2014

Learning Collective Behavior in Multi-relational Networks

Xi Wang
University of Central Florida

LEARNING COLLECTIVE BEHAVIOR IN MULTI-RELATIONAL NETWORKS

by

XI WANG
M.S. University of Central Florida, 2010

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2014

Major Professor: Gita Sukthankar

ABSTRACT

With the rapid expansion of the Internet and WWW, the problem of analyzing social media data has received an increasing amount of attention in the past decade. The boom in social media platforms offers many possibilities to study human collective behavior and interactions on an unprecedented scale. In the past, much work has been done on the problem of learning from networked data with homogeneous topologies, where instances are explicitly or implicitly interconnected by a single type of relationship. In contrast to traditional content-only classification methods, relational learning succeeds in improving classification performance by leveraging the correlation of the labels between linked instances. However, networked data extracted from social media, web pages, and bibliographic databases can contain entities of multiple classes and linked by various causal reasons, hence treating all links in a homogeneous way can limit the performance of relational classifiers. Learning the collective behavior and interactions in heterogeneous networks becomes much more complex.

The contribution of this dissertation include 1) two classification frameworks for identifying human collective behavior in multi-relational social networks; 2) unsupervised and supervised learning models for relationship prediction in multi-relational collaborative networks. Our methods improve the performance of homogeneous predictive models by differentiating heterogeneous relations and capturing the prominent interaction patterns underlying the network structure. The work has been evaluated in various real-world social networks. We believe that this study will be useful for analyzing human collective behavior and interactions specifically in the scenario when the heterogeneous relationships in the network arise from various causal reasons.

To my father and his eternal love

ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor, Dr. Gita Sukthankar. I knew I can not make this happen without her tremendous guidance and support during the past 5 years. Accomplishing a Ph.D. degree is never easy, and Dr. Gita's constant effort and encouragement in my academic research made this journey less difficult than I thought. The knowledge and skills that I have learned from her will continue to shape me in many aspects of my future life.

Secondly, I would like to thank my committee members, Dr. Michael Georgiopoulos, Dr. Georgios Anagnostopoulos, Dr. Haiyan Hu and Dr. Marshall Tappen, for their precious time and valuable comments on my dissertation.

Additionally, I want to give thanks to my colleagues and friends at UCF. Because of them, my Ph.D. life was even more precious and memorable. I would like to thank Liyue Zhao and Fahad Shah for guiding me at the beginning of my Ph.D. study. I also want to thank Yuyan Bao, Qian Zhang and members at Intelligent Agent Lab (Mahsa Maghami, Bennie Lewis and many others) for traveling this journey along with me. Without their consistent help and encouragement, it would be much more difficult for me accomplishing this dissertation.

Finally, I owe the deepest gratitude to my family: my sister and best friend, Chen Wang, for offering me unconditional support during my bad and good times, my mother, for her selfless devotion and sacrifices, and my beloved father. Completing a Ph.D. degree is not only my dream but also my father's hope, and this dissertation is dedicated to his endless love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

With the rise in popularity of social media, platforms such as Facebook, Twitter, YouTube, and Blog have attracted growing attention among consumers, marketers and even business companies. Social media has completely revolutionized people's lives in the way that we can now express opinions, stay in touch with distant friends, and share videos. It has been reported that there are 955 million active users on Facebook that spend an average of six hours and 35 minutes per month on the network this year (desktop only); 61% of job seekers choose LinkedIn as their primary professional networking site; and 27% of small and 34% of medium businesses are using social media for business, which represents an annual increase of approximately 10% [93]. Social media provides a platform that empowers new and easy forms of communication, collaboration and spreading intelligence. The online personal reply, such as comments on photos or messages on Facebook "wall", provides a convenient yet private channel for people to interact with each other, without the pressure of response that comes with e-mail or phone; a huge number of users collaboratively create or edit encyclopedia articles in many languages on the Wikipedia website; consumer behavior has been greatly influenced by the previous buyers' reviews on online shopping website such as Amazon and Ebay. The boom of social media opens up a vast range of possibilities to study human collective behavior and interactions on an unprecedented scale.

Social media is able to provide a massive amount of useful data for investigating the problem of human collective behavior. First of all, with an exponentially increasing number of active users, the amount of user interactions is growing rapidly. For instance, Facebook has 232 million users in Europe, 222 million in North America, and 219 million users in Asia as of September 2012, and an average of 3.2 billion likes and comments are posted every day [93]. The huge quantity of user participation and interaction enable the construction of a connected network of users, thus providing the opportunity to study human collective behavior in a networked data structure. Secondly, "tagging" in social media offers meaningful semantic annotations to represent users' in-

terest and personality, and is valuable for learning the behavior correlations between a user and his friends. The usage patterns of tags serve as one simple way to track people's interests and investigate the spread of shared concepts through a user community. Online interest groups in YouTube and Blog are other mechanisms for representing user interest. Thirdly, since the objects in social networks are usually highly dynamic, the addition of new users and new interactions happens quickly. Social media is able to provide the time information to support the study of the evolution of new connections in the underlying social structure.

To better analyze human collective behavior and interactions, people embedded into a social context are represented in a network structure, where nodes define people and edges represent interaction, collaboration, or influence between entities. Besides social networks, there are many other examples of networked data that exist in real life. For instance, research papers are connected by citations, web pages are linked by hyperlinks and proteins interact with each other in biological networks. In the past decade, much attention has been paid to the problem of learning from networked data [60, 107, 101, 69]. Given a network structure, the areas that attract research interest have mainly fallen into the following domains: (1) models of how nodes in the network influence each other (for example, who infects whom in an epidemiological network), (2) models for predicting an attribute of interest based on observed attributes of objects in the network (for example, predicting a user's interest group based on his online interactions in social media), (3) identifying important nodes in the network (for example, critical nodes in communication networks), (4) predicting the relationship between node pairs based on the relationship between their connectors (for example, predicting whether two people are friends based on their friends' interactions). In most cases, a critical step in achieving the above goals is classifying, or labeling, the nodes in the network, and this can be referred to as the *node classification problem* [11].

While the structure of networked data can provide an enormous volume of information (personal information, opinion and interactions), it also complicates many machine learning problems. The data are no longer independent and identically distributed (i.i.d.), which makes the learning and

inference procedure more difficult. The area of Statistical Relational Learning (SRL) [38] focuses on learning models of networked data–objects or entities that are represented by an uncertain, complex and relational structure. Essentially, SRL learns models of networked structure by utilizing the correlation between labels of linked objects. The tendency of individuals to associate and bond with similar people, described as "homophily", has been verified in vast array of network studies [72]. Over the past decade, the algorithms developed for relational classification can be categorized into the following categories: 1) heuristic methods 2) models that represent formal probabilistic semantics, such as graphical models or relational learning. The first group consists of the heuristic methods that do not necessarily correspond to formal probabilistic semantics [77, 67, 68]. Other approaches rely on constructing graphical models that capture the probability distribution of the data during the inference process, assuming that the structure of the network corresponds at least partially to the structure of the network of probabilistic dependencies [69, 79]. Relational learning enhances the tractability of estimating the full joint probability distribution of the data by making a first-order Markov assumption that the label of one node is only dependent on that of its immediate neighbors in the graph.

In conventional classification problems, the dataset is usually divided into training and testing sets for learning and predicting. Separation in networked data is complicated and also removes the useful background information from the training data that facilitates classification inferencing. *Within-network classification* refers to a special case in relational learning, when both training and testing instances are connected in one network, which allows the use of specific node identifiers to aid inference [69].

Collective inference has been widely adopted in *within-network classification*. It makes statistical estimations of various interrelated values simultaneously, and finds a equilibrium state such that the inconsistency between neighboring nodes is minimized. Similar inference models have already been applied within Markov random fields, in which the estimate of a node's neighbors' labels is used to influence the estimation of the node's label-and vice versa [10, 36]. By

exploiting network connectivity information, relational classification models have been shown to outperform traditional classifiers [80, 112].

1.1 Networked Data with Heterogeneous Links

Collective inference in relational learning relies on the assumption that the connections in the network are homogeneous, meaning each pair of entities in the network are only associated with a single type of relationship. In the real world, social, natural, and information systems usually consist of a large number of multi-typed components connected by various types of interactions. Essentially, these interconnected, multi-typed networks or systems can be described as heterogeneous information networks [45]. Clearly, heterogeneous information networks form a critical component of modern information infrastructure, and analysis on these networks can be much more complex. In this dissertation, we focus on networks with a single type of entity but heterogenous links. We refer to this type of network as a "multi-relational network". For example, in friendship networks extracted from social media, interactions between people can be driven by multiple personal interests. Examples of interests include joining the same chat group, reading about the same topic, or watching the same online video. For this reason, social media links from a single person are generally not of homogeneous origin. Person **A** connects with person **B** because they go to the same tennis club whereas his interaction with person **C** is mainly due to attending the same math class. It is also possible for links between people to be based on multiple shared interests; person **A** may have first encountered person **D** in math class before mutually deciding to join a chess club together. In the familiar example of collaboration networks, scientific authors usually have multiple research interests and seek to collaborate with different co-authors for different types of work. For instance, Author A cooperates with author B on publishing papers in machine learning conferences whereas his/her interaction with author C is mainly due to work in the data mining area. If these connections are not grouped or tagged, it is possible to implicitly

treat all of these connections as originating from the same type of social interaction. However it can result in erroneous classification results [109].

This dissertation centers around the following two topics:

• Collective behavior classification in multi-relational networks.

• Relationship prediction in multi-relational networks.

These two topics are directly related to the problems of *node classification* and *link prediction* in networked data.

1.1.1 Node Classification in Multi-relational Networks

In social networks, people usually connect with each other for different causal reasons. The link heterogeneity in social connections leads to the situation that instances can be associated with more than one label that represents people's interests. From this point of view, classifying an object's type in such a network can be regarded as a multi-label classification task. Collective classification becomes particularly challenging in multi-label settings since the label dependencies among related instances are more complex. Currently, most collective inference models do not differentiate in their treatment of connections between instances. Although some tools such as Facebook contain mechanisms for grouping and tagging links, many social media datasets do not possess this additional information. Distinguishing multiple types of links becomes quite important in this scenario.

1.1.2 Link Prediction in Multi-relational Networks

With massive amounts of dynamic objects, users and their interconnections change rapidly over time through adding new connections to the social structure. The link prediction problem aims to understand the underlying mechanism of interaction evolution within the network structure. Within scientific collaboration networks, it is a common phenomenon that a new collaboration will be formed between two researchers if they have a large number of former collaborators in common.

In previous work, the similarity between two people is a widely used criteria to determine the existence of new connections. However, the evaluation of the "closeness" between two people is no longer solely based on geography. Instead, most of the link prediction models mainly rely on the network topology at a previous time period to predict new connections in the future.

Link prediction in multi-relational datasets can be much more complex since each instance is now associated with multiple affiliations. Treating these instances and relationships identically loses useful discriminative information that can improve prediction performance. For instance, in a network with heterogenous relationships such as friend, family, colleague and classmate, it may be far more reasonable for a person to form a new interaction with the colleague of a colleague than with the parent of a colleague. Methods that integrate diverse relationship information would be beneficial for link prediction in heterogeneous information networks.

1.2 Research Contribution

The main goal in this dissertation is to study human collective behavior in networks with multi-relational relationships. In this scenario, each person can be associated with a subset of multiple labels from the candidate label set. In the first of part of this dissertation, we present two novel frameworks to learn collective behavior in multi-relational networks. One is to construct a low-dimensional social feature space by applying Fiedler embedding to an edge-based social feature space. The other one is a multi-label iterative relational neighbor classifier that incorporates a class propagation probability distribution obtained from instances' social features. We demonstrate that both proposed approaches are capable of identifying semantically similar users within the social network and outperform several benchmark methods in various real world datasets.

In the second part of this dissertation, we introduce a new supervised link prediction framework, Link Prediction using Social Features (LPSF), which incorporates a reweighting scheme for the network based on nodes' features extracted from patterns of prominent interactions across

the network. Experiments on coauthorship networks demonstrate that our proposed reweighting method in LPSF better expresses the intrinsic relationship between nodes and improves prediction accuracy for supervised link prediction techniques. We also compare the unsupervised performance of the individual features used within LPSF with two new diffusion-based methods: LPDP (Link Prediction using Diffusion Process) and LPDM (Link Prediction using Diffusion Maps). Experiments demonstrate that LPDP is able to identify similar node pairs, even far away ones, that are connected by weak ties in the coauthorship network using the diffusion process.

1.2.1 Extracting Social Dimensions using Fiedler Embedding

When we examine networks extracted from social media data, one important finding is that the links (connections) possess a strong correlation with the affiliation class. Connections in the social network are mainly affiliation-driven, and each connection in the social network can be regarded as principally resulting from one affiliation. Since each person usually has more than one connection, the involvement of potential groups related to one person's edges can be utilized as a representation for his true affiliations. Previously, Tang and Liu proposed a framework based on *Social Dimensions* to address link heterogeneity [110]. This framework extracts social dimensions based on network connectivity to capture the potential affiliations of actors. In their initial study, they proposed the use of modularity maximization to calculate social dimensions [108]. But it suffers the computation complexity problem when the dataset is large (i.e., more than 1 million). Edge-Clustering based social dimensions use an unsupervised clustering algorithm, K-means clustering, to partition the edges into disjoint sets such that each set represents one potential affiliation [109]. By grouping the edges, actors who have more links are more likely to associate with different types of links since they are usually involved in multiple affiliations. In this dissertation, we proposed an alternative framework to construct social dimensions based on Fiedler embedding that captures the correlations between different potential affiliations extracted after Edge-Clustering [109]. We start by constructing an initial social feature space, an edge-based

representation of social dimensions using the network topology to capture the person's potential affiliations as described in [109]. Then these initial social feature dimensions are embedded into a lower-dimensional space that reveals the relationships between different entities in the network. The Fiedler embedding uncovers hidden similarities between individual users as well as between connections. A discriminant classifier is trained using the embedded social features and finally, the class labels of the unlabeled nodes are predicted using the trained classifier. Experiments on two real-world social media datasets demonstrate that the proposed framework offers a better feature representation for multi-label classification problems on social media.

1.2.2 Multi-label Iterative Relational Neighbor Classifier using Social Context Features

Our second contribution is a multi-label relational classifier that accounts for this inhomogeneity in connections and is designed for classification problems on multi-label networked datasets. Our proposed method, multi-label relational neighbor classifier using social context features (SCRN), extends the relational neighbor classifier (RN) [68] by introducing a node class-propagation probability that modulates the amount of propagation that occurs in a class specific way based on the node's similarity to each class. Although the class propagation probability can be determined by the node's intrinsic features, it can also be based on node's social features. These features capture link patterns between a node and its neighbors and can be extracted from network topology in instances when the node lacks intrinsic features. SCRN's ability to differentiate between classes during the inference procedure allows it to outperform previous methods in a real-world multi-label relational dataset and a set of synthetically generated sets designed with challenging network parameters.

