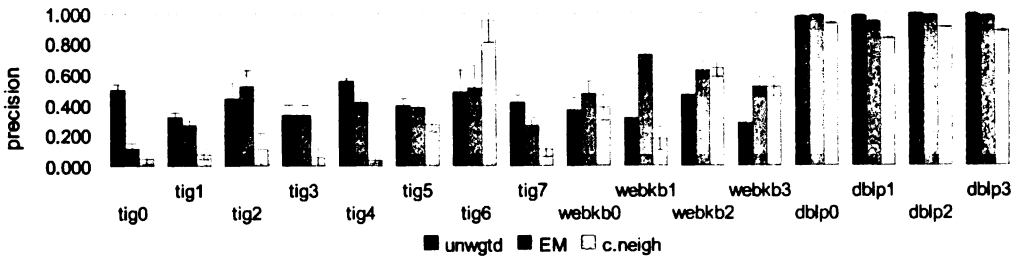
For the WebKb sets, the attributes were the words from web pages but the processed set used here only had the binary data. So while it cannot be reported which words were important in linking it can be concluded that there were in fact some words that were more important than others.

For the DBLP our framework did worse than using unweighted attributes for three of the sets. This is easily explained and remedied (see the section below on using the regularizer). These sets are bibliographic data sets where the authors are linked by co-authorship and the attributes are the words in the publication titles. The problem with trying to predict the missing links here is that the links represent papers that two authors co-wrote. The unique words that appear in the title of that paper may not appear in any other publications which means that they will be weighted lower. So in effect the weights are overfitted to the training data.

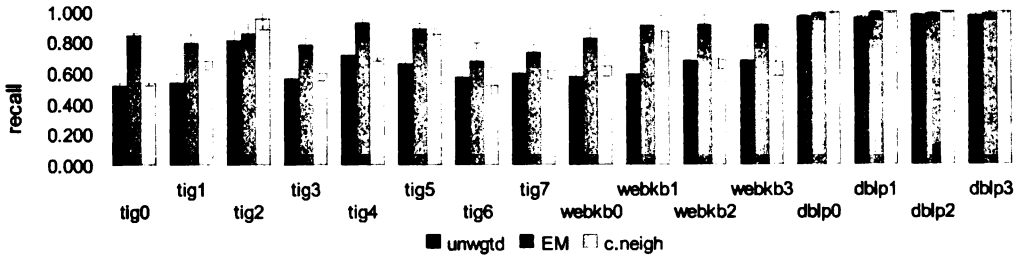
Incorporating Topological Data

According to the study by Liben-Nowell and Kleinberg [68], *common neighbors* is often one of the most effective topological metrics for link prediction. Thus we chose it to be added to our framework in order to improve the predictions. The experiments performed in this section compare a linear combination of the unweighted attributes plus *common neighbors* and the weighted combination as described in Section 4.3.3.

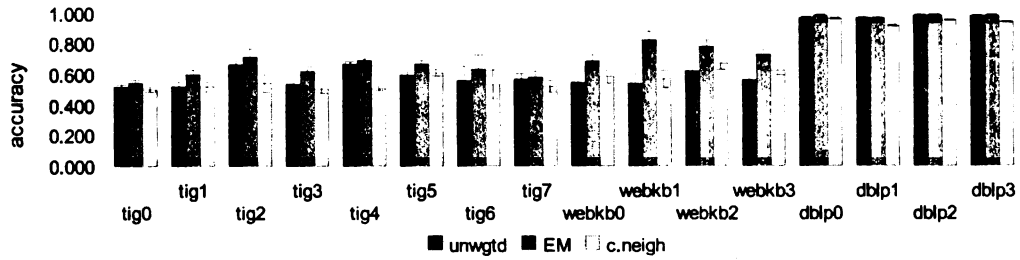
Figure 4.3 shows the results of the tests. Again the results are presented for (a) precision, (b) recall, (c) accuracy and (d) F measure. Using the topology boosts the accuracy



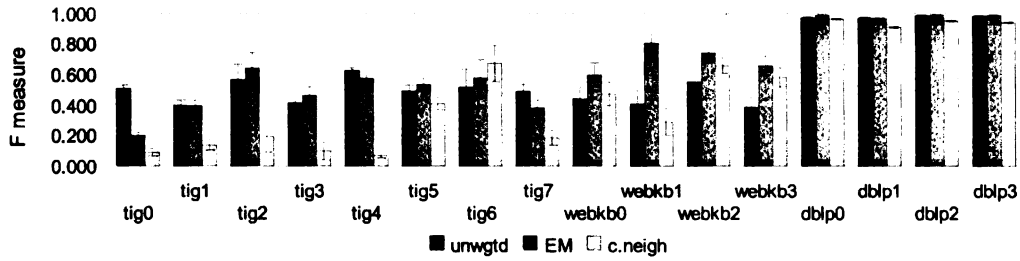
(a)



(b)



(c)



(d)

Figure 4.3: Comparison of unweighted vs. *EM* for prediction using topological data

for unweighted and EM methods for all of the sets. The results are also compared to predicting links based only on the common neighbors statistic. It can be seen that while

common neighbors can be very predictive, the EM approach is still better. As with the previous experiments EM is consistently better at predicting both the links and the non-links. And again, it finds more links but at the expense of predicting more non-links as links.

The weighted predictions in these tests were all better than the unweighted predictions. Additionally, using EM with topology, in all cases, was at least as good as using it without topology. With WebKb and TIG the accuracy was the same for EM using topology or not. For DBLP, using topology improved the accuracy for all of the sets. When the weights are examined after the tests the topology feature was higher when for sets where common neighbors was predictive and lower for sets where it was not as helpful.

As with the other tests, the framework identifies the important factors involved in making a link prediction. Instead of aligning just the attributes to the links it aligns a linear combination of the attributes and topological metrics to the links. This shows how our method is easily extensible to accepted more than just the attribute data.

Improvement Using the Regularizer

Data sets like DBLP make it difficult to use weights for link prediction because of the large number of attributes and the fact that many of the attributes are important to a small number of links. Thus if those links are missing, the associated attributes will be lower weighted. Placing more weight on the attributes associated with the training data leads to overfitting.

By using larger values for the regularizer the framework can avoid overfitting. Figure 4.4 shows the results of using our framework to predict links for the DBLP3 set with regularizer (r) values of 1, 10 through 100. The predictions plateaued when the regularizer gets close to 100. In a worse case scenario, the regularizer can be increased to a very large number which will force the weights become uniform. In this case predictions using XWX^T will be exactly the same as XX^T. Such extreme circumstances could mean that the attributes are perfectly aligned to the links without weighting or, more likely, that the training sets are inadequate to predict which attributes will be important in the test sets.

Figure 4.5 shows the results re-running the prediction for all four DBLP sets using a

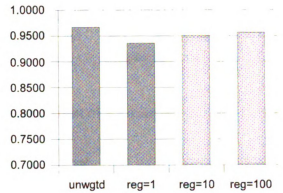


Figure 4.4: Reducing overfitting by changing the regularizer

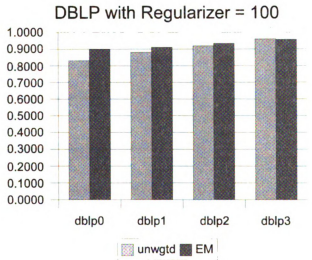


Figure 4.5: Predicting links for DBLP using a Regularizer of 100

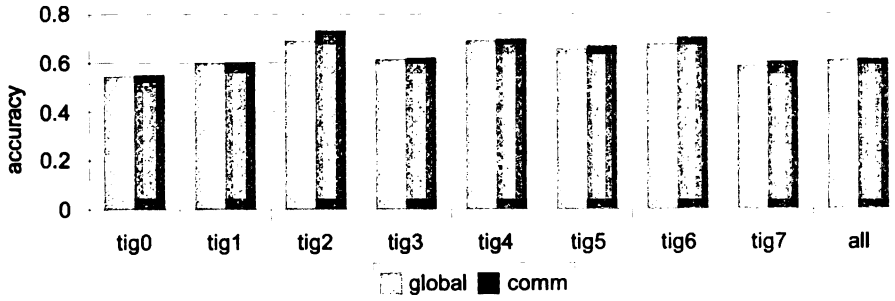


Figure 4.6: Comparison of accuracy for EM vs. community weights

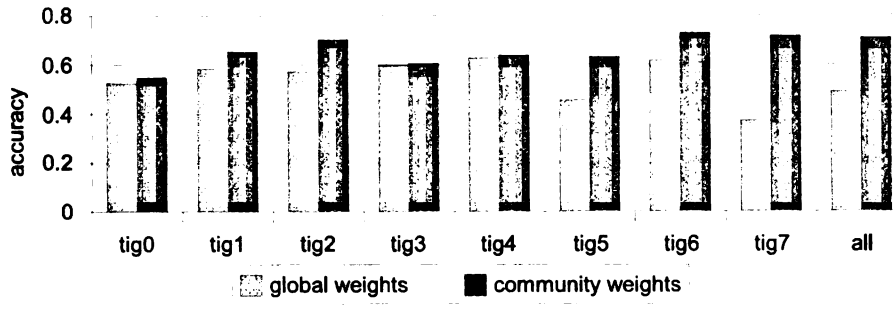


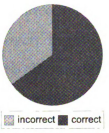
Figure 4.7: Comparison of links within communities using global vs. community weights

regularizer of 100. Each set improved so that the weighted predictions were better than unweighted in all tests.

There are methods to find the best regularizer automatically. For the *missing links* problem it is suggested that cross-validation be used on the training set with varying regularizer values, selecting the one that results in the best accuracy.

Simultaneous Learning of Weights and Communities

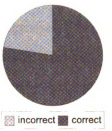
The communities that are found using the method proposed in Section 4.3.4 are not communities in the traditional sense. Some algorithms group together nodes of similar attributes. Others find densely linked groups of nodes. Whether the algorithm uses attributes or links (or both), there is the underlying assumption that the objects within the groups are "similar" and "disimilar" to objects in other groups. The matrix alignment method finds communities that have common linking behavior, that is, they share a common set of weights.



(a)

geography-based

communities



(b)

MatAlign communities

Figure 4.8: Within-community predictions: ratio of correct to incorrect predictions

In this section we demonstrate the usefulness of simultaneously finding the weights and communities using the matAlign algorithm on the TIG data set. For these tests we used all eight of the TIG communities as well as the all of the nodes in the entire set. Predicting links for two nodes *within* the same community is straightforward. However, for *between* pairs where the nodes are from different communities it is not so clear; a number of strategies can be employed such as averaging the two sets of weights or using the global weights. After some preliminary tests, it was decided to use the global weights for nodes from different communities. The tests were run using $k = 10$ for the number of communities to find.

The overall results comparing predictions made using the community weights versus those made using the global weights (those found without using communities) are shown in Figure 4.6. In every test, using the community weights resulted in accuracy at least as high as using global weights. Since these results include a large number of between pairs where the accuracy is identical it is helpful to compare the prediction using only the within pairs. Figure 4.7 compares the accuracy of these pairs using the global weights and using the community links. This gives a clearer picture of the improvement possible using the community weights.

One concern was that by splitting up the data set by geographic region some natural communities in the data would also be split up. To illustrate this effect we examined the within community pairs and compared the pairs that were predicted correctly by the

community weights but incorrectly by the global weights (labeled *correct*) against those that were predicted incorrectly by the community weights but correctly by the global weights (labeled *incorrect*). Figure 4.8 shows the comparison of these two numbers when they are summed across all geographic communities (Africa, Asia, etc.) versus when the prediction is made based on the MatAlign communities. For the geographic communities the percentage of correct predictions is 65% and for MatAlign it was 77%. This suggests that there is a better chance of finding good communities when MatAlign is used.

Table 4.6: Sample Communities

ID	**nodes**	**links**	**comm. acc.**	**global acc.**	**highly weighted attributes**
34	37	173	0.75	0.45	age<20 age>50
97	19	20	1.00	0.50	citizenship=U.S. residence=Brazil
116	28	32	1.00	0.00	citizenship=U.S. citizenship=misc born in U.S.
160	37	74	0.83	0.16	age>50 citizenship=Brazil

A closer examination of the test that used all of the TIG data, reveals that there were several communities where the prediction accuracy was much higher using the community weights than for the global weights. Table 4.6 shows a recap of four communities and the attributes that were highly weighted. In community 34, where the accuracy using community weights was 75% versus 45% for the global weights. The two highest weighted attributes were $age < 20$ and $age \geq 50$. This means that within the community, two users who are both under twenty are likely to be linked and, as well, two users fifty or older are likely to be linked. On closer inspection, in that community, 100% of the pairs where both members are under 20 are linked, as are 100% of the pairs where both are 50 or over. Faced with a missing link where the two members are under 20 or over

50 it makes sense to weight those attributes higher, in order to improve the chances of predicting it as "linked". Generally, attributes common to tightly linked node pairs tend to be highly weighted.

Tests on the DBLP data sets were also run but the results are not presented because they are not as meaningful as the TIG data. The nature of the DBLP set is that the similarity between the attributes is well aligned with the links as demonstrated in earlier tests. As a result, the tests with community weights had higher accuracy than the EM solution but lower than the unweighted solution when the regularizer was set to 1. When it was set to 100, the accuracy for the community weights was exactly the same as for EM (since the weights were close to being uniform). The other data sets were not used since there are no descriptions for the attributes making the community analysis less meaningful.

4.5 Conclusions and Future Work

A discriminative framework to predicting links was presented that aligns attributes and link metrics to the link structure. The two general approaches to link prediction so far have been either generative or discriminative. While each approach has its advantages and disadvantages, it has been suggested in general "that discriminative classifiers are almost always to be preferred to generative ones".

Algorithms that predict links, can use many different sources of information, such as attributes, links and communities. Of the algorithms presented thus far, the matrix alignment framework is the only one to make use of all three sources.

The framework has the advantage of being flexible, allowing many extensions to the framework. It was shown in this chapter that it could be extended to incorporate topological data. This framework can also be extended in other ways, e.g., by using kernels in cases where the relationship between the link and attributes is not necessarily linear.