1.2.3 Predicting Interactions in Overlapping Communities based on Reweighted Networks

Community detection utilizes the notion of "structural equivalence" which refers to the property that two actors are similar to one another if they participate in equivalent relationships [85].

Inspired by the connection between structural equivalence and community detection, Soundarajan and Hopcroft proposed a link prediction model for non-overlapping communities; they showed that including community information can improve the accuracy of similarity-based link prediction methods [102]. Since community information is not always readily available, community detection techniques can be applied to partition the network into separate groups [2]. In this dissertation, we present a new link prediction framework for networks with overlapping communities that accounts for the hidden community information embedded in a set of heterogeneous connections. Our supervised link prediction framework, Link Prediction using Social Features (LPSF), weights the network using a similarity function based on features extracted from patterns of prominent interactions across the network. Given an unweighted network structure, the nodes' social features are first extracted from the network topology using an edge-based clustering method and used to reweight the network. These generated link weights represent the holistic strength of relationships and also provide useful information for evaluating the similarity of node pairs that are not directly connected. Experiments on coauthorship networks demonstrate that the choice for measuring link weights can be critical for the link prediction task. Our proposed reweighting method LPSF better expresses the intrinsic relationship between nodes and improves prediction accuracy for supervised link prediction techniques.

CHAPTER 2: LITERATURE REVIEW

The study of networked data has drawn an increasing amount of interest from researchers over the past decade. Initially, most work only focused on the study of homogenous networked data [80, 38, 112]. With the prevalence of social media and information systems, the study of heterogeneous networks has attracted increased research attention [45, 110, 111]. The topics covered in this dissertation can be regarded as an extension to the corresponding problems in homogeneous networks. Therefore, we will start describing the related work in homogenous networks, and then move on to the topics in networks with link heterogeneity. The related work we are going to discuss is organized as follows:

- Collective classification in homogeneous networks
- Collective classification and community detection in multi-relational networks
- Link prediction in homogeneous networks
- Link prediction in multi-relational networks
- Combining collective classification and link prediction
- Other research topics related to this dissertation
- Public datasets

2.1 Classification on Homogeneous Networked Data

Classifying, or labeling the nodes in the network, is an important step in achieving many problems related to networked data [101]. Collective classification (CC) is one of the important tasks that has been closely investigated in *statistical relational learning* (SRL) research [21, 51, 77, 112]. Traditional learning methods, such as independent classification, make predictions based

on the assumption that instances are independent of each other, which loses the critical correlation information that existed in networked data. In general, collective classification is a methodology that simultaneously classifies related instances in the network. It takes advantage of the variable correlations between inter-connected nodes. Homophily [72] is one of the simplest types of correlation, which describes the phenomenon that nodes with similar labels are more likely to be linked. Essentially, collective classification utilizes three different types of correlations to determine the labels of a node v given its network [101]: (1) The correlation between the label of v and the observed attributes of v. (2) The correlation between the label of v and the observed attributes (including the observed labels) of nodes in its neighborhood. (3) The correlation between the labels of v and the unobserved labels of nodes in the neighborhood of v. Collective classification has shown significant improvement in classification accuracies over non-collective methods when instances are interrelated [78, 67, 68].

2.1.1 Problem Definition

Generally, the problem of supervised classification on networked data can be formulated in the following two distinct ways:

• Training with a fully labeled dataset. In this case, we are given two disjoint graphs containing relational data interconnected by links: $G_{tr} = (V_{tr}, E_{tr})$ and $G_{test} = (V_{test}, E_{test})$. G_{tr} is the training graph consisting of instances with correct labeling assignments whereas the labels in G_{test} are unknown and need to be determined. The task is learning a probability distribution which assigns the maximum probability to the correct labeling assignment from the fully labeled training data G_{tr} and then applies this learned probability distribution to determine the most likely labeling assignment for G_{test}. Note that since G_{tr} and G_{test} are completely separate graphs, they do not share any variable (V, E) in common.

• Classification models are learnt using a partially labeled dataset. A set of relational data is represented by a graph $G = (V, E)$, where $V = \{V_1, V_2, ...V_n\}$ is a set of nodes, and E

describes the underlying connections between nodes. V is further divided into two sets of nodes: V_{tr}, the nodes for which we know the correct values (observable variables), and V_{test}, the nodes whose values need to be determined. The task is to determine the labels of the unknown nodes, V_{test}, from the label set L, based on their interactions with other nodes (from V_{tr} and V_{test}) in the network. Because V_{tr} and V_{test} share links with each other, the process of learning and predicting is conducted on the whole graph. Deliberately separating the network into two disconnected graphs might lose valuable information for classification [100].

Collective classification satisfies the second framework where the training and testing procedure are executed simultaneously. The goal of CC is to determine the set of labels L^U of the unlabeled nodes V^U that minimizes the zero-one loss or error rate. Any algorithm which does this using the links E in G is called a collective classification algorithm (CCA). Much real-world data involves large graphs which make exact inference intractable. Consequently, an approximate collective classification algorithm (ACCA) has to be used to find an approximately optimal solution [101]. In this dissertation we will only focus on the classification cases that are conducted on partially labeled datasets, since most of the work completed in the following sections belongs to this topic.

To better explain how collective classification performs, we will start with a toy example from a citation network, the CORA dataset, which is shown in Figure 2.1. In this example, each paper is indicated by a box, the associated topic of the paper is indicated by a rectangular box and the words in the paper are represented using a circle inside the box. The words $T = \{T1, T2, ..., T6\}$ contained in the paper are used as node attributes. The observed random variables are shaded, whereas the unobserved ones are not. The value for the unobserved label variable, L, can be selected from "GA" (abbreviated to "Generic Algorithm") and "NN" (abbreviated to "Neural Network").

In Figure 2.1, there are two unobserved variables (Y_2 and Y_3) and eight observed variables (X_1,X_2,...,X_8), where X_1, X_4 and X_5 are labels of papers and others are attributes of papers. To

predict the labels of paper W_1, $\mathcal{Y}_2$, collective classification methods consider the word attributes (X_8) associated with W_1 and its citation links to paper W_1, W_3 and W_5.

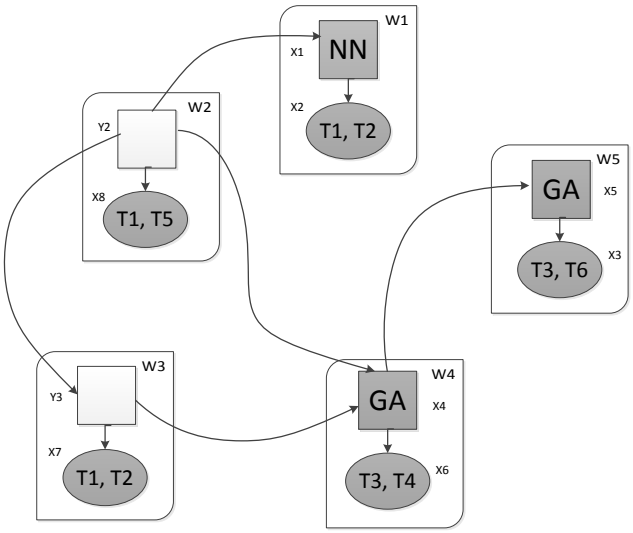


Figure 2.1: A small classification problem in the CORA dataset: Each box denotes a paper, each directed edge between a pair of boxes denotes a citation link, each rectangular box denotes the label of the paper, and the words associated with the paper are represented in circles. The shaded shape denotes an observed variable, whereas an unshaded oval node denotes an unobserved variable whose value needs to be predicted. In this toy example, we have the set of label values $L = \{\text{“}GA\text{”}, \text{“}NN\text{”}\}$, representing “Generic Algorithm” and “Neural Networks” respectively.

A large amount of work has been done in the area of collective classification in the past few years. Sen et al. divided the approaches in ACCA into two distinct categories [101]: one in which we use a collection of unnormalized local conditional classifiers and one in which we define the collective classification problem as one global objective function to be optimized. Because of the interconnected structure among instances in the network, collective inference has to be used in the learning process. To better discuss these approaches in the following sections, we list the notations that are used throughout this dissertation in Table 2.1.

Table 2.1: Notations used in this dissertation

Symbol used	Definition
G	the graph that defines the set of connected nodes
V	the set of nodes in G
E	the set of edges in G
L	the set of labels for nodes V
Y	the set of target nodes in V which require labeling
X	the set of observed nodes in V with known labels
v_i	the ith member of V
X_i	the ith member of X
Y_i	the ith member of Y
L_i	the label assigned to v_i
C	the set of class that L can be chosen from

2.1.2 ACCA based on Local Conditional Classifiers

When applying local classifiers for collective classification, the most commonly used approximate inference algorithms are the iterative classification algorithm (ICA) and Gibbs sampling (GS).

2.1.2.1 Iterative Classification Algorithm

The original version of the iterative classification algorithm (ICA) for networked data was proposed by Jennifer Neville and David Jensen [77]. In ICA, each instance in the network is initially represented by a feature vector containing two types of *relational attributes*: static relational and dynamic relational. Static relational attributes represent the intrinsic attributes of related objects and they remain constant over classification process. Dynamic relational attributes are inferred from the intrinsic attributes of related objects. An example of the construction of the relational attributes is shown in Figure 2.1. In this example, labels of the connected instances are used to compute the dynamic relational attributes. In order to support supervised classification, aggregation operators such as *count*, *mode*, or *prop* are used to process the dynamic attributes from

the linked instances into a fixed length. ICA dynamically updates the relational attributes of some instances as new inferences are made about related objects. Predictions made with high confidence in initial iterations are kept through the inference process and are used to make subsequent inferences about related objects. The process terminates when a given number of cycles are completed and all classifications have been accepted. Table 2.2 shows the framework of ICA [77].

Compared to independent classification methods, collective inference in collective classification models produces more accurate predictions in networked data [51]. However, the classification accuracy gain in CC is greatly affected by the relational autocorrelation between interconnected data. In networks with low autocorrelations, the dynamic attributes will barely aid classification results. Also, if a classifier can make excellent accurate predictions on the static attributes, there is little room for improvement via collective inference.

Table 2.2: Iterative Classification Algorithm

Iterative Classification Algorithm
1. Build model on fully labeled training set . 2. Apply trained model to test set of N instances. For each iteration i: 1 to m. • Calculate values for dynamic relational attributes. • Use model to predict class labels. • Sort inferences by probability. • Accept k class labels, where $k = N(i/m)$. 3. Output final inferences made by model on test set.

The iterative algorithm has also been applied into other collective models. Lu and Getoor proposed a link-based model derived on ICA that considers the link distributions and the local attributes (i.e., text features) of the linked objects [67]. To capture the link patterns, link features are constructed by considering three sets of related nodes: (1) the set of incoming neighbors (2) the set of outgoing neighbors (3) the set of objects co-cited. Logistic regression is applied as the local

classifier and experiments are tested on two real-world bibliography data sets (CORA and CiteSeer) and one web page dataset (WebKB). Cautious iterative collective classification (*cautious* ICC) proposed in [71] improves the performance of the original ICA by introducing a generalization that cautiously exploits relational instances. When calculating the node's relational attributes, cautious ICC selects only the "best" K of the most confident current label assignments, "commits" those labels, and sets all other labels to unknown. Moreover, in the first iteration of cautious ICC, the relational features are computed using only the links to instances in the training set (via setting all test set labels to unknown in step 1). Both [67] and [71] use a confidence-based ordering derived from Neville and Jensen's algorithm, but they differ in the way that confidence is used: early iterations in [71] utilize only those instance labels that have high confidence, treating others as unknown; whereas the algorithm in [67] uses all the neighboring labels. Experiments show that by modifying the order of accepting label assignments and ignoring assignments with low confident, *cautious* ICC exhibit better classification accuracies.

Macskassy et al. summarized four relational classifiers that use iterative inference for learning in [69]. These relational classifiers take advantage of the first-order Markov assumption on the network structure: only a node's local neighborhood is necessary for classification. Moreover, the first-order Markov assumption is particularly restrictive in the univariate case studies: only the class labels of the local neighbors are necessary. The local network is defined by the user, analogous to the user's definition of the feature set for propositional learning. A prior probability is assigned to entities whose class labels are unknown for certain local classifiers. Now we are going to discuss four relational classifiers mentioned in [69] in more detail.

2.1.2.2 *Relational Neighbor Classifier (RN)*

and Weighted-vote Relational Neighbor Classifier (wvRN)

The Relational Neighbor (*RN*) classifier proposed by Macskassy et al. is a simple relational probabilistic model that makes predictions for a given node based solely on the class labels of its neighbors, without machine learning or additional inherent attributes [68]. *RN* estimates class-membership probabilities by assuming the existence of homophily in the dataset; that is entities connected to each other are similar and likely to belong to the same class. Assuming each instance in the network is only associated with a single class $c \in C$, given an *unlabeled* node ($v_i \in V^U$), the relational-neighbor classifier estimates $P(L_i = c|v_i)$, the class-membership probability of a node v_i belonging to class c, as the (weighted) proportion of nodes in the neighborhood that belong to the same class. We define neighbors N_i as the set of *labeled* nodes that are linked to v_i. Thus:

$$P(L_i = c|v_i) = \frac{1}{Z} \sum_{v_j \in N_i} w(v_i, v_j) \times I(L_j = c), \tag{2.1}$$

where $Z = \sum_{v_j \in N_i} w(v_i, v_j)$. $w(v_i, v_j)$ is the weight of the link between node v_i and v_j and $I(.)$ is an indicator variable.

Instead of making a hard labeling during the inference procedure, the weighted-vote relational neighbor classifier (*wvRN)* extends *RN* by tracking changes in the class membership probabilities. *wvRN* estimates $P(L_i|v_i)$ as the (weighted) mean of the class membership probabilities of the entities in the neighborhood (N_i):

$$P(L_i = c|v_i) = \frac{1}{Z} \sum_{v_j \in N_i} w(v_i, v_j) \times P(L_j = c|N_j), \tag{2.2}$$

where Z is the usual normalization factor.

In both *RN* and *wvRN*, entities whose class labels are unknown are either ignored or are assigned a prior probability, depending on the choice of the local classifier. Since only a small

portion of the nodes in G have known labels, a collective inference procedure is needed to propagate the label information through the network to related instances, using either the *RN* classifier or *wvRN* classifier in its inner loops. Both *RN* and *wvRN* perform surprisingly well on relational datasets, even compared to more complex models, such as the Probabilistic Relational Model and Relational Probabilistic Tree [68].

2.1.2.3 Class-distribution Relational Neighbor Classifier (CDRN)

Simple *wvRN* assumes that neighboring class labels are likely to be the same. Learning a model of the distribution of neighbor class labels is more flexible, and may lead to better discrimination. Following the work of Perlich and Provost [90], *CDRN* defines node v_i's class vector $CV(v_i)$ to be the vector of summed linkage weights to the various (known) classes, and class c's reference vector $RV(c)$ to be the average of the class vectors for nodes known to be of class c. Specifically:

$$CV(v_i)_k = \sum_{v_j \in \mathcal{N}_i, L_j = c_k} w_{i,j}, \tag{2.3}$$

where $CV(v_i)_k$ represents the kth position in the class vector of node v_i and $c_k \in X$ is the kth class. Based on these class vectors, the reference vectors can then be defined as the normalized vector sum:

$$RV(c) = \frac{1}{V_c^K} \sum_{v_i \in V_c^K} CV(v_i) \tag{2.4}$$

where $V_c^K = \{v_i | v_i \in V^K, L_i = c\}$.

Neighbors in V^U are ignored in the training process. For prediction, estimated class membership probabilities are used for neighbors in V^U, and Equation 2.4 becomes:

$$CV(v_i)_k = \sum_{v_j \in \mathcal{N}_i} w_{i,j} P(L_j = c_k | \mathcal{N}_j) \tag{2.5}$$

Given $v_i \in V^U$, the class-distribution relational-neighbor classifier estimates the probability of class membership, $P(L_i = c|\mathcal{N}_i)$, by the normalized vector similarity between v_i's class vector and class c's reference vector:

$$P(x_i = c|\mathcal{N}_i) = \text{sim}(CV(v_i), RV(c)) \tag{2.6}$$

where $\text{sim}(a, b)$ is any vector similarity function(such as L_1,L_2 and cosine), normalized to lie in the range [0,1].

*2.1.2.4 Network-only Bayes Classifier (*NBC*)*

Network-only Bayes classifier (NBC) is another relational classifier that is based on the algorithm described by Chakrabartiet et al. [21]. Given a single node v_i in V^U, NBC uses multinomial Naïve Bayesian classification based on the classes of v_i's neighbors:

$$P(L_i = c|\mathcal{N}_i) = \frac{P(\mathcal{N}_i|c) \cdot P(c)}{P(\mathcal{N}_i)}, \tag{2.7}$$

where

$$P(\mathcal{N}_i|c) = \frac{1}{Z} \prod_{v_j \in \mathcal{N}_i} P(L_j = \tilde{L}_j|L_i = c)^{w_{i,j}} \tag{2.8}$$

where Z is a normalizing constant and $\tilde{L}_j$ is the class observed at node v_j. The reason NBC is called "network-only" is to emphasize that it does not need to use any local node attributes for inference.

*2.1.2.5 Network-only Link-based Classifier (*NLB*)*

Network-only link-based classifier (NLB) is a network-only derivative of the link-based classifier [67]. It creates a feature vector for a node by aggregating the labels of neighboring nodes, and then uses logistic regression to build a discriminative model based on these feature

vectors. This learned model is then applied to estimate $P(L_i = c|\mathcal{N}_i)$. Same as in NBC, local attributes (e.g., text in document network) are not considered for learning NLB in [69], .

2.1.2.6 Gibbs Sampling

Another approximate inference techniques based on local calculation is Gibbs sampling (GS). Gibbs sampling is one of the simplest Markov chain Monte Carlo (MCMC) algorithms for obtaining a sequence of random samples from multivariate probability distribution (i.e. the joint probability distribution of two or more random variables). It is widely regarded as one of the most accurate approximate inference procedures [39]. Gibbs sampling can be adopted in collective classification in order to sample for the best label estimate of L_i given all the values for the nodes in the neighborhood using the local classifier F for a fixed number of iterations [71, 69]. The pseudocode for Gibbs sampling is shown in Figure 2.3.

One of the benefits of both ICA and GS is that they simply make use of any local classifier. Previous experiments have found that some local classifiers tend to produce higher accuracies than others, at least in some application domains. For instance, [71] reports that Naive Bayes has a better performance than K-NN in bibliography data sets and web page classification problems, whereas [67] mentioned that logistic regression outperforms Naïve Bayes in their link-based classification model. Another type of approximate collective classification algorithm is based on a global formulation (e.g., the joint probability of the nodes in the whole network). Firstly, data variables are represented in a pairwise Markov Fields (MRF), and then collective inference is adopted to learn the best assignment of the unobserved variables that maximize the global objective function. In order to better explain how ACCA using global formulation works, we will start with the pairwise Markov Fields and discuss two important collective inference techniques: (loopy) belief propagation (LBP) [89] and relaxation labeling (RL) [96] used in MRF.

A pairwise Markov Fields (pairwise MRF) is a undirected graph where each node denotes

Table 2.3: Pseudocode for Gibbs sampling [69]

1. Initialize $v_i \in V^U$ using the local classifier model F. Specifically, for $v_i \in V^U$:
   - $\hat{c_i} \leftarrow F(v_i)$, where $\hat{c_i}$ is a vector of probabilities representing F's estimates of $P(L_i|\mathcal{N}_i)$. $\hat{c_i}(k)$ is the kth value in $\hat{c_i}$, which represents $P(L_i = c_k|\mathcal{N}_i)$.
   - Sample a value c_s from $\hat{c_i}$, such that $P(c_s = c_k|\hat{c_i})$.
   - Set $x_i \leftarrow c_s$.
2. Generate a random ordering, O, of vertices in V^U.
3. For elements $v_i \in O$ in order:
   - Apply the relational classifier model: $\hat{c_i} \leftarrow F(v_i)$.
   - Sample a value c_s from $\hat{c_i}$, such that $P(c_s = c_k|\hat{c_i}(k))$.
   - Set $L_i \leftarrow c_s$.
4. Repeat prior step 200 times without keeping any statistics. This is known as the burning period.
5. Repeat again for 2000 iterations, counting the number of times each X_i is assigned a particular value $c \in X$. Normalizing these counts forms the final class probability estimates.

a random variable and every edge denotes a correlation between a pair of random variables [112]. We will still use the previously defined notation $G = (V, E)$ to represents the graph of pairwise MRF, where V consists of two types of random variables, the unobserved variables, Y, which need to be assigned values from label set L, and observed variables X whose values are known. The objective function that we aim to maximize is defined on a bunch of clique potentials, which are functions defined over a small set of random variables to calculate the potentials of the joint assignments of these random variables. Let Φ_i denotes the clique potential, then $\phi_i(y_i)$ in the

example shown in Figure 2.1 can be computed as:

$$\phi_i(y_i) = \Psi_i(y_i) \prod_{(Y_i, X_j) \in E} \Psi_{ij}(y_i) \tag{2.9}$$

where Ψ_i and Ψ_{ij} denote the clique potentials on variable L and each $(Y_i, X_j) \in E$ (or $(Y_i, Y_j) \in E$) respectively.

Given a pair (G, Ψ), where G is a graph and Ψ is a set of clique potentials with Φ_i and Ψ_{ij} as defined previously, and an assignment y to all the unobserved variables Y, the pairwise MRF is associated with the probability distribution:

$$P(y|x) = \frac{1}{Z(x)} \prod_{Y_i \in Y} \phi_i(y_i) \prod_{(Y_i, Y_j) \in E} \Psi_{ij}(y_i, y_j) \tag{2.10}$$

where x denotes the observed values of X and $Z(x)$ is the normalization factor defined as:

$$Z(x) = \sum_{y'} \prod_{Y_i \in Y} \phi_i(y'_i) \prod_{(Y_i, Y_j) \in E} \Psi_{ij}(y'_i, y'_j) \tag{2.11}$$

Given a pairwise MRF, extracting the best assignments to each unobserved variable in the network becomes conceptually simple. However, it becomes intractable in large networks when computing one marginal probability requires summing over an exponentially large number of terms. Therefore, approximate inference algorithms are adopted to address this problem. Two important approximate inference algorithms that have been widely used in ACCA with joint relational models are (loopy) belief propagation (LBP) [89] and relaxation labeling (RL) [96]. Both methods use the estimated class distributions directly, rather than the hard labeling used by iterative classification. Details of LBP and RL are explained in the following section.

2.1.2.7 *Loopy Belief Propagation*

Intuitively, loopy belief propagation (LBP) is an iterative message-passing algorithm that can be defined as follows:

$$m_{i \to j}(y_j) = \alpha \sum_{y_i \in C} \psi_{i,j}(y_i, y_j)\phi_i(y_i) \prod_{Y_k \in \mathcal{N}(Y_i) Y_j} m_{k \to i}(y_i), \forall y_j \in L \tag{2.12}$$

$$b_i(y_i) = \alpha \phi_i(y_i) \prod_{Y_j \in \mathcal{N}(Y_i)} m_{j \to i}(y_i), \forall y_i \in L \tag{2.13}$$

where $m_{j \to i}$ is a message sent by Y_i to Y_j, α is a normalization constant and $\mathcal{N}(Y)$ denotes the set of target random variables in the neighborhood of Y. The algorithm proceeds by making all the target random variables Y_i communicate messages with its neighbors in $\mathcal{N}_i \bigcap y$ until the messages stabilize. When the messages stabilize we compute $b_i(y_i)$ (approximate marginal probability) for every target random variable Y_i for every label y_i and assign Y_i the label y_i with the highest marginal probability. Table 2.4 describes the pseudocode.

2.1.2.8 *Relaxation Labeling*

Relaxation labeling (RL) is another inference mechanism that is widely used in collective classification, and was firstly introduced by Chakrabarti et al. in [21]. Rather than treating G as a specific labeling "state" at each point (as Gibbs sampling does), RL retains the uncertainty, and keeps tracking of the current probability estimations for target nodes Y. Moreover, the estimates of the test nodes are not updated individually in RL. All the current estimates at step t are kept such that at step $t + 1$ all nodes will updated based on the estimates from step t. The relaxation labeling algorithm is shown in Table 2.5.

Table 2.4: Pseudocode for Loopy Belief Propagation [101]

for each $(Y_i, Y_j) \in E(G)$ s.t. $Y_i, Y_j \in y$ **do**
  for $y_i \in L$ **do**
    $m_{i \to j}(y_j) \leftarrow 1$
  end for
end for
repeat // perform message passing
  for each $(Y_i, Y_j) \in E(G)$ s.t. $Y_i, Y_{j \in y}$ **do**
    for each $y_j \in L$ **do**
      $m_{i \to j}(y_j) = \alpha \sum_{y_i \in C} \psi_{i,j}(y_i, y_j) \phi_i(y_i) \prod_{Y_k \in \mathcal{N}(Y_i) Y_j} m_{k \to i}(y_i)$
    end for
  end for
until $m_{i \to j}(y_j)$ stop showing any change
for each $Y_i \in y$ **do** // compute beliefs
  for each $y_i \in L$ **do**
    $b_i(y_i) = \alpha \phi_i(y_i) \prod_{Y_j \in \mathcal{N}(Y_i)} m_{j \to i}(y_i)$
  end for
end for

Collective classification approaches with joint relational models have attracted a lot of attention in recent years. Different than the models that concentrate on variable equilibrium of local structures, the joint relational models are constructed to exploit the autocorrelation of the entire network. Essentially, joint relational models aim to estimate a joint probability distribution over the entire relational data set and collectively infer the labels of unobserved instances. Probabilistic graphical models for estimating the joint probability distribution of relational data that consist of non-independent and even heterogeneous instances are referred to as probabilistic relational models (PRMs) [34].

Neville et al. defined PRMs as a combination of three graphs in [75]: the data graph G_D, the model graph G_M, and the inference graph G_I, corresponding to the skeleton, model, and ground graph as outlined by Heckerman et al. [48]. The data graph $G_D = (V_D, E_D)$ presents the input network, where nodes are the objects in the data and edges represent the relationships among the

objects. Each node $v_i \in V_D$ and edge $e_j \in E_D$ is associated with a type $T(v_i) = t_{v_i}, T(e_j) = t_{e_j}$. Additionally, the type $t \in T$ of each node or edge is associated with a number of attributes X_t. Each node v_i and link e_j are also associated with a set of attribute values, $x_{v_i}^{t_{v_i}}$ and $x_{e_j}^{t_{e_j}}$, determined by their types, t_v and t_e, respectively. A PRM represents a joint probability distribution over all attribute values in the data graph, $x = \{x_{v_i}^{t_{v_i}} : v_i \in V_D, T(v_i) = t_{v_i}\} \bigcup \{x_{e_j}^{t_{e_j}} : e_j \in E_D, T(e_j) = t_{e_j}\}$.

The model graph $G_M = (V_M, E_M)$ represents the dependencies among attributes at the level of item types. Attributes of an item depends probabilistically on other attributes of the same item and on attributes of other objects or links it relates to in G_D. Each node in V_M corresponds to an attribute $X_i^t \in X^t$ where $t \in T$. The attributes with the same type in G_D are tied together. Consequently, G_M describes how different type attributes depend on each other and the conditional

Table 2.5: Pseudo-code for relaxation labeling [69]

1. For $v_i \in V^U$, initialize the prior: $\hat{c_i}^0 \leftarrow F(v_i)$, where $\hat{c_i}$ is defined as in Table 2.3.

   For the case study, the local classifier model returns the unconditional marginal class distribution estimated from x^K.

2. For elements $v_i \in V^U$:

   - Estimate L_i by applying the relational model:
     $\hat{c_i}^{t+1} \leftarrow \mathcal{M}_R(v_i^{(t)})$,
     Where $\mathcal{M}_R(v_i^{(t)})$ denotes using the estimates $\hat{c_j}^t$ for $v_j \in \mathcal{N}_i$, and t is the iteration count. This has the effect that all predictions are done pseudo-simultaneously based on the state of the graph after iteration t.

3. Repeat for T iterations. $\hat{c}^{(T)}$ will comprise the final class probability estimations.

probability distribution (CPD) associated with the nodes in G_M.

The inference graph $G_I = (V_I, E_I)$ represents the probabilistic dependencies among all the variables in a single test set. It can be instantiated by a roll-out process of G_D and G_M. Each item-attribute pair in G_D duplicates the corresponding CPD from G_M. The way that G_M is unveiled form G_I is determined by the relations in G_D. Therefore both G_D and G_M control the structure of G_I.