Chapter 5

Relationship between Attributes and Links: Collective Classification

5.1 Introduction

Although predicting the class of an instance is a well established area of research, classification of nodes in a network is an active [19, 39, 70, 107, 114, 123] subject of study. Networks have links between the nodes, which potentially offer more information to make more accurate predictions. However, making use of the links has proven to be more than just extending an existing model. The reason is the different nature of link and attributes. Classifiers use attribute information which ordinarily results in a simple $n \times d$ data matrix – there are n nodes, each having d attributes. Links establish relationships between nodes commonly represented by an $n \times n$ adjacency matrix. The problem of utilizing the link data becomes one of integrating the node-based data matrix with the node-pair-based adjacency matrix.

The problem is illustrated in Figure 5.1. In this simple network, the shape of a node reveals its class. Imagine that we wish to decide whether the node at the bottom belongs to the square or the oval class. In this network, we can deduce that the first two attributes are helpful in determining the class as well as the links. The new node has a strong attribute similarity to the oval nodes but is strongly linked to the square ones. It would be helpful in this case to know the relative importance of links versus attributes when predicting the

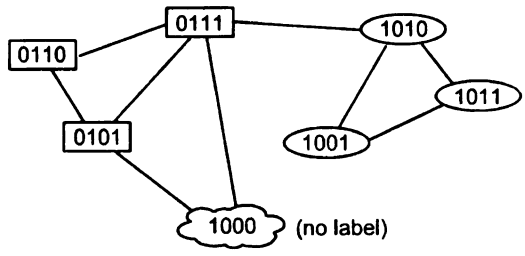


Figure 5.1: Sample network

class.

This chapter presents a new approach to the problem of collective classification. This framework learns a set of weights that can be applied to the attributes and the links simultaneously. Besides being predictive the weights also provide a relative measure of the importance of not only the attributes but also the links. This means that we can assign nodes to classes and then be able to tell why they were assigned to the class. In this formulation it is assumed that there are no missing attribute or link values.

The methodology for this approach is described in Section 5.2. The results of the experiments are presented in Section 5.3. The conclusions are presented in Section 5.4.

5.2 Methodology

Using both the link and attribute data to predict the class of a node is inherently troublesome because of the nature of the two data sources. Attribute data is node based while link data is node-pair based. Integrating the two types provides an interesting challenge. To meet this challenge, in the matrix alignment approach, the goal is to form a relationship between links and the attributes by using $n \times n$, node-pair, matrices.

Let a network be represented as a graph $G = (V, E)$, where $V = \{1, 2, ..., |V|\}$ is the set of nodes and $E \subset V \times V$ is the set of edges. The adjacency matrix $A = (a_{ij})_{n \times n}$ represents the links structure, where $a_{ij} = 1$ if there is a link between nodes i and j and zero otherwise. Each node $v \in V$ references an object (e.g. person, gene or web page) and is associated with a set of d attributes. The attribute information for each node can be encoded in an $n \times d$ data matrix $X = (x_{ik})_{n \times d}$, where each row of the matrix corresponds

to a vector of attribute values for a node in the network. For brevity, it is assumed the node attributes for all the networks considered in this study have been properly normalized or standardized. As a result, the attribute similarity between nodes can be represented using the dot product of the attribute matrix XX^T.

5.2.1 Alignment of Links and Attributes

This chapter is concerned with predicting the class of a node. Since it is not possible to align these $n \times n$ matrices to the class vector $\vec{y}$, instead, the $n \times n$ co-class matrix Y is used, where $Y_{ij} = 1$ if i and j are in the same class and $Y_{ij} = 0$ otherwise. To align Y with the adjacency matrix A and the attribute similarity maxtrix XX^T, the scalar weight w_0 for A and the two d length vectors $\vec{w}$ and $\vec{v}$ are used. The formulas use the square diagonal matrices W and V where $W_{ii} = \vec{w}_i$, $V_{ii} = \vec{v}_i$ and zeros on the off-diagonals. The initial formulation was:

$$L = \|Y - w_0 A - XWX^T\|_F^2$$

This objective function is designed to learn a set of weights that will maximize the degree of alignment between the class co-occurrence matrix and the link structure and attribute similarity matrices. Small values of this expression indicate a high degree of alignment.

This design is too rigid though. Recall that Y_{ij} is 1 if both i and j have the same class and zero otherwise. So if i and j are in the same class it would be ideal for the sum of $w_0 A_{ij}$ and $X_i W X_j^T$ to be 1 and if they are in different classes the sum should be zero. Consider the following example with nodes i, j and k, where j and k have identical attribute vectors which are very dissimilar to i's attribute vector. Further, assume that they are all in the same class ($Y_{ij} = 1$, $Y_{ik} = 1$, and $Y_{jk} = 1$) and that $A_{ij} = 0$, $A_{ik} = 1$ and $A_{jk} = 1$. It is impossible to find weights to satisfy the formula above for these three nodes. The problem lies in balancing the link and attribute similarity information.

The desired formulation, then should consider two sets of attribute weights; one for the attributes of linked pairs and another the attributes of unlinked pairs. To overcome this

problem, then, the following formulation is proposed:

$$L = \|Y - w_0 A - XWX^T \circ A - XVX^T \circ A^c\|_F^2 \tag{5.1}$$

where $\circ$ is the Hadamard product and $A^c = 1 - A$. There are two main advantages of this framework. First, the co-class information is captured in a single objective function that combines both the links and the attribute data. This allows us to observe the relative importance that each has on the class co-occurrence. Second, the weights of the objective function are easily and efficiently solved using linear regression.

It should be noted that while W and V are diagonal matrices other possibilities exist. Diagonal matrices were chosen for two reasons. First, it makes the calculations more efficient. Second, the weights w_0, $\vec{w}$ and $\vec{v}$ clearly represent the relative importance of the links and attributes and are thus descriptive.

To solve for the weights, the first step is to take the derivative of L with respect to w_0 and each w_m, $1 \leq m \leq d$ and each v_m, $1 \leq m \leq d$ (note that the diagonal elements of W and V are represented using w_k and v_k):

$$\begin{aligned}
\frac{\partial L}{\partial w_0} &= -2 \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} \left(y_{ij} - w_0 a_{ij} - \sum_{k=1}^{d} a_{ij} x_{ik} x_{jk} w_k \right. \\
&\quad \left. - \sum_{k=1}^{d} (1 - a_{ij}) x_{ik} x_{jk} v_k \right) \cdot a_{ij} \\
\frac{\partial L}{\partial w_m} &= -2 \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} \left(y_{ij} - w_0 a_{ij} - \sum_{k=1}^{d} a_{ij} x_{ik} x_{jk} w_k \right. \\
&\quad \left. - \sum_{k=1}^{d} (1 - a_{ij}) x_{ik} x_{jk} v_k \right) \cdot a_{ij} x_{im} x_{jm} \\
\frac{\partial L}{\partial v_m} &= -2 \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} \left(y_{ij} - w_0 a_{ij} - \sum_{k=1}^{d} a_{ij} x_{ik} x_{jk} w_k \right. \\
&\quad \left. - \sum_{k=1}^{d} (1 - a_{ij}) x_{ik} x_{jk} v_k \right) \cdot a_{ij} x_{im} x_{jm}
\end{aligned}$$

and set them to zero. We can then arrange these $2d+1$ equations into a matrix-vector multiplication problem $b = Z\vec{q}$ where $q =$ where:

$$q = \begin{bmatrix} w_0 \\ \vec{w} \\ \vec{v} \end{bmatrix} \qquad b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \qquad Z = \begin{bmatrix} Z_{00} & Z_{01} & Z_{02} \\ Z_{10} & Z_{11} & Z_{12} \\ Z_{20} & Z_{21} & Z_{22} \end{bmatrix}$$

The sub sections are built by letting:

$$b_0 = \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij} \cdot a_{ij}$$

$$b_1 = \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij}a_{ij} \cdot x_{im}x_{jm} \text{ for } 1 \leq m \leq d$$

$$b_2 = \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij}(1-a_{ij}) \cdot x_{im}x_{jm} \text{ for } 1 \leq m \leq d$$

$$Z_{00} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2$$

$$Z_{01} = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ik}x_{jk} \cdot a_{ij}^2 \text{ for } 1 \leq k \leq d \text{ and } Z_{10} = Z_{01}^T$$

$$Z_{02} = [0]_{1\times d} \text{ and } Z_{20} = Z_{02}^T$$

$$Z_{11} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2 \cdot x_{im}x_{jm} \cdot x_{im}x_{jm}$$

$$\text{for } 1 \leq k \leq d \; 1 \leq m \leq d$$

$$Z_{12} = [0]_{d\times d} \text{ and } Z_{21} = Z_{12}^T$$

$$Z_{22} = \sum_{i=1}^{n}\sum_{j=1}^{n} (1-a_{ij})^2 \cdot x_{im}x_{jm} \cdot x_{im}x_{jm}$$

$$\text{for } 1 \leq k \leq d \; 1 \leq m \leq d$$

Note that $a_{ij}(1-a_{ij}) = 0$. The weight vector $\vec{q}$ can be computed using Gaussian elimination. The complexity for assembling the b vector and Z matrix is approximately $O(n^2d^2)$ (although it can be sped up if the matrix X is sparse) and $O(d^3)$ for the Gaussian elimi-

nation.

5.2.2 Regularization

To avoid overfitting the weights to the training data, a regularization technique similar to ridge regression can be employed. The regularization penalty term is $\lambda w_0 + \lambda \|W - 1\|_F^2 + \lambda \|V - 1\|_F^2$. The objective function thus becomes:

$$\begin{aligned} L &= \|Y - w_0 A - XWX^T \circ A - XWX^T \circ A^c\|_F^2 \\ &\quad + \lambda w_0 + \lambda \|W - 1\|_F^2 + \lambda \|V - 1\|_F^2 \end{aligned}$$

This will reduce to the original objective function in Equation 5.1 when $\lambda = 0$. As $\lambda \to \infty$, $w_0 \to 0$, $w_i \to 1$ for $1 \leq i \leq d$, and $v_i \to 1$ for $1 \leq i \leq d$. In this case the expression $w_0 A + XWX^T \circ A + XWX^T \circ A^c$ is reduced to XX^T and the class co-occurrence is predicted by attribute similarity only.

After taking the partial derivatives, setting to zero and rearranging the terms, we again have the matrix equation $b = Z\vec{w}$ as above except that λ is added to the diagonal elements of Z and to the elements of b (except b_0).

5.2.3 Label Prediction

Once the weights have been learned it still remains to predict the class of unlabeled nodes. It is assumed that we are given a network with a complete set of links with some nodes labeled and others not. The weights can be learned using the network induced by the labeled nodes. After the weights are learned, they can be applied to two nodes i and j using

$$s_{ij} = w_0 a_{ij} + \sum_{k=1}^{d} a_{ij} x_{ik} x_{jk} w_k + \sum_{k=1}^{d} (1 - a_{ij}) x_{ik} x_{jk} v_k$$

The scoring function s_{ij} can be thought of as a relative measure of the likelihood of i and j having the same class. The matrix $S = \left[s_{ij}\right]_{n \times n}$ then can be thought of as the features

to be input to a classifier. Each row represents a node and its co-class similarity measures with all other nodes. The rows for labeled nodes would be used as the training set and the unlabeled rows being the test set.

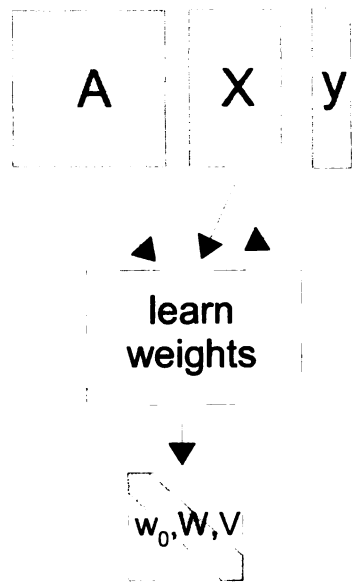


Figure 5.2: Learning the weights

The two problems with this approach are that it does not scale well, since it grows geometrically with the number of sample nodes, and that in tests performed the accuracy is not very high. Since each of the columns in S represents a node, the columns can be summed by the class of each node to create S', an $n \times c$ matrix. The entire process is illustrated by Figures 5.2 through 5.4. First, the scalar w_0 and the diagonal matrices W and V are learned using the adjacency and data matrices for just the labeled instances (Figure 5.2). Then, using the learned weights, the matrix S is calculated as shown in Figure 5.3. In Figure 5.4, the matrix S' is created and then using the labeled rows of S' with the label vector y, a classifier is trained. The class is then predicted for the unlabeled rows using the classifier. Because the matrix S' has a fixed number of columns (c), it grows linearly with the number of nodes in the network.

Note that S has a row and column for each node (labeled or unlabeled) in the network. So, during the summarization step, the columns that are unlabeled are ignored. To improve the performance of the classification we use an iterative algorithm. First, the class of the

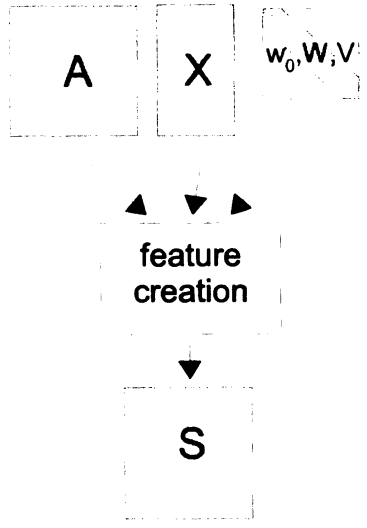


Figure 5.3: Creating the features

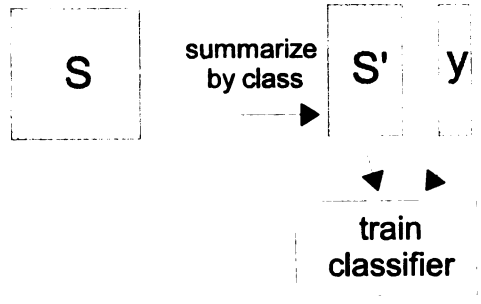


Figure 5.4: Training classifier

nodes is predicted as described above. Then, the predicted labels are used to re-map $S \rightarrow S'$, relearn the classifier and predict new labels. This process continues until the labels converge to a steady state.