Over the past decade, several novel PRMs have been proposed [79, 112, 37], which can be generally categorized into directed PRMs, such as relational Bayes networks (RBNs) and undirected PRMs, such as relational Markov models (RMNs). The relational Bayes network [37] extends the traditional graphical model, the Bayes network, into the relational domain by structuring the autocorrelation dependencies in an acyclic manner. Because of this, some domain knowledge is needed when constructing the model. In certain domains, the acyclic relationship is unknown or does not exist. For example, it is hard to decide the casual direction of the dependency between the topics among hyper-linked web pages. Therefore, the capability of directed PRMs may be limited in those relational domains. Rather than building one BN and using it separately for each entity, RBNs directly link shared variables through generating one large network with connected entities, where collective inferencing can be performed. The calculation of RBNs over variables $\mathbf{X}$ uses a directed model graph $G = (V, E)$, where each node $v \in V$ corresponds to an $X_k^t \in \mathbf{X}$, and a set of conditional probability distributions P for each variable based on their parents ($P(x_k^t | pa_{x_k^t})$). Given (G, P), the joint probability for a set of values $\mathbf{x}$ is computed as a product over the item types T, the attributes of that type X^t, and the items of that type v, e:

$$p(\mathbf{x}) = \prod_{t \in T} \prod_{X_i^t \in X^t} \prod_{v:T(v)=t} P(x_{vi}^t | pa_{x_{vi}^t}) \prod_{e:T(e)=t} P(x_{ei}^t | pa_{x_{ei}^t}) \tag{2.14}$$

Taskar et al. proposed Relational Markov Networks (RMN) [112], which are built on an undirected graphical model. Instead of defining a joint probability model on a collection of related

instances, RMN constructs discriminative models using conditional Markov networks which are simply Markov networks reformatted to model a conditional distribution. To extend the framework of Markov networks to the relational setting, the conditional distribution in RMN is calculated over all the labels of target instances given the relational structure and the content attributes. Due to the complexity of relational model graphs in practice, the majority of PRMs use approximate inference. Here, *belief propagation*, is adopted in order to make the inference task in RMN tractable in large networked datasets.

Approximations are also needed in other statistical relational learning models. The Relational Dependency Network (RDN) [79], which is an extension of Dependency Network (RN) in a relational setting, approximates the joint distribution over the attribute value of a relational dataset with a set of conditional probability distributions (CPDs) that are learned using collective inference. Gibbs sampling is used as the approximate inference method in RDN. The strength of RDNs is due to the use of pseudolikelihood learning techniques, which estimate an efficient approximation of the full joint distribution.

Other relational models that use conditional relational learners as a subcomponent to approximate the individual CPDs consistent with the joint distribution have been represented in relational Bayesian classifiers (RBCs) [82] and relational probability trees (RPTs) [81]. RBCs extend Bayesian classifiers to a relational setting. RBCs treat heterogeneous relational sub-graphs as a homogenous set of attribute multisets. RPTs are selective models that extend classification trees to a relational setting. Similarly, RPTs also treat heterogeneous relational subgraphs as a set of attribute multisets. Different than RBCs, rather than modeling the multisets as independent values drawn from a multinomial, the RPT algorithm projects a set of values into a single feature value using aggregation functions. We refer interested reader to the references for more details.

2.2 Node Classification in Multi-relational Networks

In the literature, the networks that involve heterogeneous types of interactions or actors are referred as heterogeneous networks, which can further be categorized into two types [107]: (1) Multi-Dimensional Networks, which contain multiple types of interactions between the same set of users. A multi-dimensional network is also called a multiplex network, multi-relational network, or labeled graph in various domains (2) Multi-Mode Networks, which involves heterogeneous actors connected by various types of links with each mode representing one type of entity.

2.2.1 Learning across multiple networks

Recently more work has been done on multi-relational graphs. The simple relational data mining (RDM) algorithms look for patterns in the data by exploring the relational structure which is usually represented in a single data table. Multi-relational data mining (MRDM) approaches look for patterns that involve multiple tables (relations) from a relational database [30]. RDM tools can be applied directly to multi-relational data in order to find corresponding relational patterns. The simplest way is propositionalization, which converts multiple relational data into a single flat data relation, using joins and aggregations. This, however, could lead to the generation of a huge, undesirable "universal relation" and loss of meaning or information [44, 30].

In order to make the best use of the existing algorithms for single-relational networks to solve problem in multi-relational networks, Rodriguez et al. proposed a multi-relational path algebra for defining abstract paths through a multi-relational network in order to derive a single-relational network representing vertex connectivity based on such path descriptions [95]. The multi-relational path algebra operates on $n \times n$ adjacency matrix "slices" of a $n \times n \times m$ three-way tensor representation of the multi-relational networks, and generates a $n \times n$ (weighted) path matrix. Consequently, all homogeneous network algorithms can then be applied on this single-relational representation to yield meaningful results. Kato et al. proposed an interesting approach

for transductive inference on multiple networks where label propagation is adopted over a linear combination of graph Laplacians [53]. In his work, each relation type is considered as a different graph. Moreover, a label propagation algorithm for inferring node labels is proposed in [12]. It computes a feature vector for each node according to the neighboring labels and then applies a 1-nearest neighbor algorithm to find the labels of the unlabeled nodes. This algorithm is regarded as the link-based classification method since only the structure of the multigraph is used in the model.

2.2.2 Learning on a single network with known link types

Other researchers focus on algorithms that can be directly applied on heterogeneous networks. One observation they make is that the labels (types) of connections provide useful information for the classification problem, and can be incorporated with a node's intrinsic or relational features to build a discriminative classifier for better prediction. Goldberg et al. found that in social media, nodes may link to one another even if they do not have similar labels [41]. In their paper, they use two edge types to denote the affinity or disagreement in the class labels of linked objects and incorporate the link type information into a semi-supervised learning model. In [47], a Link Type Relational Bayes Classifier is proposed to predict the node's class labels according to the neighbors' labels as well as their link types. To be specific, the probability of any specific node, x_i, being in a particular class, c_i, is defined as $P_r(x_i = c_i|\mathcal{N}, \mathcal{L})$. That is, the probability of any particular node being in a class is determined by its neighbors, and the set of links that define those neighbors. In this case, the traditional Naive Bayes calculation can be modified to:

$$Pr(x_i = c_i|\mathcal{N}, \mathcal{L}) = \frac{Pr(\mathcal{N}, \mathcal{L}|x_i = c_i)Pr(x_i = c_i)}{Pr(\mathcal{N}, \mathcal{L})} \tag{2.15}$$

The weight associated with each link type then can be easily incorporated into Equa-

tion 2.15:

$$Pr(x_i = c_i|\mathcal{N}, \mathcal{L}) = \prod_{n_i \in \mathcal{N}, l_i \in \mathcal{L}} [\frac{Pr(n_i, l_i|x_i = c_i)Pr(x_i) = c_i}{Pr(n_i, l_i)}] \tag{2.16}$$

Peters et al. extended the *Iterative Classification algorithm* to the task of multi-label classification for multi-relational graphs [91]. They proposed the IMMCA model: Iterative Multi-label Multi-relational Classification Algorithm, that performs iteratively by propagating scores according to the multi-relational structure of the data, and then use the learned propagation scheme to iteratively label each unlabeled node.

2.2.3 Learning on a single network without knowing link types

Unfortunately, the link type information is not always available in the dataset. Kong et al. proposed another extension of ICA [54], *Iterative Classification of Multiple Labels* (ICML) that captures the dependencies among the label sets for a group of linked instances and the dependencies among the multiple labels within each label set simultaneously. Empirical studies on two DBLP datasets demonstrate that the proposed *ICML* approach can effectively boost classification performances over traditional collective classification methods in multi-label relational datasets.

With limited information and the network connectivity, differentiating the connections into distinct groups has become the critical yet difficult problem. A number of methods have been proposed to exploit the shared link pattern embedded in the network structure. Tang et al. proposed the *SocDim* framework to directly address the link heterogeneity problem. Because the same connection can be associated with multiple affiliations and similar actors tend to interact with each other more frequently than dissimilar actors, people who behave in a similar pattern are more likely to be grouped into the same affiliation. Therefore, latent social dimensions can be constructed with each dimension representing a plausible affiliation among actors, which is naturally connected to the "community detection" or "graph partitioning" tasks in social network analysis. A number of approaches for extracting the latent social dimensions have been proposed. In [108], *SocDim* is

extracted from the network via the top eigenvectors of the modularity matrix B defined in Equation 2.17 to capture the potential affiliations of the entity.

$$B = A - \frac{\mathbf{d}\mathbf{d}^T}{2m} \tag{2.17}$$

where A is the interaction matrix with n vertices and m edges. $\mathbf{d}$ is the degree vector, in which the value of d_i refers to the degree of node i. Finally, a discriminant classifier such as SVM or logistic regression is adopted for prediction based on the social dimensions.

Social dimensions can also be extracted using graph partitioning approach, such as the spectral clustering method, on a network graph. In [106], *SocDim* is constructed using the first k smallest eigenvectors of the normalized graph Laplacian $\tilde{L}$. Both modularity maximization and spectral clustering involve the calculation of the top (smallest) eigenvectors of a specific matrix (modularity matrix or graph Laplacian), which may cause a computational complexity problem when applying them to large-scale networks. To address this problem, an approach for extracting social dimensions using an edge-centric clustering scheme is proposed in [109] and scalable k-means is adopted for grouping edges. Essentially, each edge is first represented in a feature-based format, where the nodes that define the edges are used to represent features. Based on the features of each edge, K-means clustering is used to separate the edges. Since the edge feature data is very sparse, the clustering process can be accelerated wisely. In each iteration a small portion of relevant instances (edges) that share features with cluster centroids are identified, and only the similarity of the centroids with the relevant instance need to be recomputed. By using this procedure, k-means partition algorithm is normally faster and more scalable. The clustering task can be completed within minutes even for networks with millions of nodes.

Conventional relational classification models focus on the single-label classification problem, which assumes that each instance is only associated with one label among a finite set of candidate classes. However, one common observation in many real multi-relational networks is

that each instance can be associated with multiple labels. For example in YouTube, users can subscribe to different interest groups. From this point of view, the classification problem in multi-relational networks can be treated as a multi-label classification problem [109, 108, 106]. In the prediction phase, different from existing relational learning methods, collective inference becomes unnecessary as the selected social dimensions have already included the relevant network connectivity information. Moreover, social dimension based approaches are more flexible and allow for the combination of other features for classification, such as user profiles or social content information if they are available. *SocDim* makes it easy to combine the extracted potential group features with the node's intrinsic features and then apply a discriminative learning model for predictions. As shown in [106, 108], this simple combination of network information and user's features allows for integration of data in disparate formats and can lead to more accurate classification in general.

2.3 Link Prediction Problem in Non-overlapping Communities

Another interesting topic in social network analysis is link prediction, which concerns the problem of predicting missing links amongst nodes in a social network, or in the dynamic interactions scenario, the links that will be formed in the future. In most cases, the link prediction problem can be formalized as the following: suppose we have a social network $G = \langle V, E \rangle$, where V represents the nodes in the network, and E is the edge set for node pairs (v_i, v_j) that connect with each other. Given a target node pair (v_m, v_n), the link prediction model aims to predict whether a link exists between them. In the dynamic context, assuming $G[t, t']$ represents the subgraph of G, where $e_{ij} \in E$ represents the interactions between node v_i and v_j that happened between the time interval $[t, t']$ ($t < t'$), the model we are seeking for is to predict a list of edges that are not present in $G[t, t']$, but tend to appear in $G[t1, t1']$ ($t < t' < t1 < t1'$). We often refer to $[t, t']$ as the training interval and $[t1, t1']$ as the test interval.

A variety of techniques for the link prediction problem have been introduced, ranging from

graph theory, metric learning, statistical relational learning to matrix factorization and probabilistic graphical models [66, 60]. Liben-Nowell and Kleinberg had the first comprehensive study on link prediction problem in social networks using the topological features derived from graphs [60]. Generally, link prediction models have been categorized into three areas: (1) the node-wise similarity based methods, which use a similarity measures to determine link existence. (2) models that focus on exploiting (local or global) topological patterns in the network. (3) probabilistic models that are learned from the underlying structure of the observed data in the current network, and regenerate the unobserved part of the network using the learned model.

2.3.1 Node-wise Similarity Based Approaches

Given a node pair (x, y), node-wise similarity based approaches calculate its similarity score, $score(x, y)$. Node pairs with a higher similarity score are more likely to share a link. Usually with a predefined threshold (TH), a link will be estimated to exist between x and y if $score(x, y) > TH$. Many similarity measures have been proposed; examples include information content and mutual information. Even though the similarity score can be calculated based on instances' intrinsic features (e.g., words in a person's blog), most of the existing similarity measures are not designed to measure the node-to-node proximity in the network, and they do not take the network topology into consideration. Here we list several similarity measures that have been widely used in the networked data domain. All of them are constructed solely based on the network structure.

Given a graph G where a node x and its immediate neighbors $\mathcal{N}_x$ is known, a number of approaches have been proposed based on the idea that it is more likely to form a link between nodes x and y if their neighbors $\mathcal{N}_x$ and $\mathcal{N}_y$ have a large overlap. A natural intuition behind this idea is that when nodes x and y represent authors with many colleagues in common, it becomes more likely that they will end up with contacting themselves. Jin et al. defined abstract models for network growth using this principle, in which an edge $< x, y >$ is more likely to form if edges

$< x, z >$ and $< z, y >$ are already present for some z [52]. Most commonly used unsupervised link prediction methods that are based on the topological patterns in the network are shown below:

• *Common neighbors* measures the similarity of node pair (x, y) based on the number of neighbors that x and y have in common: $Comm(x, y) = |\mathcal{N}_x, \mathcal{N}_y|$. As has been verified in the context of collaboration networks [83], the probability of collaboration between two authors is strongly positively correlated with their number of mutual acquaintances in the network.

• *Jaccard's coefficient* is a commonly used similarity metric in information retrieval [99]. It measures the probability that both x and y have a feature f that is randomly selected from the feature set of either x or y. Here, Jaccard's coefficient can be modified to use node's neighbors as features.

$$Jaccard(x, y) = \frac{|\mathcal{N}_x \bigcap \mathcal{N}_y|}{|\mathcal{N}_x \bigcup \mathcal{N}_y|} \tag{2.18}$$

• *Adamic/Adar* is another variation of common neighbors, which weight the impact of neighboring nodes inversely with respect to their total number of connections [friends and neighbors on the web]. The neighbors with fewer connections will have more influence on similarity.

$$AA(x, y) = \sum_{z \in \mathcal{N}_x \bigcap \mathcal{N}_y} \frac{1}{\log|\mathcal{N}_z|} \tag{2.19}$$

• *Preferential attachment* measures the closeness of x and y by the product of the current number of neighbors of x and y. The empirical study in [83] and Barabasi et al. have shown a strong correlation between the probability of coauthorship of x and y and the product of the number of their previous collaborators [5].

$$PA(x, y) = |\mathcal{N}_x| \times |\mathcal{N}_y| \tag{2.20}$$

As mentioned in [60] topological information is quite useful when compared to a random predictor. In particular, the Adamic-Adar and Katz measures appear to be more effective than

the other topological features. Moreover, link prediction performance can be improved when additional structural information is available. Soundarajan and Hopcroft improve the accuracy of similarity-based link prediction methods by incorporating the similarity calculations with community information [102]. First of all, link or node communities are generated based on the training data using greedy modularity optimization (Mod) [14], Infomap (IM) [97], or the Link Communities method (LC) [3]. Experimental results on 10 different real-world datasets demonstrate that the community information often boosts the performance of the baseline similarity metrics (*Common Neighbor* and *Resource Allocation*).

2.3.2 Approaches Exploiting Path Based Topological Patterns

The network topology describes the connectivity pattern between any pair of nodes by the ensemble of all paths between them. The key idea among path-based topological link prediction models is that if there are many paths indirectly connecting node x with node y, it is likely that there is a link connecting them. A number of previous works based on path-based topology have implicitly considered the ensemble of all paths between two nodes. Some of the important approaches are listed below:

• *Katz* defines a measure that directly sums over this collection of paths, exponentially damped by length to count short paths more heavily. This leads to the measure:

$$score(x,y) = \sum_{t=1}^{\infty} \beta^l \cdot |paths_{x,y}^{\langle l \rangle}| = \beta A_{xy} + \beta^2 (A^2)_{xy} + \beta^3 (A^3)_{xy} \tag{2.21}$$

Obviously, a very small β produces a measurement close to CN since the long paths make very little contribution. The similarity matrix can be written as:

$$score(x,y) = (\mathcal{I} - \beta A)^{-1} - \mathcal{I} \tag{2.22}$$

Note that, β must be lower than the reciprocal of the largest eigenvalue of matrix A to ensure the convergence of Equation 2.21.

• *Hitting time*. Given a graph G, a random walker starts at a node x and iteratively moves to one node chosen uniformly at random from its neighboring set $\mathcal{N}(x)$. The hitting time $H_{x,y}$ from x to y is calculated by the expected number of steps for a random walker starting at x to reach y. Because the hitting time is not in general symmetric, a more natural way is to consider the commute time $C_{x,y} = H_{x,y} + H_{y,x}$. Both measures naturally calculate the proximity of node pairs. Thus the prediction score can be calculated by the negated value of $H_{x,y}$ and $H_{y,x}$. One difficulty with hitting time as a measure of proximity is that $H_{x,y}$ is quite small even when y is a node with a large *stationary probability* π_y. To address this problem, normalized versions of the hitting and commute times are defined as follows:

$$score(x, y) := -H_{x,y} \cdot \pi_y \tag{2.23}$$

or

$$score(x, y) := -(H_{x,y} \cdot \pi_y + H_{y,x} \cdot \pi_x) \tag{2.24}$$

• *PageRank* [17] can also be used for link prediction. The PageRank (PR) algorithm of Google fame was introduced to represent the significance of a node in a network based on the significance of other nodes that link to it. If we assume that the probability of linking has a positive correlation between the product of the attributes (here, importance) of the node pair, an assumption implicit in preferential attachment predictions, then the PageRank of the target node represents a useful statistic [25].

• *Rooted PageRank* [60] derives from the original PageRank where the link prediction scores correspond to the probability of visiting the target node during a random walk process from the source. A parameter $\alpha \in [0, 1]$ is the stationary probability of y in a random walk that returns

to x with probability α each step, moving to a random neighbor with probability $1-\alpha$. α helps to avoid getting trapped in directed networks or dense areas.

• *Random Walk with Restart (RWR)* [66]. Consider a random walker starting from node x, who iteratively moves to a random neighbor with probability c and returns to node x with probability $1-c$. If we use q_{xy} to denote the probability this random walker is located at node y in the steady state, then:

$$\vec{q_x} = cP^T\vec{q_x} + (1-c)\vec{E_x}, \tag{2.25}$$

where P is the transition matrix with $P_{xy} = 1/k_x$ if x and y are connected, and $P_{xy} = 0$ otherwise. The solution is straightforward, as

$$\vec{q_x} = (1-c)(I - cP^T)^{-1}\vec{e_x} \tag{2.26}$$

The RWR index is thus defined as

$$score(x,y) = q_{xy} + q_{yx} \tag{2.27}$$

• *SimRank*. Two nodes are similar to the extent that they are joined to similar neighbors. Numerically, this is specified by defining similarity$(x,x) = 1$ and

$$score(x,y) = \gamma \cdot \frac{\sum_{a\in\mathcal{N}(x)}\sum_{b\in\mathcal{N}(y)} score(a,b)}{|\mathcal{N}(x)|\cdot|\mathcal{N}(y)|} \tag{2.28}$$

for a parameter $\gamma \in [0,1]$. SimRank can be interpreted in terms of a random walk on the collaboration graph: it is the expected value of γ^ℓ, where ℓ is a random variable giving the time at which random walks started from x and y first meet.

• *PropFlow* predictor introduced in [61] is also a path-based unsupervised prediction method that models the link prediction probability as being propagated from the source node to the target

nodes within l steps (l is a predefined parameter) using random walk. During the propagation process, the link weights (if available) can be adopted as transition probability, and the random walker selects links based on the link weights. The walk starts at node v_i and terminates when it reaches the target node v_j or revisits any nodes within the distances of l including v_i. The score obtained through this process can serve as an estimation of the likelihood of new links. The main difference between *PropFlow* and *Rooted PageRank* is that *PropFlow* does not require the "restarting" process and is not guaranteed to converge. Because of this, *PropFlow* can be executed very rapidly using modified breath-first search restricted to length l. Table 2.6 shows the *PropFlow* algorithm on weighted, directed network.

[60] mentioned some "meta-approaches" in solving link prediction problem that can be combined with the methods discussed above to improve the prediction performance. We list three meta-approaches as follows:

• **Clustering** methodology can be adopted to improve the quality of a link predictor by deleting edges with less "confidence" in graph G through a clustering procedure, and then running the predictor on the resulting "cleaned-up" subgraph. For instance, in order to compute the confidence value for node x and y, $score(x, y)$, we first calculate $score(u, v)$ for all edges in G, and delete the ($\frac{1}{N}$) fraction of these edges whose the score is the lowest. Then the $score(x, y)$ is recomputed for all pairs $< u', v' >$ on this subgraph. In this way we determine node proximities using only edges for which the proximity measure itself has the most confidence.

• **Unseen bigrams** method augments the estimates of the similarity score between x and y ($score(x, y)$) by taking the nodes z similar to x under measurement $score(z, y)$ into consideration. Suppose we have $score(x, y)$ under one of the measures discussed in previous sections. Let $S_x^{<W>}$ denote the W nodes that are most "similar" to x under $score(x, \cdot)$, and $W \in \mathbb{Z}^+$. The improved score measurement can be defined in terms of the nodes in the combined set:

$$Score_{unweighted}(x, y) =\mid \{z : z \in \mathcal{N}_y \bigcap S_x^W\} \mid \tag{2.29}$$

$$Score_{weighted}(x, y) = \sum_{z \in \mathcal{N}_y \bigcap S_x^W} score(x, z) \tag{2.30}$$

• **Low-rank approximation**. Many approaches mentioned above can be equally well formulated in terms of the graph matrix M. For example, the score value for node x and y ($score(x, y)$) in the common neighbors method can be simply defined by the inner product of

Table 2.6: Pseudocode for *PropFlow* [61]

Algorithm: PropFlow Predictor

Require: network $G = (V, E)$, node v_s, max length l
Ensure: score S_{sd} for all $n \leq l -$ degree neighbors v_d of v_s
insert v_s into *Found*
push v_s onto *NewSearch*
insert $(v_s, 1)$ into S
for *CurentDegree* $\leftarrow 0$ to l **do**
  OldSearch $\leftarrow$ *NewSearch*
  empty *NewSearch*
  while *OldSearch* is not empty **do**
    pop v_i from *OldSearch*
    find *NodeInput* using v_i in S
    SumOutput $\leftarrow 0$
    for each v_j in neighbors of v_i **do**
      add weight of e_{ij} to *SumOutput*
    end for
    Flow $\leftarrow 0$
    for each v_j in neighbors of v_i **do**
      $w_{ij} \leftarrow$ weight of e_{ij}
      Flow $\leftarrow$ *NodeInput* $\times \frac{w_{ij}}{SumOutput}$
      insert or sum $(v_j, Flow)$ into S
      if v_j is not in *Found* **then**
        insert v_j into *Found*
        push v_j onto *NewSearch*
      end if
    end for
  end while
end for

the rows $r(x)$ and $r(y)$ in M. However, it becomes a problem when computations are carried on large scale matrix M. One common technique for handling this problem is to choose a relatively small number k and compute the rank-k matrix M_k that best approximates M with respect to any of a number of standard matrix norms [60]. Consequently, the similarity computation can be calculated on M_k rather than M. Singular value decomposition is a widely adopted method for semantic analysis in the area of information retrieval, and can be used in this case as well.

2.3.3 Probabilistic Based Link Prediction Model

The third type of link prediction model is to learn a set of parameters θ given the observed network $G(V, E)$ and use them to estimate the likelihood of the unobserved node pair. Given a unobserved node pair (v_i, v_j), its corresponding link existence can be determined by $P(e^{<i,j>}|\theta)$. From this point of view, a relational learning model serves as a good tool to address the link prediction problem due to its ability to capture not only the objects' intrinsic information but also the attribute correlations between linked objects. With the learned model, both vertices and edges in the data graph can be re-generated. Taskar et al. adopted one of the collective classification models, relational Markov networks (RMNs), to collectively predict and classify both the object type and the links over the entire link graph [113]. The discriminative relational model aims to maximize the (object and) link labels based on the objects' known attributes (e.g. the words on the page, hyperlinks, and structural features). And finally probabilistic inference, belief propagation, is adopted to the relational model to predict and classify the links in the test set.

We note that some of the above approaches are only designed for connected graphs; since each graph G that we consider has a giant component–a single component containing most of the nodes–it is natural to restrict the predictions for these measures to this component.

2.3.4 Supervised Link Prediction Methods

Due to their simplicity, unsupervised prediction methods have remained popular in the link prediction literature but have been shown to be very sensitive to underlying network properties, such as the imbalance in the size of network communities, and experience difficulty adapting to dynamic interdependencies in the network [61].

Hasan et al. studied the use of supervised learning for link prediction in coauthorship networks [46]. They identify a set of link features that are key to the performance of their supervised learner including (1) *proximity features*, such as keywords in research papers, (2) *aggregated features*, obtained from an aggregation operator, and (3) *topological features*. The combination of these features showed effective prediction performance on two collaborative network datasets. Popescul et al. introduced an alternate approach to generating features. First, they represent the data in a relational format, and candidate features are generated through database queries. Then the features are selected using statistical model selection criteria, and finally logistic regression is applied on the selected features for classification [92].

The work of Popescul et al. shows another interesting approach to integrating different kinds of information. They represent the data in a relational format, generate candidates for features by searching through database queries, select features using statistical model selection criteria and use Logistic Regression using the selected features for classification [92]. A potential problem with this method is that the features are simply generated by aggregation functions of the column values in the result set of the join queries (i.e., count and average). It can be hard to represent more complex features such as cosine similarity between bag-of-words representations using simple SQL aggregation functions.

Lichtenwalter et al. present a high-performance link prediction (HPLP) framework that incorporates a variety of features to represent a given node pair (x, y) [61]. The features include both nodes' in-degree and out-degree, values from basic unsupervised models, such as *common*

neighbors and Jaccard's coefficients, path-oriented measures such as the number of shortest path, the maximum flow that can travel from x to y within k steps as well as the prediction scores from *PropFlow*. Experiments on several social networks demonstrate the advantage of this supervised prediction model that relieves the structural imbalance in sparse networks.

Zheleva et al. found that when social network contains tightly-knit family groups, which can be referred to as family circles, link prediction performance can be improved by making use of the power of overlaying family ties [129]. In additional to the inherent actor attributes and the structural features extracted from social network topology, the proposed link prediction model incorporates the family-circle features based on the likely structural equivalence of participants in these groups. Experiments on datasets extracted from Dogster and Catster confirms that family relationships were very useful in predicting friendship links.

2.3.5 Random Walk based link prediction methods

Recently, some researchers started applying random walk models to solve the link prediction problem. For instance, Backstrom and Leskovec developed a supervised random walk algorithm that combines the information from the network structure with node and edge level attributes and evaluated their method on coauthorship networks extracted from arXiv. The edge weights are learned by a model that optimizes the objective function such that more strength is assigned to new links that a random walker is more likely to visit in the future [4]. However, they only focus on predicting links to the nodes that are 2-hops from the seed node. Liu et al. proposed a similarity metric for link prediction based on type of local random walk, the Superposed Random Walk (SRW) index [64]. By taking into account the fact that in most real networks nodes tend to connect to nearby nodes rather than ones that are far away, SRW continuously releases the walkers at the starting point, resulting in a higher similarity between the target node and the nearby nodes. Apparently this assumption is invalid in DBLP and other scientific collaboration datasets. Similarly Yin et al. estimated link relevance using the random walk algorithm on an augmented social graph

with both attribute and structure information [126]. Their framework leverages both global and local influences of the attributes.

2.3.6 Link Prediction in Weighted Networks

The link prediction task has also been extended to weighted networks. de S$\acute{a}$ and Prud$\hat{e}$ncio investigate the benefits of using weights to improve the performance of supervised link prediction [26]. In their work, the traditional unsupervised link prediction measures in unweighted networks have been extended to capture the influence of the weights in the weighted network. The experiments on DBLP dataset using supervised link prediction revealed satisfactory results when link weights were considered. Murata et al. also solve the link prediction problem on two real-world weighted social networks [74]: Yahoo! Answers and Windows Live QnA (a Question-Answering Bulletin Boards (QABB) dataset). Similar to the approach proposed in [26], in this dissertation the classic unsupervised link prediction methods have been modified to make use of the link weights. Experiments have shown that the weighed link predicted approach outperforms traditional unsupervised methods especially when the target social networks are sufficiently dense.

2.4 Link Prediction Problem in Multi-relational Networks

Most of the link prediction models assume the links in the network are generally homogeneous, meaning there is only one type of link among the nodes. However many real-world systems are complex networks, which may contain different types of instances with heterogeneous interactions. Predicting links in such network can be more complicated. It is well known that different types of social ties have different influences on people. Tang et al. proposed a framework to classify the type of social relationships using learning across heterogeneous networks [104]. Social theories have been incorporated into a machine learning model within the framework, which improves the accuracy of inferring the type of social relationships in a target network, by using knowledge

from a different source network. The relationship prediction is made using a transfer-based factor graph (TranFG) model learned based on four different types of features constructed according to basic social psychological theories (social balance, structural hole, social status and opinion leaders). Loopy belief propagation (LBP) is adopted to approximate the variable distribution in order to maximize the log-likelihood objective function in TransFG model.