5.3 Experimental Evaluation

Data for the experiments was taken from the Cora and CiteSeer data sets from the Linqs [69] website and the teenage friends and lifestyle study data set from Tom Snijder's research group [116]. The Cora dataset represent Machine Learning papers grouped into seven classes (Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning and Theory). From the original set only papers

that cited or were cited by at least one other paper were selected resulting in 2,708 papers. Attributes are represented by the 1,433 unique words (after stemming and removing stopwords) which had a document frequency of ten or more.

The CiteSeer data is a selection of papers from the CiteSeer dataset which have been grouped into six classes(Agents, AI, DB, IR, ML and HCI). The papers and attributes were selected in the same way as the Cora set resulting in 3,312 papers and 3,703 attributes.

The teenage data contains attributes and links of 50 pupils in a school in the west of Scotland. The data were collected at three different time points over a three year period. The attributes describe the social behaviors of smoking, drugs, alcohol, sports involvement and family (a binary attribute on whether or not the number of individuals in the household changed). For the tests alcohol usage was chosen as the class.

The results of our matrix alignment framework are compared against classification of the attributes only and the ICA (iterative classification algorithm) approach from Lu and Getoor's [70, 107] paper. We used logistic regression for both the attributes only and the ICA approach since it yielded the highest accuracy of the classifiers used in their paper. The tests were performed using random splits and 10-fold cross validation.

The results of our tests can be seen in Table 5.1. The first column has the results for prediction using attributes-only, and under the ICA column is the results for the iterative classification algorithm proposed by [70]. The experiments that we ran for these two algorithms were comparable to the results for the same algorithms as reported in [107], so we simply used their results. Our co-class alignment method is in the column labeled CoCA. For the COCA test the standard deviation is also reported.

In the paper by Sen, et al. [107], besides ICA and attributes-only, they also compared a number of other collective classification algorithms (CCA). Nearly all of the CCA algorithms outperformed using only the attributes and ICA either had the highest, or very nearly the highest, accuracy. The authors also pointed out that it was also a relatively fast algorithm. It is for these reasons that we chose to compare ours to ICA.

It can be seen that the accuracy of the CoCA solution is very close to that of ICA for both CiteSeer and Cora. In addition, CoCA is more efficient than ICA. For Citeseer, we

[1]Taken from Sen, et al. [107]

Table 5.1: Results for attributes only, ICA and the co-class alignment algorithm on Citeseer and Cora data sets

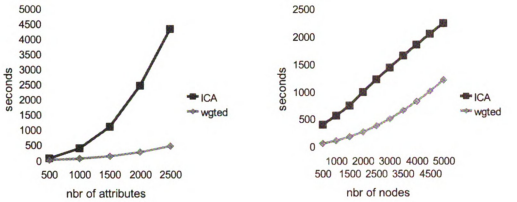
	attr [1]	ICA [1]	COCA
CiteSeer	0.7310	0.7732	0.7667 ± .027
Cora	0.7580	0.8796	0.8774 ± .007
teenYr1	0.6200	0.8000	0.8200 ± .220
teenYr2	0.6600	0.6600	0.7000 ± .282
teenYr3	0.8200	0.8600	0.8600 ± .245

computed the average runtime for ICA to be 11,258 seconds and for matrix alignment, 4,794. For Cora the averages were 1,237 and 714 respectively. For the teenage set the CoCA solution performed as well or better than ICA for all three of the network time points.

To further analyze the running times, synthetic data sets were created, with varying numbers of nodes and attributes. In the first experiment, the number of nodes was held at 500 while the attributes grew from 500 to $2,500$. In the second experiment, the attributes were held at 1000 and the nodes grew from 500 to $5,000$. The results of the run times can be seen in Figure 5.3. Note that as the number of attributes grows the running time of the ICA solution grows at an increasingly faster rate while the weighted solution grows slower and more linearly. As the number of nodes increase, both solutions increase at an apparent linear rate.

To see the relative importance of the links and attributes refer to table 5.2. The top row is the weight for w_0 and the remaining five rows are the top weighted attributes for $\vec{w}$ and $\vec{v}$. For both data sets, when the pairs are linked, the link is much more important than any of the attributes, whose values are close to 1. The attributes of the linked pairs of CiteSeer have more separation between high and low weights than Cora because in Cora, nodes are more likely to link to nodes of the same class. Ordinarily it would be possible to list the names of the significant attribute names but the datasets used in this paper came from the Linqs website and did not have the actual attribute names.

Table 5.3 shows the weights for the attributes of the teenage data set for all three time



(a) 500 nodes and varying attributes (b) 500 attributes and varying nodes

Figure 5.5: Run times with varying nodes/attributes

Table 5.2: w_0 and five highest weights for $\vec{w}$ and $\vec{v}$

Cora		CiteSeer	
w	v	w	v
4.475		4.082	
1.007	2.599	1.123	9.754
1.002	2.501	1.122	9.307
1.002	2.475	1.121	8.470
1.001	2.293	1.118	8.435
1.001	2.147	1.108	8.304

Table 5.3: w_0, $\vec{w}$ and $\vec{v}$ for the attributes of the teenage set

	teenYr1		teenYr2		teenYr3	
	w	v	w	v	w	v
link(w_0)	1.438		1.363		0.554	
drugs	0.155	0.337	0.086	0.105	0.460	0.113
family	1.000	1.000	0.260	0.110	0.887	0.412
smoke	-0.022	0.124	-0.006	0.101	0.351	0.083
sport	-0.001	0.028	0.004	0.037	0.329	0.094

points. In two of the three time points the weights for links are higher than any of the attributes. Family is the highest weighted attribute for both linked and non-linked pairs. Thus it appears that for predicting the alcohol usage in these teens, their friendships are most important followed by their changes in their family. In [88], Pearson, et al., analyzed the data set and found that there was strong evidence for homophily (selecting friends that have similar attributes) among the pupils that used alcohol which lends support to the high values of w_0. The Pearson paper did not directly compare the effect of family on drinking but they did compare both drugs and smoking. They report that drugs have a greater effect on alcohol use than smoking which is also supported by the weights above.

5.4 Conclusions and Future Work

In this chapter, a new collective classification algorithm for networks was proposed, based on the matrix alignment framework. The algorithm is fast and comparable to other CCA methods. Compared to these methods, it has the advantage of capturing the co-class association in a formula using both the links and the attributes. This allows a comparison of the relative weights of both the links and attributes.

Our CoCA algorithm can be extended to make use of the temporal information of an evolving network. The class distribution of the nodes, the link structure and the attributes may change over time which can be exploited to improve the prediction. The framework can capture this using multiple adjacency, co-class and attribute matrices.

For the future, it is expected that even better accuracy can be obtained by extending the framework to utilize community and topological data, like the link prediction formulation did in Chapter 4. Another extension would be to make use of kernel functions as it will be shown for the link prediction formulation in Chapter 6.

Chapter 6

Relationship between Attributes and Links over Time

6.1 Introduction

Social networking is becoming an increasingly active research topic due to its broad appeal to a wide range of disciplines, including physics, social sciences, statistics, and computer science. Much of the early work has focused only on static networks whereas real social networks are often dynamic, with nodes changing their attributes and links while reacting to the changes in other nodes. For example, in an on-line social network such as FaceBook, the nodes (members) change their attributes (e.g. favorite movies), make new links (friends), and join new groups. Dynamic networks pose a significant challenge for network analysts not only because of the modeling complexity, but also the sensitivity of the modeling results on the preprocessing decisions made during the construction of the network.

The objective of this chapter is to study the nature of dynamic networks to better understand the consequences that arise from pre-processing decisions and from other network forces. Specifically studied is:

- the effect of accumulating the link or attribute data over time. For example, if a person is a smoker at one time and then quits, should a model "forget" that information or retain it. The same holds for friends – an argument can be made either way for

retaining links to lost friends.

- how historical data can be effectively utilized. Does the addition of older data improve the accuracy of link prediction? Or is it more important to emphasize the more recent data?

- whether the importance of attributes, in terms of linking behavior, change over time. For example, at some time point, two students might be likely to become friends if they both like some new, obscure sport. As the sport becomes more popular and more students become involved, though, it may become less of a factor in forming friendships.

- how much influence and selection takes place in dynamic networks. Influence refers to the process of people acquiring the attributes of their friends and selection is the name for when people select friends with attributes similar to their own.

As part of the study several metrics are presented to measure the forces of selection and influence from dynamic network data. Although these processes have been extensively studied in the past [3, 23, 88], the metrics presented here are unique in that they allow for measurement of the selection and influence based on changes in the adjacency and data matrices of the dynamic network data.

To measure the relationship between the links and attributes in a dynamic network, a metric called the alignment distance is applied, which was previously developed in Chapter 4 for link prediction in static networks. Recall that in that chapter the framework establishes the alignment between links and attributes using a weighted similarity between the node pairs. In a network with perfect alignment, nodes that are linked to one another will have maximum attribute similarity and those that are not linked will be minimally similar. In this work, it is extended to define a metric to allow for measurements of temporal data. It is further extended to use kernel functions. The kernel version shows improvements in the accuracy due to an enlargement of the weight structure and use of the kernel function.

To study the above issues experiments were run on four dynamic network data sets. The results provide insights into the nature of dynamic networks as well as the effects that pre-processing decisions have on the relationship between attributes and links. Also

demonstrated is the effect of attribute drift, that is, the importance of individual attributes in forming links change over time. In the next section some background on the recent work in dynamic and static networks is provided. Section 6.4 presents the methods used in this study. The data sets are described in Section 6.5 and the results of the experiments are explained in Section 6.6. Finally, concluding remarks are offered in Section 6.7.

6.2 Background

A number of models have been proposed to explain the structure and growth of networks, leading to a number of network characterizations such as regular, random [33], small world [120], scale-free [7], cellular [2, 35], core-periphery[2, 16] and forest fire [66]. All of these models treat link formation as a function of the link structure of the graph and ignore the effects of attributes.

Several generative models have been proposed for networks that do incorporate both the link structure and the attributes. Taskar et al., developed a relational Markov network approach to infer missing node class [114] or links [115]. Neville and Jensen's [78] latent group model similarly learns conditional probabilities for attributes, links and groups. These models are designed specifically for static networks. The exponential random graph model is another well-known network modeling approach [111, 119]. These models are defined using network statistics such as transitivity and reciprocity. None of these studies consider the effect of preprocessing decisions during network construction.

Link prediction in a network is another active area of study. Liben-Nowell and Kleinberg [68] presented a model using only features derived from link topology. Al Hasan, et al. [47], O'Madadhain, et al. [85], and Popescul, et al. [89] proposed applying classification methods for link prediction using attributes and link-based features. Lahiri and Berger-Wolf [63] use frequent subgraphs to predict *when* a particular event (or link) will take place. Finally, Rattigan and Jensen [92] described the class skew problem with link prediction and suggested an anomaly detection-type approach to predicting interesting links. All of the attribute-based models described above are designed only for static networks.

Some recent studies have specifically targeted temporal network data. Hanneke and Xing [46] proposed an exponential random graph model that uses network statistics over multiple time periods. Kempe et al. [58] consider networks with explicit time-ordering of their edges. The paper by Guestrin et al. [45] uses a first-order Markov assumption to model a dynamic network. Sharan and Neville [109] uses kernels to summarize dynamic networks and then applies a relational classifier on the summarized graph.

Using kernel functions to improve accuracy has been an active area of research recently. Cristianini, et al. [25] introduced the notion of using kernels to align two kernel functions as a measure of a similarity between the two kernels. Three different groups [8, 10, 86] proposed using kernels to classifiy instances of paired data using a Kronecker product. Kashima, et al. [56] proposed a more efficient method of prediction by replacing the Kronecker product with a smaller, Cartesian matrix.

6.3 Preliminaries

A temporal network is considered to be a series of periodic snapshots of the network. Each snapshot is a static network of nodes and links, where the nodes have associated descriptive attributes. Notation is defined, first for static networks and then for dynamic ones.

Consider a physical or social network represented as a graph $G = (V, E)$, where $V = \{1, 2, ..., |V|\}$ is the set of nodes and $E \subseteq V \times V$ is the set of links. Let $A = [a_{ij}]_{n \times n}$ denote an adjacency matrix representation of the graph, where $n = |V|$, $a_{ij} = 1$ if there is a link between nodes i and j and zero otherwise. In this chapter, it is assumed the network contains undirected links, which means A is a symmetric non-negative matrix. Also let $X = [x_{ik}]_{n \times d}$ denote its corresponding data matrix, where d is the number of attributes and x_{ik} is the k^{th} attribute value for node i. Assume that the row vectors in X have been properly normalized (to unit length) or standardized (to have zero mean and variance equals to one) so that the matrix product XX^T (where X^T is the transpose of X) is equivalent to either the cosine similarity or correlation between every pair of nodes.

The dynamic networks are considered to be τ snapshots of the network at discrete

time steps. The network at time t, would be represented by the adjacency matrix $A^{(t)}$ and the data matrix $X^{(t)}$. We let $x_{ik}^{(t)}$ be k^{th} attribute for node i at time t and $x_i^{(t)}$ be the row vector of attribute values for node i at time t. Thus, $x_i^{(t)} x_j^{(t)T}$ is the similarity between the attribute vectors of nodes i and j at time t.

6.3.1 Selection and Influence

Selection (also called homophily) and influence (also called assimilation) are concepts from the area of social network analysis that represent forces within a social network [23]. Selection is the process where people choose friends with whom they have a lot in common, while influence is the process where people change their attributes to match those of their friends.

Most literature defined these concepts with respect to a specific node (or actor)-pair and a specific attribute [74, 88]. For example, it can be said that *Eric* exhibited selection because he befriended *Jill* who enjoys the same sports activities as Eric. In contrast, Crandall, et al. [23] defined these concepts in terms of the overall (cosine) similarity for all attributes. They also studied the network evolution in terms of its recorded activities instead of the discrete snapshots of its adjacency and data matrices. They examined the feedback effect between social selection and influence, in which social interactions are initially formed between similar nodes, but the interaction would further increase its similarity.

For many network data sets, it is not possible to follow the progression of changes in attribute values since the network data is gathered only at periodic intervals. As in [23] the similarity is measured using the cosine metric, but the effect of selection and influence is evaluated based on changes in the adjacency and data matrices.

Definition 1 *The selection process in a dynamic network* $G = \{(A^{(1)}, X^{(1)}), \cdots, (A^{(\tau)}, X^{(\tau)})\}$ *is measured as follows:*

$$Selection = \frac{p(a_{ij}^{(t)} = 1 | a_{ij}^{(t-1)} = 0, x_i^{(t-1)} x_j^{(t-1)T} > \epsilon)}{p(a_{ij}^{(t)} = 1 | a_{ij}^{(t-1)} = 0)}$$

where the denominator is the conditional probability an unlinked pair will become linked and the numerator is the same probability for unlinked pairs whose similarity exceeds the threshold ϵ. Values greater than one indicate the presence of selection.

Definition 2 *The influence process in a dynamic network* $G = \{(A^{(1)}, X^{(1)}), \cdots, (A^{(\tau)}, X^{(\tau)})\}$ *is measured as follows:*

$$influence = \frac{p(x_i^{(t)} x_j^{(t)T} > x_i^{(t-1)} x_j^{(t-1)T} | a_{ij}^{(t-1)} = 0, a_{ij}^{(t)} = 1)}{p(x_i^{(t)} x_j^{(t)T} > x_i^{(t-1)} x_j^{(t-1)T} | a_{ij}^{(t-1)} = 0)}$$

where the numerator is the conditional probability that similarity increases from time $t-1$ *to* t *between two nodes that became linked at time* t *and the denominator is the probability that the similarity increases from time* $t-1$ *to* t *between two nodes that were not linked at time* $t-1$. *As with selection, values greater than one indicate the presence of influence.*

6.3.2 Research Motivation

A static network can be easily formed by accumulating the links and attribute information for all the nodes since the start of the data collection period. Building a dynamic network is more challenging because an analyst must make many preprocessing decisions. To our knowledge, the effects of these decisions has not yet been extensively studied. In this chapter, the impact of such decisions is examined on subsequent network analysis

First, the effect of accumulating the link or attribute data over time is examined. Accumulating link data means any links established in an earlier snapshot will remain in the future snapshots of the network even though the node pairs may not interact with each other. The same applies to accumulating attribute data. There may be times when the analyst can choose whether or not to accumulate, but at other times, the data may be accumulated by default. For example, consider an online social network where members are initially asked to enter information such as musical preferences during registration time but often do not bother to change them. In this scenario, the attributes can be thought of as accumulated by default.

Second, in the adaptive modeling of dynamic networks, a predictive model has to consider the number of previous snapshots to use for making its future prediction. Should

the model utilize only the recent data or should it incorporate older history? For efficiency reasons, some models would limit the number of previous snapshots to consider. In the results it is shown that employing a first-order Markov assumption is often sufficient for many of the data sets.

In addition to studying these decisions, the effect of selection and influence is also examined in network data using the measures defined in the previous section. Although these processes have been studied previously, the measures presented here are unique in that they allow for measurement of the selection and influence in snapshot data using a single metric rather than producing a separate value for each attribute. This is useful for networks with thousands of attributes.

Also measured is the amount of attribute drift in the dynamic networks. When predicting the formation of links using the attributes, some attributes will be more predictive. As the network changes over time, the predictive power of attributes also changes, which is referred to here as attribute drift. The presence of attribute drift along with evidence of the validity of the Markov assumptions leads to the conclusion that attribute drift is at least partly responsible to the diminished value of historical data.

6.3.3 Link Prediction: Missing vs. Future Links

There have been two different approaches to the link prediction problem, either predicting missing links within a static network or by using historical data, predict new links. Chapter 4 considered the missing link version, whereas this chapter considers predicting future links. It would be interesting to compare how algorithms perform on the two versions of the problem. However this is inherently troublesome because of the fundamental differences between the two approaches.

In predicting future links, the training set can contain all of the node-pair data from the previous (several) network snapshots. For missing links, however, the training set cannot contain the data for the missing pairs. Predicting future links is therefore sensitive to the rate of change in the links. A data set where links change often would make predicting future links more difficult. This has no effect on predicting missing links. On the other hand predicting missing links is sensitive the number of missing pairs which has no effect

on the future link version. For these reasons, this dissertation does not attempt to compare the performance of algorithms on these two versions of the problem.

6.4 Temporal Alignment

In Chapter 4, a matrix alignment framework was presented that uses a set of weights to determine the important attributes for establishing links between nodes. Essentially, the goal is to learn a set of weights $\vec{w} = \{w_1, ...w_d\}$ that minimizes the objective function

$$\min_W L = \|A - XWX^T\|_F^2, \tag{6.1}$$

where the diagonal elements of W correspond to $\vec{w}$ and $\|\cdot\|_F$ denote the Frobenius norm. The expression $\|A - XWX^T\|_F^2$ can be considered as a distance measure between the links and the attribute similarity between node pairs. Smaller values of this alignment distance imply a higher degree of alignment between the attributes and links. Experiments have shown that, in general, smaller alignment distances imply more accurate link prediction (given that the link prediction technique uses attributes). Next we modify the alignment distance to allow for measurements of temporal data by substituting adjacency and data matrices from different time periods.

6.4.1 Incorporating Temporal Information

To test the value of using the historical data the objective function above needs to be modified to incorporate more than a single snapshot. Specifically, using multiple time periods to learn a single set of weights was formulated as follows:

$$L = \sum_{t=1}^{\tau} \|A^{(t)} - X^{(t-1)}WX^{(t-1)T}\|_F^2,$$

where τ is the number of network snapshots available. To avoid overfitting, a regularization technique is employed by adding a penalty term $\lambda\|W - I\|_F^2$ to the objective function:

$$L = \sum_{t=1}^{\tau} \|A^{(t)} - X^{(t-1)}WX^{(t-1)T}\|_F^2 + \lambda\|W - I\|_F^2. \tag{6.2}$$

This will coerce the weight vector $\vec{w}$ to ones for high values of λ, which is equivalent to assigning equal importance to all the attributes. To solve for the weights the first step is to take the partial derivatives with respect to W:

$$\begin{aligned}\frac{\partial L}{\partial W} &= -2X^{(t-1)T}A^{(t)}X^{(t-1)} \\ &+ 2(X^{(t-1)T}X^{(t-1)})W(X^{(t-1)T}X^{(t-1)}) + 2\lambda W - 2\lambda I\end{aligned}$$

Setting the derivative to zero and rearrange the terms, we obtain a system of linear equations, $Z\vec{w} = b$, by letting

$$\begin{aligned}b_m &= \sum_{t=1}^{\tau}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{(t)} \cdot x_{im}^{(t-1)}x_{jm}^{(t-1)} + \lambda \\ Z_{mk} &= \sum_{t=1}^{\tau}\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ik}^{(t-1)}x_{jk}^{(t-1)} \cdot x_{im}^{(t-1)}x_{jm}^{(t-1)}\ ,\ m \neq k \\ Z_{mk} &= \sum_{t=1}^{\tau}\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ik}^{(t-1)}x_{jk}^{(t-1)} \cdot x_{im}^{(t-1)}x_{jm}^{(t-1)} + \lambda\ ,\ m = k\end{aligned}$$

The weight vector $\vec{w}$ can be computed using Gaussian elimination or the conjugate gradient method.

6.4.2 Alignment Distance

In the experiments section, a number of different distance metrics and weights are used. To make those discussions clearer we give a few definitions here. The general alignment distance is the same as the objective function given in Equation (6.2). To compute the distance using matrices from different timestamps we define the following alignment functions:

- $dist_0(t) = \|A^{(t)} - X^{(t)}WX^{(t)T}\|_F^2$
- $dist_{+1}(t) = \|A^{(t+1)} - X^{(t)}WX^{(t)T}\|_F^2$
- $dist_{-1}(t) = \|A^{(t)} - X^{(t+1)}WX^{(t+1)T}\|_F^2$

$dist_0(t)$ measures the alignment distance between the links and attribute similarity for a snapshot t. Since $dist_{+1}$ calculates the alignment distance between the links at time $t+1$

and the attributes at time t, it provides a measure of the future linking behavior of nodes. In contrast, $dist\_1$ is reflective of changes to future attribute values.

We also adopt the following notation to indicate the different ways in which the weights are calculated:

- $\vec{w}$: weights calculated using $A^{(t)}$ and $X^{(t)}$ only.
- $\vec{w}_{+c}$ =weights calculated using $A^{(t)}, X^{(t-1)}, A^{(t-1)}, X^{(t-2)}, ..., A^{(t-c)}, X^{(t-c-1)}$.
- $\vec{w}_*$ =weights calculated using $A^{(t)}, X^{(t-1)}, ..., A^{(2)}, X^{(1)}$.

6.4.3 Link Prediction

Once the vector $\vec{w}$ has been learned, the expression $x_i^{(t)} W x_j^{(t)T}$ is a weighted similarity score that provides a relative measure of the likelihood of nodes i and j forming a link in time step $t+1$. As in Chapter 4, predictions are made using a pair of quadratic discriminators, g_0 and g_1. g_0 is trained on the weighted similarity scores for each pair of nodes in the training set that are not linked, and g_1 is trained on the weighted scores for linked training set pairs. Each pair in the test set is predicted to be a link/non-link depending on which discriminator g_1/g_0 is higher using the score calculated for the pair.

6.4.4 Kernel Approach to Link Prediction

The approach to link prediction above that uses the weight $\vec{w}$ has two limitations. The first is that only the diagonal values of the matrix W are used. These weights capture the relative weights for the attributes that a pair of nodes have in common but not the *cross* product of their attributes. For example, say that in a campus social network, pairs of students are likely to become linked if they major in music or in drama. But there could also be a high likelihood of a pair becoming linked where one is a drama major and the other a music major. The weight vector approach ignores this cross product.

The other limitation is that it is assumed that there is a linear relationship between the attributes and the links. It is possible that the relationship is of a higher dimension.

Recently, kernel functions have been gaining attention for converting linear solutions into non-linear ones. The classic example is the kernel trick used in the SVM classifier [24, 97].

The kernel approach presented here removes both limitations. Using the full matrix $W = [w_{ij}]_{d \times d}$, the objective function from equation 6.1 is essentially the same:

$$\begin{aligned} \mathcal{L} &= \|A - XWX^T\| + \lambda\|W - I\| \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij} - \sum_{k=1}^{d}\sum_{l=1}^{d} x_{ik}w_{kl}x_{lj}^T)^2 + \lambda\sum_{k=1}^{d}\sum_{k=1}^{d}(w_{kl} - \delta_{kl})^2 \end{aligned}$$

where $\delta_{kk} = 1$ and zero otherwise. The partial differential with respect to w_{pq} becomes:

$$\frac{\partial \mathcal{L}}{\partial w_{pq}} = \sum_{i=1}^{n}\sum_{j=1}^{n} 2\left(a_{ij} - \sum_{k=1}^{d}\sum_{l=1}^{d} x_{ik}w_{kl}x_{lj}^T\right)(-x_{ip}x_{qj}^T) + 2\lambda w_{pq} - 2\lambda\delta_{pq}$$

Now it can be set equal to zero:

$$\lambda w_{pq} = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{pi}^T a_{ij} x_{jq} - \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{d}\sum_{l=1}^{d} x_{pi}^T x_{ik} w_{kl} x_{lj}^T x_{jq} + \delta_{pq}$$

$$w_{pq} = \frac{1}{\lambda}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} x_{pi}^T v_{ij} x_{jq}\right) + \delta_{pq} \tag{6.3}$$

where

$$v_{ij} = a_{ij} - \sum_{k=1}^{d}\sum_{l=1}^{d} x_{ik}w_{kl}x_{lj}^T \tag{6.4}$$

Now we can substitute Equation 6.3 into Equation 6.4:

$$v_{ij} = a_{ij} - \frac{1}{\lambda}\sum_{k=1}^{d}\sum_{m=1}^{n}\sum_{z=1}^{n}\sum_{l=1}^{d} x_{ik}x_{km}^T v_{mz} x_{zl} x_{lj}^T + \sum_{k=1}^{d}\sum_{l=1}^{d} x_{ik}\delta_{kl}x_{lj}^T$$

Notice that each term of the last equation represents the ij^{th} cell of an $n \times n$ matrix. So the terms can be gathered into matrix form as $V = A - \frac{1}{\lambda}XX^TVXX^T + XIX^T$ (where I is the identity matrix) and rewritten as:

$$KVK + \lambda V = \lambda(A + K) \tag{6.5}$$

using $K = XX^T$. Since this depends on the dot product of the X matrix, K can be replaced with the dot product of any function of X: $K_{ij} = \phi(X_i) \cdot \phi(X_j)^T$. Note that using the linear kernel $K = XX^T$ reduces to the original formula. Experiments in Section 6.6 used both the linear kernel and the RBF kernel, $K(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$.

Instead of W, it is necessary now to solve for the $n \times n$ weight matrix V. This will be done using Schur decomposition. In Schur decomposition, a square matrix is decomposed as $K = UDU^T$ where U is orthogonal and D is an upper triangular matrix. If K is symmetric then D is diagonal. Since U is orthogonal, $UU^T = I$ and so

$$\begin{aligned} K &= UDU^T \\ U^T K U &= U^T U D U^T U \\ U^T K U &= D \end{aligned}$$

To solve for V:

$$\begin{aligned} KVK + \lambda V &= \lambda(A + K) \\ KUU^T VUU^T K + \lambda V &= \lambda(A + K) \\ U^T KUU^T VUU^T KU + \lambda U^T VU &= \lambda U^T (A + K)U \\ D\tilde{V}D + \lambda \tilde{V} &= F \end{aligned}$$

where $F = \lambda U^T(A+K)U$ and $V = U^T VU$. Because D is diagonal, the ij^{th} element of the first term will be $d_{ii}d_{jj}\tilde{v}$ and so the elements of $\tilde{V}$ can be solved with the following formula:

$$\tilde{v_{ij}} = \frac{f_{ij}}{d_{ii}d_{jj} + \lambda}$$

and the matrix V can be recovered using $V = U\tilde{V}U^T$. The weights V can be solved for by using $A^{(t)}$ and $K^{(t)}$ from time t. Then using the weights and $K^{(t+1)}$ the links $A^{(t+1)}$ can be predicted using Equation 6.5.