Davis et al. proposed a unsupervised approach which is an extension of the common Adamic/Adar method, to predict the heterogenous relationships in multi-relational networks [25]. Specifically, the proposed multi-relational link prediction (MRLP) method applies a weighting scheme for different edge type combinations. The weights are determined by counting the occurrence of each unique 3-node sub-structure in the network, traditionally called triad census. A supervised learning approach is then developed by converting the heterogeneous network into a feature representation. The features represent potential link patterns between the node pair and their common neighbors. Experiments on three real-world networks (YouTube network, Disease-gene network and climate network) demonstrate that the supervised method to link prediction can enhance performance.

Sun et al. proposed a path-based relationship prediction model, *PathPredict*, to study the coauthorship prediction problem in heterogeneous bibliographic networks [103]. First, the meta path-based topological features are symmetrically extracted from the network using measures such as path count and random walk, around the given meta paths. The meta path captures the composition relation over the heterogeneous networks. Logistic regression is then used to learn the weights associated with different topological features that best predict co-author relationships. Lee and Adorna proposed a random walk-based link prediction algorithm on a modified heterogeneous bibliographic network where all edges across heterogeneous objects in the network are weighted by using a combination of different importance measures [58]. Different to their work, in our research we only focus on weighting the heterogeneous collaboration links between authors.

Benchettara et al. cope with the link prediction problem in large-scale two-mode social net-

works by projecting a bipartite graph into two unimodal graphs based on the type of node sets [9]. The link prediction task is performed on both projected graphs using a supervised machine learning model (decision tree) based on a set of topological attributes that characterize nodes' roles as well as their similarity. To capture the structural nature of bipartite graph, they introduced new variations of topological attributes, which are constructed based on their topological attributes computed in the projected graphs, to measure the likelihood of two nodes to be connected. Experiments are carried out on two real-world datasets, the DBLP bibliographical database and data from an online music e-commerce site, which demonstrate the superior performance of the new metrics using the projected graph over the original bipartite graph.

2.5 Work Combining Collective Classification and Link Prediction

Most of previous work studied the problem of node classification and link prediction independently. Collective classification on node labels is performed with the assumption that all the links in the network are known whereas link prediction assumes the node set is complete. Real world data collections usually contains a number of missing and incorrect attributes and links, which could negatively affect the performance of both node classification and link prediction. Bilgic et al. proposed a method that iteratively performs both collective node classification and link prediction such that each task can infer information from the other and improve its own performance simultaneously [13]. Several important factors have been considered during the inference procedure: (1) only the information (label assignments and link distribution) with high confidence is exchanged during each iteration. Information with a confidence value lower than a given threshold is discarded to avoid adversely affecting the performance of the other; (2) in the iterative algorithm in collective classification, all label predictions of the test nodes are updated to the current most likely value after the "bootstrapping" phase.

Relational classifiers can fail when the unlabeled nodes have few labeled neighbors to sup-

port learning and collective inference. Gallagher et al. proposed a novel approach (*GhostEdgeNL*) that combines aspects of statistical relational learning and semi-supervised learning to address the within-network classification in networks with sparse links [35]. Under the assumption that nodes which are "closer" (similar) in the network tend to have a greater relationship dependency with their class labels. *GhostEdgeNL* method deliberately adds "ghost" links between unlabeled nodes and labeled nodes to make use of the label information already existent in the network. The "ghost" links are constructed based on the nodes' proximity, which is calculated by a variation of the fast random walk with restart (RWR) method [114] on the square format of graph matrix ($B = A \times A$) to avoid the problem of varying degree of label consistency (i.e. male-female edges in dating relationships graph). Experiments on a range of real-world datasets demonstrate that the performance of *GhostEdgeNL* is superior than other approaches such as conventional collective classification and semi-supervised approaches.

2.6 Other research topics related to this dissertation

2.6.1 Community Detection in Multi-relational Networks

Another interesting topic in social network analysis is *community detection*, whose purpose is to find the non-overlapping "communities" or groups of related properties such that the connections within the group are much more dense than those between the groups. While much work has been done in detecting communities in homogeneous networks [86, 85, 33], community detection in multi-relational networks becomes an attractive yet more complicated problem since instances in the network can now belong to more than one community.

Baumes et al. provide an efficient algorithm for initializing the seed clusters and performing the iterative improvements on the clustering results [7]. The proposed algorithm contains two steps for finding the locally optimal subgraphs: (1) initializing the seed clusters (2) improving the current clusters until one arrives at a locally optimal collection of clusters. The new seed algo-

rithm proposed in this paper, Link Aggregate (LA), modified the former Rand Removal algorithm (RaRe) proposed in [8] by first ordering the nodes according to some criterion (i.e., decreasing PageRank), and proceeding with the clustering sequentially according to this ordering. A node is added to any cluster as long as adding it improves the cluster density. If the node is not added to any cluster, it becomes a new cluster. In the refining step, the original algorithm IS explicitly constructs a cluster that is a local maximum w.r.t. a density metric. Starting with a random candidate cluster, IS algorithm updating the cluster members by adding or deleting one node at a time as long as the metric strictly improves. The algorithm terminates when no further improvement can be achieved with a single change. The original process consists of iterating through the entire list of nodes over and over until the cluster density cannot be improved. The new algorithm IS^2 works efficiently compared to the original algorithm IS by only taking the members of cluster and their immediate neighbors into consideration, since the cluster's density is only related to those objects. The algorithm running time can improve significantly when the neighborhood of a cluster is much smaller than the entire graph.

Wang et al. proposed an approach to discover overlapping communities based on the metadata (user-tag subscription information) instead of user-user links [118]. They found this user-tag structure is more informative for community detection and a co-clustering framework is used to take the advantage of correlations between users and tags, and cluster these two entities simultaneously. Tang et al. present an efficient and effective approach called MROC (*extraction of Multi-Resolution Overlapping Communities*), to extract overlapping communities at different resolutions, and then utilize them as features for behavior prediction [111]. In order to extract communities at a variety of resolutions, MROC adopts an agglomerative hierarchical clustering starting from communities at the highest resolution, where the node and its neighbors are treated as base communities. In particular, a node i with degree d_i is associated with up to d_i different communities. Then two communities with the largest similarity are recursively merged together until certain criteria are satisfied (i.e. community sizes exceed a threshold). In this work, the Jaccard index is

adopted to calculate the proximity between two communities. Similar to the *Edge-Clustering* algorithm, the computation of the agglomerative hierarchical clustering can also be accelerated by only computing the similarity between one community and the communities who share nodes with it. MROC achieves superior performance compared to *Edge-Clustering* method by considering the regularization on both community size and node's affiliation on the linear SVM model: communities with sparse connections and the influence of nodes associated with many communities are weighted less.

Cai et al. proposed an algorithm that improves the performance for community mining by exploiting the integration of heterogeneous relationships in multi-relational networks [19]. Since different relations play distinct roles in separate tasks, identifying the importance of a relation in one community becomes quite useful in understanding the underlying structure of the community. In the paper, a query-dependent relation extraction algorithm is adopted for discovering the hidden relations. Objects under each relation are characterized by a weighted graph where the weight reflects the relation strength between the two corresponding objects. Then a regression-based community mining algorithm is performed in order to find an optimal linear combined relation which makes the relationship between the intra-community instances as tight as possible while the relationship between the inter-community examples is as loose as possible. The obtained combination can better match the user's desire and consequently leads to better performance on community mining.

Some community detection work applies the idea of partitioning links instead of nodes into discovering community structure in overlapping communities [122]. A node in the original graph is called overlapping if links connected to it are put in more than one cluster. In [3], links are partitioned via hierarchical clustering of edge similarity. Given a pair of links e_{ik} and e_{jk} incident on a node k, a similarity can be computed via the Jaccard Index which is defined as:

$$S(e_{ik}, e_{jk}) = \frac{|N_i \bigcap N_j|}{|N_i \bigcup N_j|} \tag{2.31}$$

where N_i is the neighborhood of node i including i. Single-linkage hierarchical clustering is then used to build a link dendrogram.

Evans extends the line graph, the graph which represents the adjacency between edges in a specific network, to clique graphs [31]. The cliques of a given order are represented as nodes in a weighted graph, and the membership strength of a node i to community c is given by the fraction of cliques containing i which are assigned to c.

Gregory proposed an algorithm, named COPRA (Community Overlap Propagation Algorithm) for finding overlapping community structure in very large networks based on the label propagation technique of Raghavan, Albert, and Kumara [94]. Like the original label propagation algorithm, in COPRA, node labels are propagated between neighboring nodes such that the node's membership reach a consensus eventually. However, since nodes can belong to multiple communities, during the label propagation process, the node's labels whose belonging coefficient are less than a predefined threshold $(1/k)$ are deleted while the coefficients of other labels are normalized to 1.

Tang et al. describe the procedure for detecting communities in multi-dimensional networks, where each dimension represents one facet of diverse interactions [107]. A p-dimensional network can be represented as:

$$A = \{A^{(1)}, A^{(2)}, ..., A^{(p)}\} \tag{2.32}$$

where $A^{(i)} \in \{0, 1\}^{1\times 1}$ indicates the interaction among actors in the ith dimension, and p is the total number of activities. The shared community structure can be discovered by integrating information from multiple dimensions. Integration strategies can be used to detect the communities on the shared network structure. [107] mentioned four types of integration strategies: network integration, utility integration, feature integration, and partition integration. Take the "network integration" method as an example. The intuition behind network integration is quite intuitive, that

one's different interactions contribute their overall connections to others in a union. The average interaction among actors is:

$$\bar{A} = \frac{1}{p} \sum_{i=1}^{p} A^{(i)} \tag{2.33}$$

Within $\bar{A}$, classical community detection in a single-dimensional network can be applied on this unified network.

A number of community detection algorithms for overlapping communities which adopt different techniques to tackle the problem have been proposed in recent years. Algorithms for overlapping community detection are categorized into five classes in [122]: Clique Percolation, Line Graph and Link Partitioning, Local Expansion and Optimization, Fuzzy Detection and Agent Based and Dynamical Algorithms. These categories reflect the means used to identify communities. We refer interested reader to [122, 107] for more detailed descriptions of these algorithms.

2.6.2 Semantic Analysis

The Fiedler embedding technique used in Section 3.3 is strongly related to several popular techniques in area of information retrieval. One is Latent Semantic Analysis (LSA) [57], which aims to extract and represent the contextual-usage of words by statistical computations applied to a large corpus of text. A canonical example of LSA begins with a term-document matrix in which matrix rows correspond to keywords or terms, and matrix columns are documents. LSA extracts the "latent semantics" embedded in the data by mapping the terms and documents into a geometric spaces, after which geometric algorithms can facilitate analysis [57].

When calculating the mapping space, LSA is concerned with documents and terms while Fiedler embedding concentrates on general entities and their similarities. Unlike LSA, Fiedler embedding treats all entities (i.e., document and term) equivalently and as co-located in the same space. From this point of view, Fiedler embedding is also quite similar to the Co-Clustering

Table 2.7: LSA algorithms for term/document embeddings and querying [49]

LSA document embedding:
Given scaled document matrix B.
(1) Compute truncated SVD of B, $U_k \sum_k V_k^T$
(2) Assign the position of document i to be $\sum_k^{1/2} V_k^T e_i$
Querying:
Given document embedding, and query vector q.
(1) Compute query location = $\sum_k^{1/2} U_k^T q$
(2) Return documents nearest to query point (nearest in angular distance).

method [27], which places the instances (document) and their features (terms) in the same space and performs clustering on both entities simultaneously. In Section 3.3, Co-Clustering is adopted as one of comparison methods, and we evaluate its performance on edge-based social features.

2.6.3 Graph-based Semi-supervised Learning

In *within-network* classification, both the unlabeled nodes and labeled nodes co-exist in the same graph, and collective inference is adopted to learn the classification model based on the information of both the labeled and unlabeled instances simultaneously. From this point of view, *within-network* classification is closely related to graph-based semi-supervised learning [131]. Traditional machine learning approaches only use a set of labeled objects to train the classifier. However, labeled instances are often difficult, expensive, or time consuming to obtain while unlabeled data may be relatively easy to collect. Semi-supervised learning was proposed to address this problem by using large amount of unlabeled data, together with the labeled data, to build better classifiers.

Graph-based semi-supervised learning methods have attracted great attention. These methods start with a graph where the nodes are the labeled and unlabeled data points, and (weighted) edges reflect the similarity of nodes [127, 132]. Different similarity functions can be adopted to construct the weighted graph in the form of "fully connected graph", "sparse graph", "KNN-graph"

and so on. Then, based on the assumption that nodes connected by a large-weight edge tend to have the same label, the label of one node can be propagated to its neighboring nodes according to their proximity. The probabilistic transition matrix T can be defined by:

$$T_{ij} = P(i \rightarrow j) = \frac{w_{ij}}{\sum_{k=1}^{n} w_{ik}} \tag{2.34}$$

where T_{ij} is the probability of transit from node i to node j, and w_{ij} is the similarity between node i and j. The label propagation algorithms is as follows:

1. Propagation $Y \leftarrow PY$
2. Row-normalize Y to maintain the class probability interpretation.
3. Clamp the labeled data. Repeat from step 1 until Y converges.

Step 3 is critical. Instead of letting the labeled data points fade away, we clamp their class distributions to $Y_{ic} = \delta(y_i, c)$, so the probability mass is concentrated on the given class. The purpose of the second step is to make prediction persistent to the label information from labeled data. If this structure of data fits the classification goal, then the algorithm can use unlabeled data to help learning.

This label propagation scheme can also be formularized into a probabilistic framework. Zhu et al. proposed a semi-supervised learning approach that is based on a Gaussian random field model [133]. Given a weighted graph with edge weights encoding the similarities between instances (both labeled and unlabeled data), the learning problem is then represented in terms of a Gaussian random field on this graph with the energy function showing below:

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij}(f(i) - f(j))^2 = f^{\mathrm{T}} \Delta f \tag{2.35}$$

Here Δ is the the graph Laplacian, and f is a real function over the nodes. The minimum solution

of this energy function is proved to be harmonic [133], thus the learning process can be performed efficiently using matrix methods or belief propagation.

Wang et al. assumed that the label of each data can be linearly reconstructed from its neighbors' labels, and proposed a novel approach called Linear Neighborhood Propagation (NLP) [115] to solve the semi-supervised classification problem. LNP method follows similar optimization techniques as are used in Locally Linear Embedding (LLE) [98] to measure the inconsistency between the labels of the nodes and the corresponding neighbor. The nodes' local neighbor structure is adopted for recovering the labels of the unlabeled data by solving a linearly constrained quadratic optimization problem. With little modification, NLP can be easily extended to the multi-class problem and out-of-sample data. Interested readers can refer to a comprehensive survey by Zhu [131] to find more on graph-based semi-supervised learning.