In the experiments that were performed, after V was calculated, the equation $\tilde{A}^{(t)} = \frac{1}{\lambda}K^{(t)}VK^{(t)} + V - K^{(t)}$ was very accurate at reproducing the original $A^{(t)}$, i.e. the link values were close to one and the non-links close to zero. However when using the V that was learned from $X^{(t)}$ with the data from the next snapshot $X^{(t)}$, the values were not as well behaved. There was still a good separation between the links and non-links but they were not necessarily centered around one and zero respectively. Having a good separation is fruitless unless a proper boundary is known. However, if the data is centered the predicted adjacency matrix $\tilde{A}^{(t+1)}$ should have link and non-link values close to one and zero respectively.

Centering [1] a vector y is simply a matter of subtracting the mean, $y - \bar{y}$. A centered row of X then would be: $\hat{x}_i = x_i - \mu = x_i - \frac{1}{d}eX$ where e is a d length vector of 1's and μ is a row vector of means. Then

$$\hat{X} = X - \frac{1}{d}e\mu X = X - \frac{1}{d}e^T eX = (I - \frac{e^T e}{d})X$$

The problem is that this cannot be done directly in the feature space of $\phi(X)$. However, XX^T can be centered, $\hat{X}\hat{X}^T = (I - \frac{e^T e}{d})XX^T(I - \frac{e^T e}{d})^T$, which can be applied to any kernel:

$$\hat{K} = (I - \frac{e^T e}{d})K(I - \frac{e^T e}{d})^T$$

The weight matrix V can be learned from the centered kernel matrix $K^{(t)}$ and $A^{(t)}$. Then the adjacency matrix $A^{(t+1)}$ can be inferred from V and the data matrix $K^{(t+1)}$. The threshold $1/2$ can then be used to separate the links from the non-links.

6.5 Data Set Descriptions

The data sets used in this paper are described below. In each case the raw data was processed to create binary attributes and in some cases to limit the number of nodes to the most active ones.

[1] The lecture notes from Dr. Bennett (http://www.rpi.edu/ bennek/) of RPI were very helpful in the discussion on centering.

The teen data set was built from the teenage friends and lifestyle study taken from the Siena [116] website. The study followed changes in the friendship relations of students in the West of Scotland starting in 1995, recording information about their drug, smoking, and drinking habits, their sport participation and changes in their families.

The wiki data is created from articles contributed by volunteered editors on the Wikipedia website. Editors also maintain their own pages where they can leave messages for other editors to discuss differences of opinions and other subjects. The English version was downloaded and processed the data to build a network using editors as nodes and pages that they edited as attributes. Links were established based on editors leaving messages on other's pages. Of the hundreds of thousands of editors and millions of web pages, the 3900 most active editors (those who edited more than 24 pages) and the 5,027 most edited pages (those with over 90 edits) were extracted.

The DBLP bibliography website catalogs the publishing activity of computer science researchers. From the data, a network can be built using the authors as nodes and their coauthor relationships as links. Three separate networks were created using conferences associated with database (dblp1), artificial intelligence (dblp2) and computer networks (dblp3). For attributes, selected keywords from the paper titles were used. The eight time periods were based on yearly scans from 1997 to 2004.

The levant data set was constructed from data downloaded from the KEDS website [57]. It contains information gathered from twenty five years of Reuters news stories relating to countries located in the Levant (eastern Mediterranean area). The actors in the Levant set represent countries, individuals and organizations related by one of many different relationship types. The relationships are broadly grouped into twenty different types, such as *provide aid*, *engage in material cooperation* and *threaten*. We constructed an adversarial network by using two types of relationships — *fight* and *attack with WMD* — to form the links. Linked nodes are those who are in some way enemies of each other. The other 18 relationship types are used as node attributes based on participation. For example, a node that *cooperates materially* with at least one other node is assigned the attribute "cooperates materially". In this adversarial network, link prediction is the task of inferring which nodes will attack each other given their other activities.

Table 6.1: General Network Statistics

	teen	wiki	dblp1	dblp2	dblp3	levant
τ	3	8	8	8	8	25
d	5	5027	613	613	613	18
$n^{(1)}$	50	445	473	336	57	130
$\|E^{(1)}\|$	158	864	1309	996	137	308
$dist_0/n^{(1)}$	3.13	1.08	0.83	1.00	0.23	2.10
$n^{(\tau)}$	50	2344	1483	1410	878	129
$\|E^{(\tau)}\|$	167	7684	5403	5974	3074	397
$dist_0/n^{(\tau)}$	3.31	2.60	1.87	1.99	1.48	2.05
$\|XX^T \circ A\|_F^2$	0.7	0.06	0.84	0.88	0.87	0.5
$\|XX^T \circ A^c\|_F^2$	0.56	0.01	0.03	0.04	0.04	0.29

τ=nbr of periods, d=nbr of attributes

$n^{(t)}$=nbr of nodes at time t, $|E^{(t)}|$=nbr of edges at time t

$|XX^T \circ A|_F^2$=similarity of linked nodes ($\circ$ is the Hadamard product)

$|XX^T \circ A^c|_F^2$=similarity of unlinked nodes ($A^c = 1 - A$)

The diversity of the dynamic networks can be seen from the statistics listed in Table 6.1. In terms of their sizes teen and levant have fewer numbers of both nodes and attributes. Besides being larger sets, wiki and dblp also have more attributes than nodes. Next we examine the stability by measuring the changes in the network. The teen set starts and ends with the same number of nodes with a slight increase in the number of links. The levant set gains and loses nodes with fairly active changes in the links. The dblp sets have steady growth in the number of nodes and links while the wiki set experiences dramatic growth.

Another way to compare the sets is by alignment. Notice the average alignment ($dist_0/n$) for both the first and last snapshot. The teen set is has the highest alignment distance while dblp has the lowest. Also, the alignment changes little for teen and levant but more so for dblp and wiki. The poor alignment for teen can be explained by the fact that there are too few attributes to explain the friendship behavior. For dblp, because

the attributes are created from words in paper titles and the links are built from the co-authorship relations of the same papers , it is not surprising that the alignment would be good. However, the alignment gets worse over time for dblp and wiki due to the added complexity of the network as more nodes and links are introduced.

Another characteristic to consider is the differences in the similarity between linked and unlinked nodes. This can be done using the squared norm of Hadamard product of the similarity and the adjacency matrix $|XX^T \circ A|_F^2$ and the complement of the adjacency matrix $|XX^T \circ A^c|$. In a network where the attributes are very predictive of the links we would expect to see a large difference in these two statistics. Looking at the last two rows of the table, it can be seen that the similarities for teen and dblp are high for linked nodes while levant is slightly lower and wiki is quite low. The difference between the linked and unlinked is quite low for teen and levant, medium for wiki and dramatically high for dblp. Again for dblp, the difference is explained by the way the network is built. In the wiki set, the large number of attributes results in an overall low similarity even between linked nodes.

6.6 Experiments

The experiments in this section explore the effects of pre-processing decisions and network forces such as selection and influence on network analysis. Each data set is composed of several snapshots of static network states that include an adjacency and a data matrix. For the teen and levant sets, the data was already processed into yearly chunks. Each paper in the dblp data set is labeled with the year of publication, so it was natural to break them into yearly snapshots. The wikipedia data has date stamps associated with each action so any size interval could be chosen for the snapshots. For Sections 6.6.1 through 6.6.5, a window size of six months was used. In Section 6.6.6, the experiments are repeated using a one month window size.

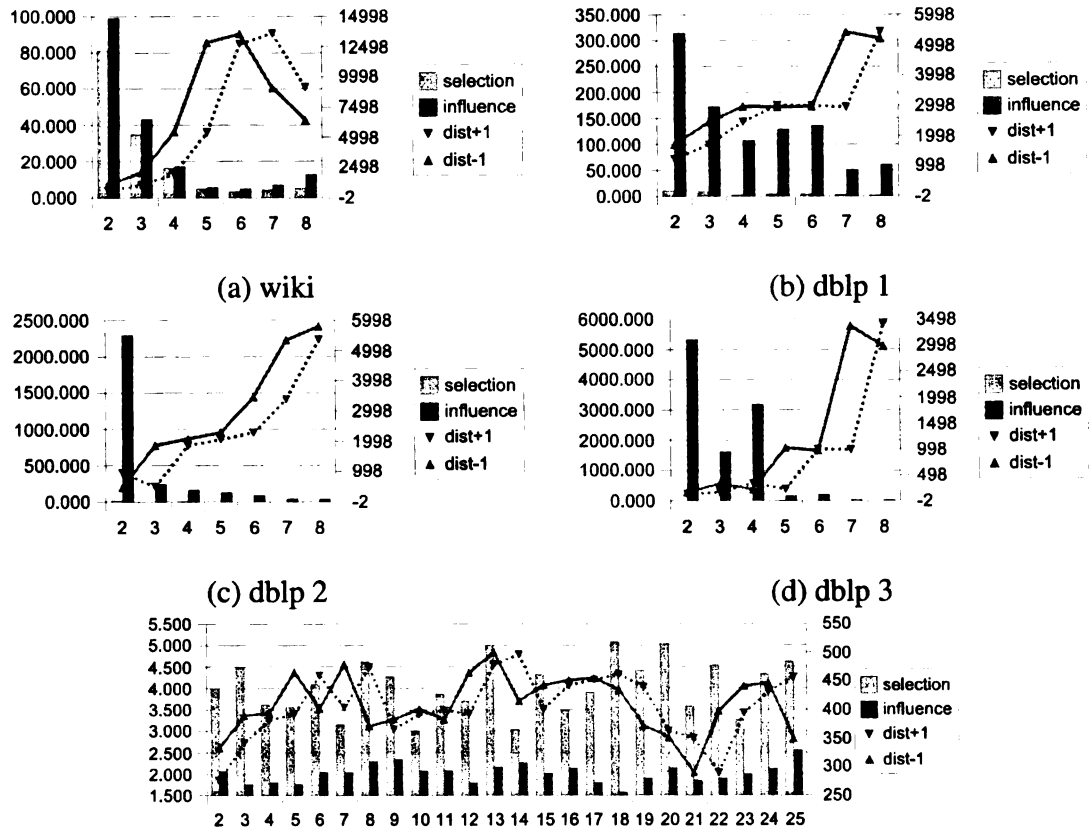


Figure 6.1: Selection and influence

6.6.1 Selection and Influence Experiments

Recall that selection is the process of forming links between pairs of nodes with common attributes and that influence is the process of changing one's attributes to be more like one's linked nodes. The purpose of this experiment is to find out the extent of selection and influence in the data sets and how these forces compare to the measurements of $dist_{+1}$ and $dist_{-1}$. Using the formulas given in Section 6.4, we calculated these metrics for all the data sets. For selection $\epsilon = 0.9$ was used.

The results in Figure 6.1 show bars for the measurements of selection and influence and lines for the values of $dist_{+1}$ and $dist_{-1}$. The bars are scaled to the left axis and the lines are scaled to the right axis. The charts show the values calculated between two snapshots so the bars labeled 2 are for the period between snapshots 1 and 2.

For the measures of selection and influence, values above 1 are evidence that these forces are present. In the wiki set, the presence of both selection and influence are very strong at first, taper off dramatically in the middle periods but then rebound slightly towards the end. To see why the statistics dropped so dramatically, recall from Table 6.1 that there was a very large growth in the number of nodes. The conjecture is that in the early years of Wikipedia there were fewer editors and more core pages. As the site matured, larger numbers of editors worked on more specialized pages. So it became less likely that two editors that became linked had many common attributes and that two linked editors would continue to work on similar pages.

In dblp, there is strong evidence for influence but much weaker evidence for selection. To explain this, recall that the links are formed by co-authorship and the title keywords provide the attributes. In many cases, two authors with similar keywords still have a low probability of collaborating which explains the low selection. On the other hand, linked authors often continue to collaborate increasing their similarity leading to a high influence score.

In levant, there is strong evidence for selection but weaker evidence for influence. There is some intuition for this as countries, or organizations may select each other to fight based on similar threatening and aggressive expressions. And while fighting parties might accelerate their expressions of aggressiveness it is likely that by the time they actually are fighting that the similarity of their expressions is already quite high and so less likely to increase.

Reviewing the formulas in Section 6.4.2 we would expect there to be an inverse relationship between selection and $dist_{+1}$ and between influence and $dist_{-1}$. This is borne out in the charts where there is often an increase (decrease) in influence for every decrease (increase) in $dist_{-1}$.

In general, all sets display at least some evidence of selection and/or influence. The values fluctuate over time which suggests a changing linking strategy. We explore these changes in the section below on attribute weight drift.

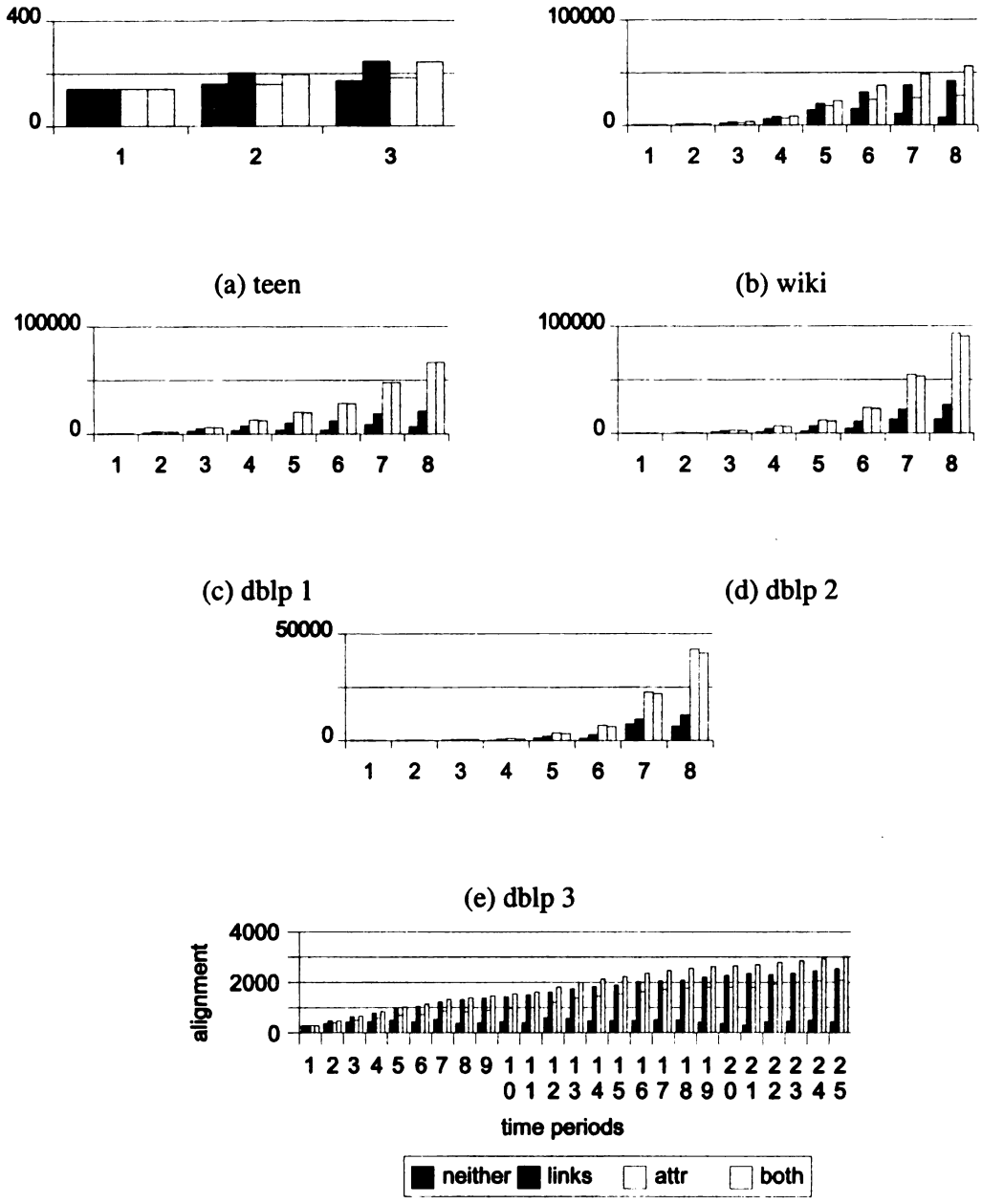


Figure 6.2: Effects of accumulation

6.6.2 Effect of Accumulation

This experiment is designed to learn how the alignment is affected when different accumulating strategies are used. To accumulate links means that a link established on one

timestamp persists. Considering bibliographic sets, an argument can be made that just because two authors did not collaborate in time $t + 1$ but did in t does not mean that they are no longer considered collaborators. Attributes can also be accumulated and a similar argument can be used with bibliographic sets.

For each of the data sets four tests were run, one with no accumulation, one where only the links are accumulated, another for which only the attributes are accumulated and a last that accumulates both links and attributes. For each of the sets the distance $dist_0$ for each snapshot was calculated.

Figure 6.2 shows the results. The four strategies above are grouped together by snapshot with the bars representing the alignment distance $dist_0$. The larger the value, the greater the distance between the alignment of the links and attributes. With all six data sets, the alignment distance grows larger with time because generally, more nodes are added to the networks over time. With other factors remaining stable and additional nodes, the alignment can only get larger.

With all data sets the lowest distance results from accumulating neither the links or attributes. In most instances, accumulating both links and attributes results in the highest distance. In teen, wiki and levant, accumulating just the links results in a higher distance than accumulating just the attributes; with dblp the situation is reversed. The important message in this experiment is that the accumulated networks have a higher distance than non-accumulated.

By accumulating the links, the network becomes less aligned. This suggests that as nodes change their attributes and links, they do so purposefully. Links are dropped for a reason that is related to the changes in attributes the node has made. The implications are important for preprocessing data. In some social networks they can be *implicitly* accumulated, like when users do not often change certain interests (like movie preferences). The network can possibly be better aligned if the analyst tries to remove old data values.

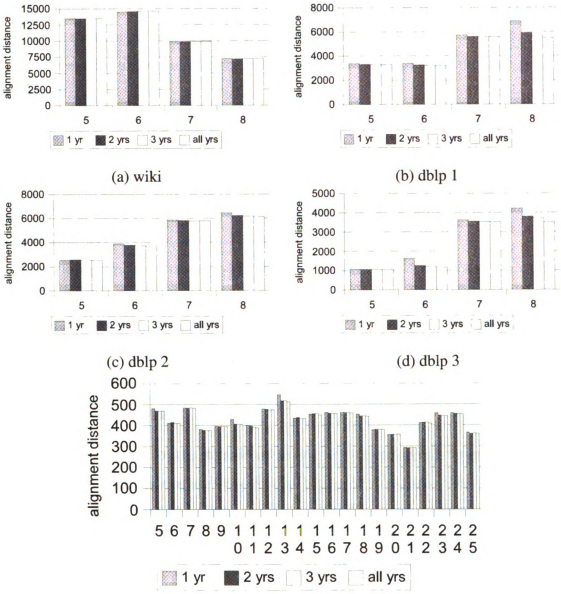


Figure 6.3: Alignment distance comparisons

6.6.3 Learning from History

This experiment attempts to discover the value of using historical data in a dynamic network. The general idea is to learn different sets of weights for increasing window sizes of historical data and use those weights to find the alignment distance for a future time period. While any of the alignment measures could have been used, $dist_{+1}$ was chosen since one of our primary interests is in the area of link prediction.

For the experiment the weights $\vec{w}_{+1}$, $\vec{w}_{+2}$, $\vec{w}_{+3}$ and $\vec{w}_{*}$ were learned to calculate $dist_{+1}$ for the time periods from t to $t+1$. The results can be seen in Figure 6.3. The bars represent the alignment distance values for different weights for the time periods predicted.

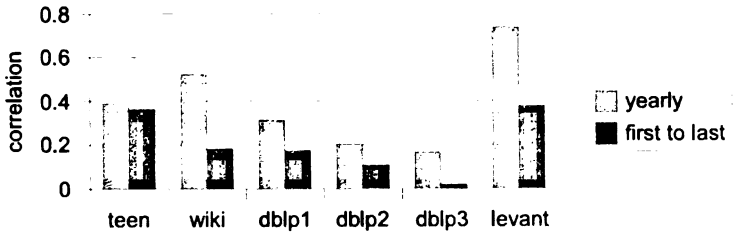
In all of the data sets, alignment distance generally becomes lower (better) or stays the same with the addition of older data. With both wiki and levant there was no or very little improvement. For these sets, the Markov assumption of using only the most recent data appears to be appropriate. In the dblp sets, the alignment improves only marginally in most time periods except year 8 of dblp1 and years 6 and 8 of dblp3.

Many link prediction models assume that only the most recent snapshot of the network is adequate for predicting the state of the next snapshot. The results support this assumption for some networks. In networks where past data is helpful, it could be due to the drift in attribute weights. Examples are provided in the next section to support this suggestion.

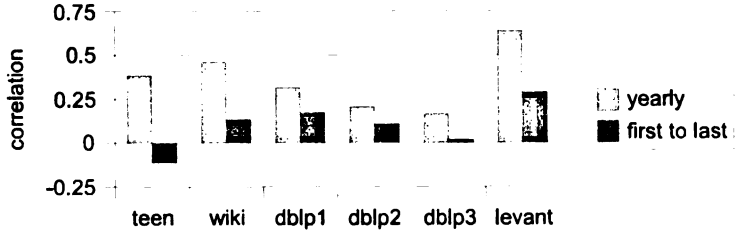
6.6.4 Attribute Drift

A possible explanation for the diminished value of historical data is the possibility that over time, the attributes that are important to linking will change. This was tested by learning the weights for each attribute and then calculating the correlation coefficient between them for each time period. The correlation was calculated for all adjacent time periods and also between the first and last time period. The results in Figure 6.4 compare the average of the period-to-period correlations with the correlation between the first and last period.

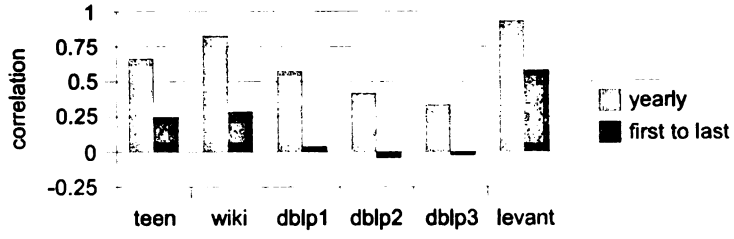
Not surprisingly, the correlation over an extended period of time is very low but is better for the average period-to-period correlation. This shows that within a network the weights of the attributes drift over time but the drift is somewhat gradual. In Figure 6.4(a), the correlation for all three dblp sets is low even for the period-to-period averages. In a network, like dblp, where the importance of attributes for linking changes more often, one would expect that just using one prior period would not be sufficient for predictive tasks. In these networks the Markov assumption may not be as appropriate as those where the drift is less significant. This is supported by the results in the previous section where



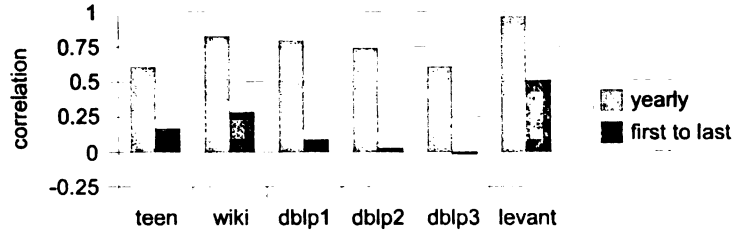
(a) neither attributes nor links are accumulated



(b) links are accumulated but attributes are not



(c) attributes are accumulated but links are not



(d) both the links and the attributes are accumulated

Figure 6.4: Correlation between attribute weights. Yearly refers to the average of the correlations between adjacent years and first to last is the correlation between the weights for the first and last periods.

alignment improves using past data for dblp1 and dblp3.

Examining the drift with respect to accumulating links and attributes, it can be seen in Figures 6.4(a)–(d) that all of the sets have higher yearly correlation when the links are accumulated. Comparing charts (a) to (b) and (c) to (d) it appears that (with the exception

of levant) accumulating attributes does not change the yearly correlation very much.

Clearly, there is a tendency for attributes to change their importance for linking. For social networks, this implies that while selection and influence take place, the attributes that encourage selection or those that are changed through influence, tend to be different over time. In other words, qualities that someone looks for in a friend today might be different than the ones they look for next year.

The set with the most consistently high first-to-last correlation was levant data which has only 18 attributes. This is reasonable since the attributes in this set are positive or negative actions that the actors can take, such as *offer assistance* or *threaten*. As the links are determined by actors who *fight*, it is unlikely that the attribute weights would drift much over time.

6.6.5 Link Prediction Experiments

In addition to investigating the impact of using historical data on the theoretical measure of alignment it also necessary to consider the practical issue of link prediction. In this section the result of changing window size of historical data when predicting new links is tested. Accordingly, different weights were applied – learned with differing amounts of historical data – to the link prediction method described in Section 6.4.

Results for link prediction can be found in Figure 6.5. The bars in the plots show the precision for the different weights used, for each time period. We chose to report the precision because the class skew (many more non-links than links) makes the accuracy misleadingly high. Link prediction is notoriously difficult so the precision is quite low [92]. The important information in the plots though is the relative difference between the weights. With the exception of levant, using only the most recent data ($\vec{w}_{+1}$) results in better link prediction (except one timestamp in dblp3). Even in the levant set, for many of the timestamps, $\vec{w}_{+1}$ is either the best or very close to the best. These results support the previous findings that increasing the window size provides minimal benefit. In fact, the prediction results indicate that the first-order Markov assumption provides the best results for the majority of the sets.

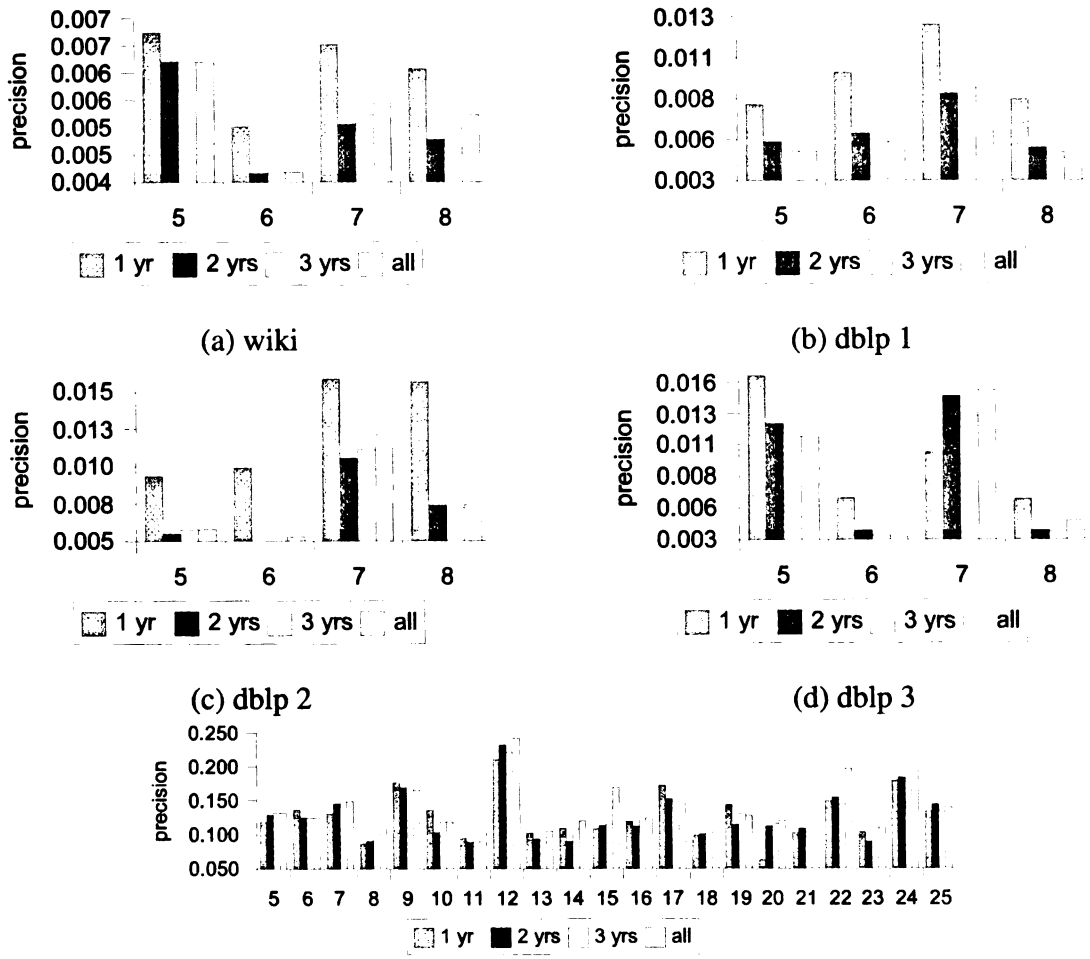
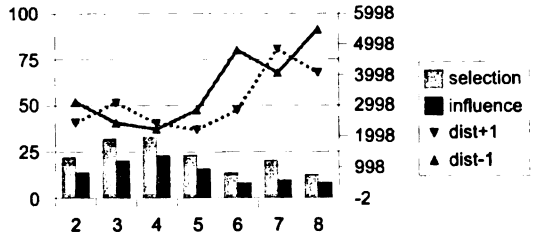