2.6.4 Multi-label Classification

Multi-label classification (MLC) is a variant of classification where each instance is associated with multiple labels. Given a set of training samples, each of which is associated with a set of labels, MLC aims to learn a model that outputs a bipartition of the labels into those relevant and irrelevant with respect to a query instance. One simple way of addressing multi-label learning is to transform the multi-label classification problem into a set of independent, single-label classification problems, e.g., the most intuitive one-vs-rest learning methods [59]. More sophisticated approaches focus on exploiting the correlations between different labels to improve the label set prediction performance. For instance, Guo et al. proposed a generalized conditional dependency network model for multi-label classification in [42]. The proposed conditional dependency network exploits the dependencies of multiple labels, and the conditional distributions are defined using binary classifiers.

2.7 Public Datasets

2.7.1 Real-world datasets

This section summarizes several relational datasets that are publicly available and have been widely used in solving node classification or link prediction problems in various domains.

Bibliographic dataset:

• **CORA**[1]: A citation dataset which comprises computer science research papers in seven different machine learning areas (*Case Based*, *Genetic Algorithms*, *Neural Networks*, *Probabilistic Methods*, *Reinforcement Learning*, *Rule Learning* and *Theory*.) [70]. CORA contains the citation information between papers as well as the topic and words information associated with each paper.

• **Citeseer**[2]: A collection of research papers drawn from CiteSeer [67]. The documents are labeled with one of the following class labels (*Agents*, *AI*, *DB*, *IR*, *ML* and *HCI*) and each document cites or get cited by another document in the corpus.

• **WebKB**[3] contains WWW-pages collected from computer science departments of various universities in January 1997 by the WebKB project [24]. The pages have been manually separated into 7 categories: *course*, *department*, *faculty*, *project*, *staff*, *student*, or *other*.

Social Network dataset:

• **Facebook**[4] is a subset of the largest social networking website [121].

• **Terrorists**[5] is a social network data set containing information about terrorists and their relationships [121].

---

[1] `http://www.cs.umass.edu/mccallum/data/cora-refs.tar.gz`

[2] `http://www.cs.umd.edu/sen/lbc-proj/data/citeseer.tgz`

[3] `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/webkb-data.gtar.gz`

[4] `http://dvn.iq.harvard.edu/dvn/dv/t3`

[5] `http://www.cs.umd.edu/sen/lbc-proj/data/TerroristRel.tgz`

• **Reality Mining** Dataset[6] contains information about the activities of one hundred human subjects over the course of nine months and represents approximately 500,000 hours of data on users' location, communication and device usage behavior. Survey data (Network data) was collected at the middle of the 9-month study, The dataset includes call logs, Bluetooth devices in proximity, cell tower IDs, application usage, and phone status (such as charging and idle).

Recommendation System:

• **Netflix**[7], MovieLens[8] and EachMovie[9] are three widely used movie recommendation datasets.

Protein Interaction:

• **KEGG PATHWAY**[10] is a collection of manually drawn pathway maps representing our knowledge on the molecular interaction and reaction networks for metabolism.

Other Test Beds:

• **Enron** emails dataset[11] was collected and prepared by the CALO Project (A Cognitive Assistant that Learns and Organizes). It contains email data from about 150 users, mostly the senior management of Enron. In link type prediction experiments in [104], two types of relationships, i.e., manager-subordinate and colleague, are crawled from the original dataset.

• **HEP-TH** (high-energy physics citations from arXiv)[12]. The HEP-Th database presents information on papers in theoretical high-energy physics. The object and link properties such as title, authors and journal were derived from the abstract and citation files provided for the 2003 KDD Cup competition.

---

[6] http://reality.media.mit.edu/download.php

[7] http://www.netfixprize.com

[8] http://www.grouplens.org/

[9] http://www.cs.cmu.edu/lebanon/IR-lab.htm

[10] http://www.cs.cmu.edu/lebanon/IR-lab.htm

[11] http://www.cs.cmu.edu/~enron

[12] http://kdl.cs.umass.edu/data/hepth/hepth-info.html

Datasets that contain multi-relational information:

• **BlogCatalog**[13]: A blog in BlogCatalog can be associated with various pieces of information such as the categories the blog is listed under. Each blog category can be treated as a label that denotes the blogger's personal interest. Moreover, the blogger can specify his/her social connections with other bloggers. The blogCatalog dataset contains 39 categories, and the average number of categories that each instance (blog) belongs to is 1.6 [109].

• **YouTube**[14] is a popular website for sharing videos. Each user in YouTube can subscribe to different interest groups and add other users as his contacts. This dataset contains a total of 47 interest groups [109].

• **Flickr**[15] is collected from the popular photo sharing site Flickr with total of 195 interest groups, and each user in Flickr can subscribe to more than one interest group [109].

• **DBLP**[16]: The DBLP dataset provides bibliographic information for millions of computer science references. Each author in the dataset can have multiple research interests, and collaborate with other researchers in different research areas.

• **IMDb**[17]: The IMDb dataset contains a variety of information about directors, actors, actress and plots of a movie. Each movie in the dataset can be assigned with a subset of multiple movie genres among 27 candidate genres.

• **Epinions**[18] is a network of product reviewers. Each user on the site can post a review on any product, and other users can rate the review with trust or distrust.

• **Slashdot**[19] is a site for sharing technology related news. In 2002, Slashdot introduced the

[13] http://www.public.asu.edu/~ltang9/social_dimension.html

[14] http://www.public.asu.edu/~ltang9/social_dimension.html

[15] http://www.public.asu.edu/~ltang9/social_dimension.html

[16] http://www.informatik.uni-trier.de/~ley/db/

[17] ftp://ftp.fu-berlin.de/pub/misc/movies/database/

[18] http://www.trustlet.org/wiki/Epinions_datasets

[19] http://slashdot.org/

Table 2.8: Real-world datasets used in the node classification and link prediction problems

	Node Classification	**Link prediction**
Homogeneous networks	[1-6]	[1-12]
Heterogeneous networks	[13-17]	[6],[11],[13-16],[18-20]

Slashdot Zoo which allows users to tag each other as friends (like) or foes (dislike). The data set is comprised of 77,357 users and 516,575 edges of which 76.7% are friend relationships.

• **Arnetminer**[20] is the dataset crawled from Arnetminer.org [105]. The dataset contains researcher profile information such as position, affiliation and personal photos. Tang et al. collected advisor-advisee information from researchers' homepage and created a small dataset for their link type prediction experiments [104].

• **NIPS-co-authorship**[21] consists of co-authorship information among researchers on publications in the NIPS conference over the years from 1987 to 2003 (17 years). This dataset includes joint distribution of words and authors [40].

Table 2.8 summarizes the application domains of the above real-world datasets. Datasets are referred by their footnote indices.

2.7.2 Non-overlapping Synthetic Networked Data Generator

The study of networked data has also relied on synthetic datasets as well. Several synthetic data generation algorithms have been proposed in the past few years. Sen et al. proposed an algorithm for generating synthetic datasets that closely follows the algorithm described in [15]. The synthetic data generation starts growing the graph from an empty set of nodes. The number of the nodes desired in the final generated graph contains is provided by the user at the beginning of

[20] `http://arnetminer.org/lab-datasets/profiling/`

[21] `http://ai.stanford.edu/~gal/data.html`

the generation process. The network data generation process is controlled by two parameters: the link density parameter (ld) and homophily (dh). The whole process proceeds as follows:

Table 2.9: Synthetic Data Generator for non-overlapping networked data

Synthetic Data Generator (*numNodes*, *ld*, *numLabels*, *attrNoise*, *dh*)

Set $i = 0$;
G = NULL;
While $i < numNodes$ do
  sample r uniformly at random
  If $r <= ld$ then
    connectNode(G, *numLabels*, *dh*)
  else
    addNodes (G, *numLabels*, *dh*)
    i = i+1
  end if
end while
for i = 1 to $numNodes$ do
  Attributes = genAttributes(v, *Attributes*, *label*, *attrNoise*)
  v: ith node in G
end for
return G

At each step, we either add a link between two existing nodes or create a new node and link this new node to an existing node. The ld parameter controls the total link density of the graph (the higher ld value means higher link density, that is, more links in the graph). Roughly, the final graph should contain $\frac{1}{(1-ld)} \times numNodes$ number of links.

When adding a link, we choose the source node randomly (a set of uniform class priors is used in the experiment), but we choose the destination node using the dh parameter (higher dh value means the higher percentage that a node's neighbor is of the same type) as well as the out-degree of the candidates (preferential attachment, where the nodes with higher out-degrees have better chance of being linked to).

When we are generating a new node, we randomly sample an action class for it (we again used a set of uniform class priors). Then we add a link between the new node and one of the existing nodes, again using the homophily parameter and the degree of the existing nodes.

The algorithm for generating synthetic datasets is outlined in Table 2.9.

2.7.3 Overlapping Synthetic Networked Data Generator

Lancichinetti et al. proposed the LFR benchmark [55] [22] that can be used to create a variety of networks with feature distributions consistent with real-world networks [122]. The LFR model provides a way to generate overlapping communities by introducing heterogeneity into degree and community size distribution, and has been tested on a number of community detection problems [123, 22, 56]. The procedure for generating a graph containing n communities with overlapping nodes using LFR model consists of the following steps:

1. We first define the number of memberships (i.e., classes), v_i, possessed by each node i in the network. Additionally, the node's degree, k_i, is assigned according to a power law distribution with exponent τ_1. A topological mixing parameter μ_t is introduced to describe the percentage of neighbors of node i that do not share any membership in common with i.
2. The community sizes $\{s_\xi\}$ are selected by drawing random numbers from a power law distribution with exponent τ_2. The procedure of assigning the node's community membership is similar to constructing a bipartite network where the two classes are the n communities and N nodes. A node's community membership is assigned subject to the requirement that the sum of community sizes equals the sum of node memberships.
3. Before generating the whole graph, n separate random subgraphs of s_ξ nodes with degree sequences $\{s_i(\xi)\}$ are generated using a configuration model [73] with a rewiring procedure to avoid multiple links.

[22] `http://sites.google.com/site/andrealancichinetti/files`

4. Once each subgraph is built, in order to connect different communities, extra external links are added to the communities without changing the internal degree sequences.

The LFR benchmark provides a rich set of parameters to control the network topology, including the mixing parameter μ_t, the average degree $\bar{k}$, the maximum degree k_{max}, the maximum community size c_{max}, and the minimum community size c_{min}.

2.8 Reader Guide

In this section we will briefly describe the topics that are covered in each of the following chapters:

1. Chapter 3: collective behavior classification in multi-relational networks
   (a) Section 3.1: problem formulation
   (b) Section 3.2: introduction of social features
   (c) Section 3.3: a new proposed social feature representation using Fiedler embedding
   (d) Section 3.4: a new proposed relational model (SCRN)
2. Chapter 4: link prediction in multi-relational networks
   (a) Section 4.1: problem formulation
   (b) Section 4.2: link prediction problems in heterogeneous networks
   (c) Section 4.3: a new proposed link prediction method (LPSF)
   (d) Section 4.4: two new proposed diffusion-based link predition models (LPDP and LPDM)
3. Chapter 5: conclusion

CHAPTER 3: LEARNING COLLECTIVE BEHAVIOR IN MULTI-RELATIONAL NETWORKS

Previously, Tang and Liu account for the possibility of connection heterogeneity by extracting social dimensions based on network connectivity [109, 110, 108]. In this dissertation, we present two alternative frameworks for solving this problem. One is to construct a low-dimensional social feature space by applying Fiedler embedding to an edge-based social feature space. The other one is a multi-label iterative relational neighbor classifier that incorporates a class propagation probability distribution obtained from instances' social features. We demonstrate that both proposed approaches are capable of identifying semantically similar users within the social network and outperform several benchmark methods in various real world datasets. In this section, I will present two proposed methods for learning collective behavior in multi-relational networks.

3.1 Problem Formulation

People's actions in social networks are often highly influenced by their connectors—friends, relatives, colleagues and neighbors. Homophily, the phenomena that like-minded individuals have an increased propensity to be connected [72], can directly lead to behavior correlations between linked individuals. As a result of homophily, we expect to observe more links between people with the same affiliation than those with different affiliations. This property facilitates the use of collective classification to identify networked instances as discussed in [116].

Here, we aim to predict the collective behavior in the multi-relational network by utilizing the behavior correlations embedded in the heterogeneous links. Essentially, we treat this problem as a multi-label classification problem, since each person in the network can belong to one or more affiliations. Assuming that each person can be associated with K affiliations, a binary class vector, $C = \{C_1, C_2, ..., C_K\}$, can be used to represent the user's involvement in each affiliation.

When $K = 1$ (each person only has a single affiliation), this problem is simplified to a binary classification problem.

We formulate our problem as follows: a society of N individuals are connected by the network graph $G = \{V, E, L\}$. In this graph, the set of nodes, $V = \{V_1, ..., V_N\}$, represent the social media contributors. The interactions between people are described by the edge set E. $L_i \subseteq L$ is the class label associated with node $V_i \in V$. The set of nodes, V, is further divided into two disjoint parts: X, the nodes for whom we know class labels (behavior categories), and Y, the nodes whose labels need to be determined. Our task is to determine the labels of the unknown nodes, Y, from the label set L, based on their interactions with other nodes (from X and Y) in the network.

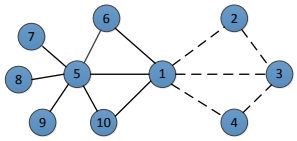
The problem we are addressing here is closely related to within-network classification [69, 67] and relational learning [38]. However, due to heterogeneity and noise in the social network, traditional relational inference methods have enjoyed only limited success on challenging datasets.

3.2 Edge-based Social Feature Extraction

Collective inference approaches, such as belief propagation [120] and iterative classification [77], have been widely adopted in relational classification. Those inferences mainly rely on the fact that the connections in the network are homogeneous: each node's connections are the outcome of one relationship. However, applying collective inference directly to raw social network data can produce erroneous results in cases where the links are predominantly heterogeneous. A simple example is shown in Figure 3.1, where two types of relationships co-exist within the same network. People who are members of the same sports club are connected by the solid line while those who attend the same conference are connected by the dashed line. Person 1 is associated with 2 affiliations since he participates in both activities. If all interactions are treated homogeneously, both affiliation labels will be propagated to all other people in the network. The question

now becomes how can we make the best use of affiliation correlations without being misled by by connection heterogeneity.