Figure 6.5: Precision for link prediction

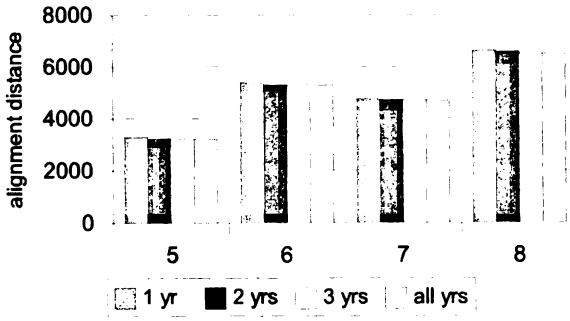
6.6.6 Snapshot Interval Size

As discussed earlier, one of the decisions when creating the network matrices that might impact the analysis is the interval size or the amount of time between snapshots. To study this further, an additional eight monthly snapshots – from January to August of 2006 – from the wikipedia data were created . The experiments were repeated on this new data set and the results are presented in Figure 6.6. In chart (a), selection and influence again appear to be present in the data. Like in the six month snapshots, selection appears to play a larger role than influence.

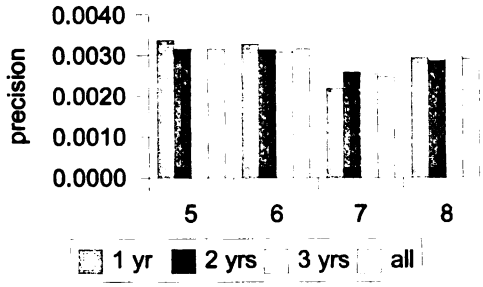
In chart (b), once again, the alignment distance is not only very close but actually



(a) selection and influence



(b) learning comparisons



(c) precision for link prediction

Figure 6.6: Experiments on wiki monthly data set

decreases with the amount of historical data used to learn the weights. Finally, in chart (c), with one exception, the prediction precision is slightly lower when using weights that were learned using more than one period of data.

6.6.7 Kernel Link Prediction

As pointed out in Section 6.4.4, the limitations of using a single vector of weights are ignoring the cross-product effect of the attributes and linear relationship assumption between the attributes and links. The kernel approach that was developed in Section 6.4.4 was tested on the wiki, dblp and levant data sets and the results are presented here.

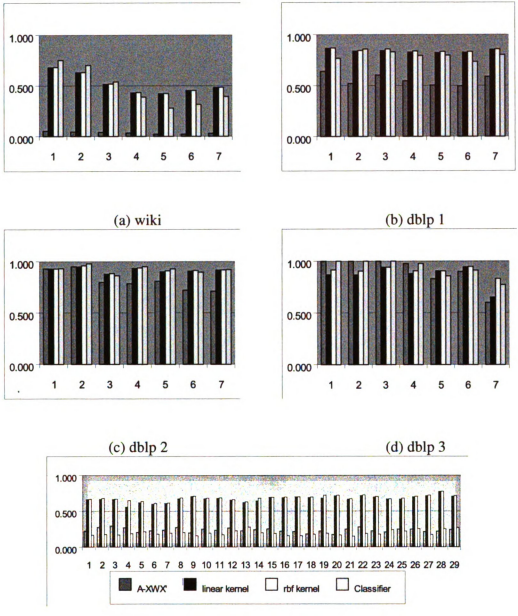
Experiments were run using both the non-accumulated data and the data where both the links and attributes are accumulated. In all experiments, the weights V were learned from $A^{(t)}$ and $X^{(t)}$ of a particular snapshot t and the links were predicted for the adjacency matrix $A^{(t+1)}$. The kernel formulation was tested using the linear kernel, XX^T, and the RBF kernel, $\exp\left(-\frac{|x_i - x_j|}{2\sigma^2}\right)$, where x_i and x_j are the attribute vectors for nodes i and j and σ is given. The exponential expression in the RBF kernel can be expanded using the Taylor series to an infinite polynomial series, so it can fit a more complex relationship of potentially infinite dimension.

The results compare the linear, weight vector, formulation ($A - XWX^T$), the two kernels described above and a classifier approach proposed by Al Hasan, et al. [47]. Al Hasan's approach used features created for node pairs to predict the link between nodes using traditional classifiers such as SVM and tree classifiers. For the data sets in this chapter only the cosine similarity is appropriate as a feature. Topological features cannot be used, since it would be necessary to know $A^{(t+1)}$ to predict $A^{(t+1)}$. Discriminant analysis is used as the classifier.

Figure 6.7 displays the results of the experiments using the non-accumulated data. The F-measures are fairly high for all dblp sets, a bit lower for levant and lower still for wiki. The values have a downward trend in wiki until period 5 where they start to recover. In general, the two kernel methods are either close to the classifier or better in wiki and dblp. In levant the kernel methods have much better F-measures. Another general trend is that the RBF kernel does at least as well as the linear kernel and sometimes better. The linear weighted solution ($A - XWX^T$) does well in dblp but poorly in the other two.

The results of the experiments on accumulated data are in Figure 6.8. With this data the kernel methods have better F-measures than either the classifier or $A - XWX^T$ in all cases. Again, the RBF kernel does at least as well as the linear kernel. For both the classifier and $A - XWX^T$ there is a trend where the results start high and then taper off.

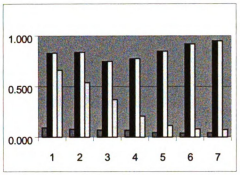
The figures appear to have a couple of curiosities that will be explained before conclusions are drawn from the results. First is the behavior of the classifier. It does better for non-accumulated and appears to decline sharply for the accumulated tests. There is some evidence that the performance of the classifier is correlated with the alignment of the net-



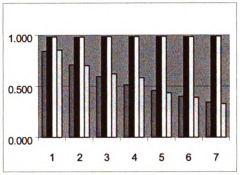
(e) levant

Figure 6.7: Link prediction comparisons using non-accumulated data

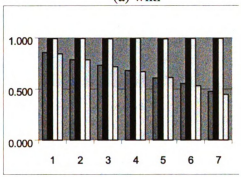
work. Figure 6.9 shows a comparison on the F-measure for the classifier (bars) against the inversion of the alignment distance (line). It appears, especially for wiki, that the classifier performs better with better aligned networks. This seems reasonable considering that the classifier uses only the cosine similarity to predict links. The better aligned a network is,



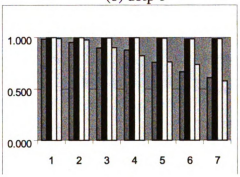
(a) wiki



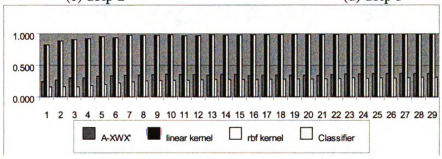
(b) dblp 1



(c) dblp 2



(d) dblp 3



(e) levant

Figure 6.8: Link prediction comparisons using accumulated data

the more predictive the attribute similarity between nodes will be. This also explains why the classifier performance tails off sharply for the accumulated tests. Recall that in Figure 6.2, the alignment gets increasingly worse over time for accumulated networks.

The second curiosity is the performance improvement for the kernel methods for the

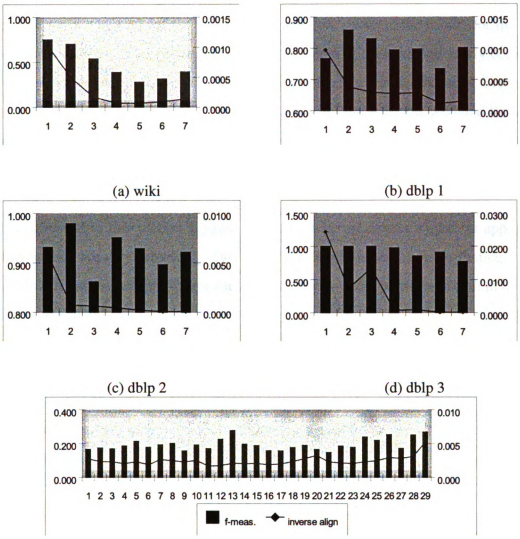


Figure 6.9: Comparison of classifier F-measure and alignment distance

accumulated networks. This appears to be the opposite of the situation with the classifier. It is important to remember the fundamental differences between the two methods. The classifier learns a threshold parameter according to the cosine similarity between pairs of nodes. On the other hand, the matrix alignment approach, learns weights associated with the attributes in order to make predictions. The classifier cannot weight the attributes because those distinctions are lost once the similarities are computed.

attribute weights differ greatly between two snapshots then the weights learned in the earlier one are less helpful for the later one. To see this, compare the results for the kernel methods in Figure 6.7 to the attribute drift in Figure 6.4. The average yearly correlation in Figure 6.4(a) is high for wiki and levant but much lower for the dblp sets. Correspondingly, the kernel approach has a higher F-measure in Figure 6.7 for wiki and levant than for dblp. In Figure 6.4(d), the correlation becomes greater with accumulated networks, so the drift becomes smaller and, as pointed out, the performance of the kernel matrix alignment approach gets better.

The results in this section suggest that using a kernel matrix alignment approach to temporal link are very promising. Two issues that need further study are finding a proper threshold for consistent prediction and avoiding the problem of attribute drift in networks that change rapidly.

6.7 Conclusions

Social network analysts have proposed the complementary processes of selection and influence as major forces at work in social networks. We were able to verify that these forces are at work in a variety of data sets. Our experiments also demonstrated that a larger window size of historical data is either not very helpful or even harmful. When network data is accumulated (using past link and attribute data), we show that the alignment distance between links and attributes widens. Results show that link prediction algorithms that rely on similarity perform better with networks that have smaller alignment distances.

Using the Markov assumption to limit the amount of historical data used in learning is a convenient simplifying assumption. We have shown that with a variety of data sets it is also the best choice for optimal performance. To confirm these results we calculated the precision of predicting links using different levels of history. In many cases, using historical data actually reduced the precision. In the cases of networks where the past historical data was helpful for prediction it is suggested that significant attribute drift is at least partly responsible.

To improve the results of link prediction, the matrix alignment framework was mod-

ified to use the entire weight matrix – rather than a vector, one weight for each attribute – and to incorporate kernel functions. Experimental results show that using the kernel approach is better than the linear weighted ($A - XWX^T$) approach and is frequently better than the classifier approach. The kernel version improves even further in networks without significant attribute drift.

Chapter 7

Conclusions and Future Work

The area of network mining is growing quickly for several reasons. One reason is that networks play an essential part of our everyday life. Another is that there are many challenges involved with linked, or related, data items. The same complexity that makes linked data challenging also opens up new opportunities and problems to solve.

This thesis makes use of the link structure within networks for improved prediction and node role identification. The algorithms proposed exploited the relationships involving links, specifically the relationships between links and communities and between links and attributes. Understanding the alignment between communities and the links offers valuable insights into the roles that nodes play with respect to communities. It was shown that learning the alignment between links and attributes led to improvements in link prediction and collective classification. Finally, the changes in the relationship of attributes to links over time was examined and as a result, information helpful for decisions that are made in processing network data were revealed.

7.1 Conclusions

Knowing the alignment between links and communities can guide analyst's decisions about using different network mining algorithms. It has been shown that this alignment can be accurately measured using the metrics introduced in Chapter 3. It was also shown that by using only the link structure, it is possible to assign a relative measure of the

number of communities to which a node belongs by using the efficient rawComm metric. Not only can this metric be used in assigning a role to a node but has been shown to be effective in an extension of the influence maximization problem.

The matrix alignment framework that was developed in Chapter 4 learns a set of weights that are optimized for aligning the link structure to the attributes of the nodes in a network. The weights identify how important individual attributes are for linking. In Chapter 4 it is shown that the learned weights can be used effectively for predicting linking in the missing links problem. It is also shown that the framework is flexible so that topological and community data can also be incorporated.

In Chapter 5, the matrix alignment framework is modified to the problem of collective classification. The formulation from Chapter 4 is modified so that the attributes and links are aligned to the co-class matrix. In a network of partially labeled nodes, the weights can be learned using the labeled nodes and their links. The weights can then be used to predict the class of the unlabeled nodes. The results show that by iteratively learning the weights and predicting the class, the framework is as effective as the best known algorithms for collective classification.

Chapter 6 studies temporal networks and the effects that preprocessing decisions have on the analyses performed on them. Extensive experiments measured the extent of influence and selection in a variety of data sets. The relationship between accumulated link and attribute data and the alignment of the network was explored. The validity of assuming the first order Markov assumption is defended by other experiments.

A kernel version of the matrix alignment framework was also developed in Chapter 6. It was shown to be very effective in predicting the next state of a temporal network. In these experiments it was also shown that a relationship exists between the alignment of a network and the effectiveness of some link prediction algorithms.

7.2 Future Work

While the utility of node roles was demonstrated in Chapter 3 to the problem of influence maximization, it would be interesting to apply them to the problem of collective

classification. Another way they could be utilized is as a guide in a community finding algorithm.

There are a number of possible extensions to the matrix alignment framework:

1. In temporal networks, links and attributes change over time. This means that the labels (which are normally just a selected attribute) can also changed. The collective classification approach from Chapter 5 applies only to the missing label problem. A valuable extension would be to modify it to predict labels in a new time period. With collective classification labels are dependent on neighbor's labels, thus allowing all nodes to change labels could lead to an unstable solution. The challenge here would be allow labels to be changed incrementally so that the solution is able to converge.

2. While community information was incorporated into the framework in Chapter 4 it was only used for link prediction. Another interesting extension would be finding hidden communities while simultaneously classifying unlabeled nodes.

3. As networks change the weights need to be constantly recalculated – an expensive operation. With some networks where the attribute drift is not severe, a more efficient incremental approach to updating the weights would be valuable. It would involve analyzing the effect a single change has to the weights after Gaussian elimination and possibly maintaining a set of intermediate values.

4. There has been some work on combining link prediction and collective classification [14] using an iterative approach. This approach uses available link prediction and collective classification algorithms to alternately predict the links or the class until convergence. It would be interesting to extend to matrix alignment framework to this problem using an integrated approach. The challenge would be to somehow use a single set of weights so that it does not reduce to the iterative approach mentioned above.

5. There are a number of challenges with the matrix alignment framework that could also addressed in future work, such as dealing with links of different types and directional links. In this thesis, the links are assumed to be non-directional since

the attribute similarity, XWX^T, is symmetric and so it necessary for A to be also. Something other than similarity would be needed here. The challenge with different link types would be to integrate the multiple A matrices.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. Adafre and M. De Rijke. Discovering missing links in wikipedia. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Workshop on Link Discovery*, 2005.

[2] E. M. Airoldi and K. M. Carley. Sampling algorithms for pure network topologies. *SIGKDD Explorations*, Dec 2005.

[3] A. Anagnostopousos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[4] C. Anderson, S. Wasserman, and B. Crouch. A p* primer: logit models for social networks. *Social Networks*, 21:37–66, 1999.

[5] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[6] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Proc. of the 43d Annual IEEE Symp. on Foundations of Computer Science.*, 2002.

[7] A.-L. Barabsi and E. Bonabeau. Scale-free networks. *Scientific American*, 288: 50–59, May 2003.

[8] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

[9] S. Basu, M. Bilenko, and R. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[10] A. Ben-Hur and W. Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21, 2005.

[11] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society*, B36:192–236, 1974.

[12] S. Bharathi, D. Kempe, and M. Salek. Competitive influence maximization in social networks. In *Proceedings of WINE*, 2007.

[13] M. Bilenko, S. Basu, and R. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine learning*, 2004.

[14] M. Bilgic, G. Namata, and L. Getoor. Combining collective classification and link prediction. In *IEEE ICDM Workshop on Mining Graphs and Complex Structures*, 2007.

[15] D. Blei, S. Fienberg, A. McCallum, C. Shalizi, and M. Handcock. Panel discussion. In *Proceedings of the international conference on machine learning, statistical networ analysis workshop*, 2006.

[16] S. P. Borgatti and M. G. Everett. Models of core / periphery structures. *Social Networks*, 21:375–395, 1999.

[17] R. Bunker. *Networks, Terrorism and Global Insurgency*. Routledge, 2005.

[18] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Mining hidden community in heterogeneous social networks. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Workshop on Link Discovery*, pages 58–65, 2005.

[19] S. Chakrabarti, B. Dom, and P.Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the SIGMOD International Conference on Management of Data*, 1998.

[20] H. Chang and D.-Y. Yeung. Robust path-based spectral clustering. *Pattern Recogn.*, 2008.

[21] A. Clauset, C. Moore, and M. E. J.Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2003.

[22] A. Clauset, C. Moore, and M. E. J.Newman. Structural inference of hierarchies in networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML), Workshop on Social Network Analysis*, 2006.

[23] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[24] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[25] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, pages 367–373. MIT Press, 2002.

[26] I. Davidson and S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of SDM'06: SIAM Int'l Conference on Data Mining*, 2005.

[27] I. Davidson and S. Ravi. Towards efficient and improved hierarchical clustering with instance and cluster level constraints. Technical report, Department of Computer Science, University at Albany, 2005.

[28] H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology*, 9:67–103, 2002.

[29] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[30] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the Seveth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 57–66, 2001.

[31] P. Doreian, V. Batagelj, and A. Ferligoj. Positional analysis of sociometric data. In P. Carrington, J. Scott, and S. Wasserman, editors, *Models and Methods in Social Network Analysis*. Cambridge, New York, 2005.

[32] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2000.

[33] P. Erdös and A. Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[34] O. Frank and D. Strauss. Markov graphs. *Journal of the American Statistical Association*, 81:832–842, 1986.

[35] T. Frantz and K. M. Carley. A formal characterization of cellular networks. Technical Report CMU-ISRI-05-109, Carnegie Mellon University, 2005.

[36] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Conf. on Artificial Intelligence (IJCAI)*, 1999.

[37] J. Gao, P.N. Tan, and H. Cheng. Semi-supervised clustering with partial background information. In *Proceedings of SDM'06: SIAM Int'l Conference on Data Mining*, 2006.

[38] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Explorations*, 7:3–12, 2005.

[39] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707, 2002.

[40] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, 1998.

[41] M. Girvan and M. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci*, 99:7821–7826, 2002.

[42] J. Goldenberg, B. Libai, and E. Muller. Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata. *Academy of Marketing*, 01, 2001.

[43] K. Gordon. The power of social shopping networks. *Entrepreneur Magazine*, March 2007.

[44] M. Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83, 1978.

[45] C. Guestrin, D. Koller, C. Gearhart, and Neal Kanodia. Generalizing plans to new environments in relational mdps. In *International Joint Conference on Artificial Intelligence*, 2003.

[46] S. Hanneke and E. Xing. Discrete temporal models of social networks. In *Proceedings of the 23rd International Conference on Machine Learning Workshop on Statistical Network Analysis*, 2006.

[47] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *Proceedings of SDM'06: SIAM Data Mining Conference Workshop on Link Analysis, Counter-terrorism and Security*, 2006.

[48] P. Holland and S. Leinhardt. An exponential family of probability distributions for directed graphs. *Journal of the American Statistical Association*, 76:33–50, 1981.

[49] A. Jain and R. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[50] J.R.Tyler, D.M. Wilkinson, and B.A.Huberman. Email as spectroscopy: Automated discovery of community structure within organizations. In *Proceedings of the First International Conference on Communities and Technologies*, 2003.

[51] D. Kandel. Homophily, selection, and socialization in adolescent friendships. *The American Journal of Sociology*, 84:427–436, 1978.

[52] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. In *ACM/IEEE conference on Supercomputing*, 1995.

[53] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359–392, 1999.

[54] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 8:68–75, 1999.

[55] H. Kashima and N. Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *Proceedings of the Sixth IEEE International Conference on Data Mining*, 2006.

[56] H. Kashima, S. Oyama, Y. Yamanishi, and K. Tsuda. On pairwise kernels: An efficient alternative and generalization analysis. In *Proc. 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2009.

[57] KEDS. Kansas event data system. http://web.ku.edu/keds.

[58] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 1999.

[59] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

[60] J. Kleinberg. Sources in a hyperlinked environment. *Journal of the ACM*, 46, 1999.

[61] T. Ko and N. Berry. Agent-based modeling with social networks for terrorist recruitment. In *Proceedings of American Association for Artificial Intelligence*, 2004.

[62] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, 2001.

[63] M. Lahiri and T. Y. Berger-Wolf. Structure prediction in temporal networks using frequent subgraphs. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 2007.

[64] R. Lawrence. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE, 77(2)*, 1989.

[65] E. Lazega and M. van Duijn. Position in formal structure, personal characteristics and choices of advisors in a law firm: A logistic regression model for dyadic network data. *Social Networks*, 19:375–397, 1997.

[66] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005.

[67] M. Ley. Computer science bibliography, 1998. http://dblp.uni-trier.de/.

[68] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, 2003.

[69] linqs. Statistical relational learning group. http://www.cs.umd.edu/linqs/.

[70] Q. Lu and L. Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine learning*, 2003.

[71] G. Luger. *Cognitive science : the science of intelligent systems*. Academic Press, 1994.

[72] C. Macal and M. North. Tutorial on agent-based modeling and simulation part 2: how to model with agents. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, 2006.

[73] S. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, 2007.

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

[75] C Moore, G Ghoshal, and M. E. J. Newman. Exact solutions for models of evolving networks with addition and deletion of nodes. *Phys. Review E*, 74, 2006.

[76] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[77] J. Neville and D. Jensen. Dependency networks for relational data. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, 2004.

[78] J. Neville and D. Jensen. Leveraging relational autocorrelation with latent group models. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, 2005.

[79] J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. In *Proceedings of the Text Mining and Link Analysis Workshop, Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[80] J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. In *Proceedings of the Text Mining and Link Analysis Workshop, Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[81] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, Feb 2004.

[82] M. E. J. Newman. Mathematics of networks. In L. E. Blume and S. N. Durlauf, editors, *The New Palgrave Encyclopedia of Economics*. Palgrave Macmillan, 2008.

[83] A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2002.

[84] R. T. Ng and J. Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14:1003–1016, 2002.

[85] J. O'Madadhain, J. Hutchins, and P. Smyth. Prediction and ranking algorithms for event-based network data. *SIGKDD Explorations*, 7:23–30, Dec 2005.

[86] S. Oyama and C. Manning. Using feature conjunctions across examples for learning pairwise classifiers. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, 2004.

[87] L. Page, S. Brin, R. Motwani, and T. Winograd. Pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[88] M. Pearson, C. Steglich, and T. Snijders. Homophily and assimilation among sport-active adolescent substance users. *Connections*, 27:47–63, 2006.

[89] A. Popescul and L. H. Ungar. Statistical relational learning for link prediction. In *Proceedings of the IJCAI Workshop on Learning Statistical Models from Relational Data*, 2003.

[90] A. Potgieter, K. April, R. Cooke, and I. O. Osunmakinde. Temporality in link prediction: Understanding social complexity. *Journal of Transactions on Engineering Management*, 7, 2006.

[91] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. In *Proceedings of the National Academy of Sciences*, 2004.

[92] M. Rattigan and D. Jensen. The case for anomalous link discovery. *SIGKDD Explorations*, 7, 2005.

[93] P. Reddy and M. Kitsuregawa. Inferring web communities through relaxed cocitation and dense bipartite graphs. In *Proceedings of Data Base Engineering Workshop*, 2001.

[94] G. Robins, P. Eliot, and P. Pattison. Network models for social selection processes. *Social Networks*, 23:1–30, 2001.

[95] G. Robins, P. Pattison, and P. Eliot. Network models for social influence processes. *Psychometrika*, 66:161–189, 2001.

[96] G. Robins, P. Pattison, Y. Kalish, and D. Lusher. An introduction to exponential random graph (p*) models for social networks. *Social Networks*, 29:169–172, 2007.

[97] B. Schlkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[98] J. Scott. *Social Network Analysis: A Handbook*. Sage Publishing, 2000.

[99] J. Scripps and P. N. Tan. Clustering in the presence of bridge-nodes. In *Proceedings of SDM'06: SIAM Int'l Conference on Data Mining*, 2006.

[100] J. Scripps and P. N. Tan. Constrained overlapping clusters: Minimizing the negative effects of bridge-nodes. *IEEE Transactions on Knowledge and Data Engineering*, submitted.

[101] J. Scripps, P. N. Tan, and A-H Esfahanian. Exploration of link structure and community-based node roles in network. Technical report, Michigan State University, 2007.

[102] J. Scripps, P. N. Tan, and A-H Esfahanian. Node roles and community structure in networks. In *Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Joint Workshop on Web Mining and Social Network Analysis*, 2007.

[103] J. Scripps, P. N. Tan, and A-H Esfahanian. Exploration of link structure and community-based node roles in network analysis. In *Proceedings of the Seventh IEEE International Conference on Data Mining*, 2007.

[104] J. Scripps, P. N. Tan, and A-H Esfahanian. A matrix alignment approach for link prediction. In *Proceedings of the Nineteenth international conference on pattern recognition*, 2008.

[105] J. Scripps, P. N. Tan, and A-H Esfahanian. A matrix alignment approach for collective classification. In *International Conference on Advances in Social Networks Analysis and Mining*, 2009.

[106] J. Scripps, P. N. Tan, F. Chen, and A-H Esfahanian. A matrix alignment approach for link prediction. *Data Mining and Knowledge Discovery*, submitted.

[107] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. Technical Report CS-TR-4905 and UMIACS-TR-2008-04, University of Maryland, 2008.

[108] T. Senator. Darpa, evidence extraction and link discovery program, 2002.

[109] U. Sharan and J. Neville. Temporal-relational classifiers for prediction in evolving domains. In *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008.

[110] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 22(8), August 2000.

[111] T. Snijders. Models for longitudinal network data. In P. Carrinton, J. Scott, and S. Wasserman, editors, *Models and methods in social network analysis*. Cambridge University Press, 2004.

[112] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, Inc., 2005.

[113] C. Tantipathananandh, T. Y. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[114] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)*, 2002.

[115] B. Taskar, M. F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Neural Information Processing Systems Conference (NIPS03)*, 2003.

[116] teenage. Siena network statistical analysis program. http://stat.gamma.rug.nl/snijders/siena.html.

[117] Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[118] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge, UK, 1994.

[119] S. Wasserman and P. Pattison. Logit models and logistic regression for social networks: I an introduction to markov graphs and p*. *Psychometrika*, 61:401–425, 1996.

[120] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, pages 440–442, Jun 1998.

[121] webkb. World wide knowledge base. http://www.cs.cmu.edu/ WebKB/.

[122] Y. Weiss. Comparing the mean field method and belief propagation for approximate inference in mrfs. In M. Opper and D. Saad, editors, *Advanced Mean Field Methods*. MIT Press, 2005.

[123] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 18, March 2002.

[124] S. Zhu, K. Yu, Y. Chi, and Y. Gong. Combining content and link for classification using matrix factorization. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[125] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.