If we examine the social network, one important finding is that the links (connections) possess a strong correlation with the affiliation class. Connections in the social network are mainly affiliation-driven, and each connection in the social network can usually be regarded as principally resulting from one affiliation. Moreover, since each person usually has more than one connection, the involvements of potential groups related to one person's edges can be utilized as a representation for his true affiliations. Because this edge class information is not always readily available in the social media, an unsupervised clustering algorithm can be applied to partition the edges into disjoint sets such that each set represents one potential affiliation [109]. The edges of actors who are involved in multiple affiliations are likely to be separated into different sets which in turn facilitates the multi-label classification task.



(1) A simple example of a social network

Edges	Features (Node Index)									
	1	2	3	4	5	6	7	8	9	10
(1,3)	1	0	1	0	0	0	0	0	0	0
(1,5)	1	0	0	0	1	0	0	0	0	0
(2,3)	0	1	1	0	0	0	0	0	0	0
⋮	.....									

(2) Edge representation

Nodes	Features (Edge Cluster ID)	
	1	2
1	3	3
2	0	2
3	0	3
⋮	.....	

(3) Construct social dimensions

Figure 3.1: A simple example of a social network. The solid line represents being a member of the same sports club and the dashed line represents the activity of attending the same conference. In edge-based social features, each edge is first represented by a feature vector where nodes associated with the edge denote the features. For instance here the edge "1-3" is represented as [1,0,1,0,0,0,0,0,0,0]. Then, the node's social feature (SF) is constructed based on edge cluster IDs. Suppose in this example the edges are partitioned into two clusters (represented by the solid lines and dashed lines respectively), then the SFs for node 1 and 2 become [3,3] and [0,2] using the *count* aggregation operator.

Here we construct the initial social feature space using the scalable Edge-Clustering method proposed in [109]. Specifically, we first represent each edge in a feature-based format, where the nodes that define the edges denote the features as shown in Figure 3.1. Based on the features

of each edge, K-means clustering is used to separate the edges into groups. Each edge cluster represents a potential affiliation, and a person will be considered involved in one affiliation as long as any of his connections are assigned to that affiliation. Since the edge feature data is very sparse, the clustering process can be accelerated wisely. In each iteration a small portion of relevant instances (edges) that share features with cluster centroids are identified, and only the similarity of the centroids with their relevant instance need to be computed. By using this procedure introduced by [109], the clustering task can be completed within minutes even for networks with millions of nodes.

After clustering the edges, we can easily construct the node's social feature vector using *aggregation operators* such as *count* or *proportion* on edge cluster IDs. In [109], these *social dimensions* are constructed based on the existence of node's involvements in different edge clusters. Although aggregation operators are simply different ways of representing the same information (the histogram of edge cluster labels), alternate representations have been shown to impact classification accuracy based on the application domain [101].

3.3 Proposed Method: Extracting Social Dimensions using Fiedler Embedding

In social networks, people usually connect with each other for different causal reasons. Although some tools such as Facebook contain mechanisms for grouping and tagging links, many social media datasets do not possess this additional information. Treating these inherently heterogeneous connections in a homogeneous way can result in erroneous classification results. Tang and Liu account for the possibility of connection heterogeneity by extracting social dimensions based on network connectivity [109, 108]. In this section we present an alternative method for constructing a low-dimensional social feature space by applying Fiedler embedding to an edge-based social feature space.

3.3.1 Fiedler Embedding

Fiedler embedding was first proposed as a method for information retrieval and processing of document corpora [50]. It aims to map documents into a geometric space such that similar documents are close to each other. It has also been applied to the problem of video action recognition [62] to improve classification accuracy by combining two types of features. To the best of our knowledge, we are the first to apply Fiedler embedding to social network data. Here we use this technique to create a low-dimensional feature representation that encodes the relationship between nodes and edges implicitly contained within the edge cluster representation. The mathematical derivation of this embedding algorithm is as follows.

We begin with a graph $G = (V, E)$, where V is a set of vertices (nodes) and E is a set of edges represented by vertex pairs. An edge (V_i, V_j) connecting vertices V_i and V_j has a non-negative weight w_{ij} that describes the degree of similarity between two vertices. The more similar two vertices are to each other, the higher the corresponding weight. The vertices may represent several different classes of objects. For example, in the document case, the vertices are word terms and documents. Here, we assume that for any two vertices there is a connecting path.

The goal of Fiedler embedding is to project the vertices of the graph into a low-dimensional geometric space in such a way that similar vertices are close to each other even if they do not have a direct edge (observed relationship) between them. Following [50], this geometric embedding problem can be posed as a minimization problem. Specifically, we aim to find points in a k-dimensional space that minimize the weighted sum of squared edge lengths. If p_r and p_s are the locations of vertex r and s in the embedding space respectively, then our objective function can be written as follows:

$$\text{Minimize} \sum_{(r,s)\in E} W_{r,s}|p_r - p_s|^2 \tag{3.1}$$

Here, $W_{r,s}$ represents the weight of edge $E_{r,s}$. If the number of vertices is n, and the

geometric space has dimensionality k, then the positions of the vertices can be considered to be an $n \times k$ matrix X. The Laplacian matrix L can be defined as follows:

$$L(i,j) = \begin{cases} -w_{i,j} & \text{if } e_{ij} \in E \\ \sum_k w_{i,k} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

The Laplacian is the negative of the matrix of weights, except that the diagonal values are selected to make the row-sums zero. Thus L is symmetric and positive semi-definite. [50] reformulates the above minimization problem as:

$$\begin{aligned} & X = \arg\min_X \ \text{Trace}(X^T L X) \\ & \text{Subject to :} \\ & (i) X_i^T D 1^n = 0 \quad \text{for } i = 1, \ldots, k \\ & (ii) X^T D X = \Delta \end{aligned} \tag{3.3}$$

Here, two constraints are added. The first constraint is to make the median of the point set be the origin. 1^n denotes a vector of n ones. D is a positive diagonal weight matrix. The second constraint is to avoid the trivial solution of placing all the vertices at the origin. Δ denotes a non-negative diagonal matrix. The solution of this geometric optimization problem is $X = \tilde{Q}_k \Delta^{1/2}$, where $\tilde{Q}_k = [q_2, \ldots, q_{k+1}]$ are the generalized eigenvectors of $Ly = \lambda Dy$ sorted in non-decreasing order based on the corresponding eigenvalues λ_k. The solution is referred to as the Fiedler embedding.

In our case, two entities exist in the graph: social media users and edge cluster features. Fiedler embedding represents this bipartite graph in a low-dimensional space such that the geometric coordinates of similar nodes are closer to one other. From this point of view, Fiedler embedding is quite similar to the Co-Clustering method [27], which simultaneously clusters instances as fea-

tures. However, the Fiedler embedding is more powerful in that it allows one to capture a set of relationship between different types of entities in the graph. An additional benefit is that the embedded features are easy to update if a new vertex is added to the graph. The position of the new vertex v in the embedded space can be calculated by:

$$p_v = \frac{\Delta^{1/2}\tilde{Q}_k^T q}{||q||_1} \tag{3.4}$$

where q is the vector of similarity values for the new vertex.

3.3.2 Collective Behavior Classification Framework

In this section we describe our collective behavior classification framework using Fiedler embedding. In our work, we first extract the social features (SF) from the network topology using the Edge-Clustering method described in Section 3.2. These social features capture the nodes' involvement patterns in different potential affiliations. Once the features are computed, we apply the Fiedler embedding to discover the relationships among the nodes and edges by projecting them into a common Euclidean space. When applying Fiedler embedding, we first construct the Laplacian matrix from the whole dataset (both training and testing nodes), then the embedding space can be computed based on the k smallest eigenvalues (omitting the first one) of the Laplacian matrix. Finally, after mapping the social features into the embedding space, classification task can be performed on those embedded features. A detailed description of this collective behavior classification framework is provided in Table 3.1.

Table 3.1: Collective Behavior Classification Framework

Input: Network data, the labels of training nodes
Output: Labels of the unlabeled nodes

1. Construct social feature space A using scalable edge K-means
   - Convert the network into edge-centric view
   - Perform k-means clustering on edges
   - Create the social feature space based on the Edge-Clustering using the *proportion* aggregation operator
2. Construct the Laplacian matrix L as:
   - The user-feature similarity block is directly represented by the social feature space A.
   - The user-user similarity block is computed using Equation 3.5, and the feature-feature similarity block is calculated as the inner product of the columns of A.
3. Perform eigen-decomposition on the Laplacian matrix L according to $L = VDV^T$. V and D are the eigenvectors and eigenvalues of L in non-descending order.
4. Construct the k-dimensional embedded space $S = D_k^{1/2}V_k^T$. D_k and V_k are the k smallest eigenvalues and corresponding eigenvectors omitting the first one whose eigenvalue equals zero.
5. Compute the embedded feature of the entity via mapping the entity to the k-dimensional embedded space: $D_k^{1/2}V_k^T q/||q||_1$.
6. Train the classifier based on the embedded social features of the labeled nodes.
7. Use the classifier to predict the labels of the unlabeled nodes based on their embedded feature.

3.3.3 Construct Laplacian Matrix

The important part of computing the Fiedler embedding is constructing the Laplacian matrix L, which is a symmetric matrix constructed according to Equation 3.2. In our case, we have two entities: the users and the social features (SF). The Laplacian matrix forms a a 2×2 block structure:

$$\begin{pmatrix} D1 & A \\ A^T & D2 \end{pmatrix}$$

Here $D1$, $D2$ and A denote the similarity matrix of user-user, SF-SF and user-SF respectively. In principle, the similarity relations between entities can employ a variety of suitable measurements such as inner product between features or co-occurrence.

Since the Fiedler embedding aims to find a k-dimensional space where the relationship between different entities is captured such that semantically similar nodes, even those that do not share links, are nearby in this embedded space. In our application, the Fiedler embedding provides a way to discover users with similar behaviors in the social network, even if they have not had direct interactions with one another. For this purpose, we must select a suitable method to measure the similarity between different entities. Depending on the type of entity, different similarity measurements may be needed. For instance, the user-user similarity can be evaluated by the inner product of the social features, whereas the probability of occurrence is a much better measure for user-SF similarity.

Since values in node's social features represent the node's probability of occurrence in the edge clusters (potential affiliations), we directly use the node's social feature extracted using *proportion* aggregation operator as the measure for the similarity between users and potential affiliations. The user-user similarity can be measured using functions such as cosine similarity, inner product or Gaussian kernel. Additionally, since users are connected with each other in the social network, network topology can also be a measure for the latent relationship between people in the

network. In our case, it is reasonable to combine these two similarity measures to obtain a better results. If we denote W_f as the node affinity matrix obtained using social features and W_{adj} as the adjacency matrix of the social network, the combined similarity matrix can be obtained by the following schemes:

$$W_{\text{JOINT}} = W_f \circ W_{\text{adj}} \tag{3.5}$$

$$W_{\text{SUM}} = (1 - \alpha)W_f + \alpha W_{\text{adj}} \tag{3.6}$$

The first equation calculates the affinity matrix in the joint space. $W_f \circ W_{\text{adj}}$ denotes a per-entry (Hadamard) product of W_f and W_{adj}. The second equation defines the affinity matrix by adding W_f and W_{adj} together (α is a number between 0 and 1). The choice of the similarity function depends mainly on the data. Through our pilot experiments, we determined that W_{JOINT} performs better at clustering similar users and select as in our default user-user similarity measure. Since it is desirable to use the same measure to calculate SF-SF similarity, the method should be able to capture the relationship for both user-user and SF-SF. In our experiment, we observe that both the cosine distance and the Gaussian kernel can group the similar people together, but they were unable to group similar social features. However, the inner product method performs well at grouping both people and social features. Thus, we choose the inner product to measure the similarity for user-user and SF-SF.

3.3.4 Experimental Setup

3.3.4.1 Social Media Datasets

To facilitate direct comparison, we follow the evaluation protocol of [109]. Our proposed method is evaluated on real social network datasets extracted from two popular social media tools, BlogCatalog and YouTube. Both datasets are available from the Data Mining and Machine Learn-

ing Lab at http://www.public.asu.edu/~ltang9/social_dimension.html.

BlogCatalog: A blog in BlogCatalog can be associated with various pieces of information such as the categories the blog is listed under (e.g., "Music", "Education" and "Sports"), the blog subcategory tags (e.g., "Pop", "Science" and "Football") and the blog post level tags (e.g., "pop singers", "biology" and "top team"). Each blog category can be treated as a label that denotes the blogger's personal interest. Moreover, the blogger can specify his/her social connections with other bloggers. The blogCatalog dataset contains 39 categories and the average number of categories that each instance (blog) belongs to is 1.6.

YouTube is a popular website for sharing videos. Each user in YouTube can subscribe to different interest groups and add other users as his contacts. Here, we select a small subset of data (15000 nodes) from the original YouTube dataset in [109] using *snowball sampling*, and retain 47 interest groups as our class label. The details of the data set can be found in Table 3.2.

Table 3.2: Data Statistics

Data	BlogCatalog	YouTube
Categories	39	47
# of Nodes	10312	15000
# of Links	333,983	136,218
Network Density	6.3×10^{-3}	1.2×10^{-3}
Maximum Degree	3,992	14,999
Average Degree	65	9
Average Categories	1.6	2.1

3.3.4.2 Baseline Methods

Here we compare our proposed embedded social feature extraction method to five related methods: *EdgeCluster*, *wvRN*, *NodeCluster*, *Co-Clustering* and *Random*. A short description of these methods follows:

• *EdgeCluster* denotes the best performing method described in [109], where the social dimensions are directly extracted using the Edge-Clustering representation. The edge-based social features are constructed using the *proportion* operator, and a linear SVM is used for discriminant learning. We show the advantage of our embedded features by exploiting the relationship between different edge clusters.

• Weighted-vote Relational Neighbor Classifier (*wvRN*) [68] is a simple relational model that makes predictions based solely on the class labels of the related neighbors. The node's predicted class memberships are regarded as the weighted mean of the its neighbors. Though *wvRN* is a classifier which doesn't require learning, it performs surprisingly well on networked datasets.

• *NodeCluster* is based on [78] which assumes that each node is associated with only one affiliation. Here, we adopt the edge-based social dimensions as our raw social features because it allows one node to possess multiple affiliations. To verify this concept, we compare *EdgeCluster* with the *NodeCluster* method. For comparison, we first adopt k-means clustering to partition the network into disjoint groups, and then construct the social features using node clustering IDs as features. This comparison scheme is also examined in [109].

• *Co-Clustering* is a clustering method for partitioning the instances and their features simultaneously [27]. It was first proposed for clustering the document datasets represented as a bipartite graph between documents and their words. The k-means algorithm is applied on the top singular vectors of the scaled document-word matrix (omitting the principal singular vector). The main difference between Co-clustering and Fiedler embedding is that the Co-clustering algorithm does not take into account the similarity between different entities during its clustering procedure.

• *Random* method generates a class membership estimation randomly for each node in the network using neither network nor label information.

In our proposed method, Fiedler embedding is used to extracted the embedded social features. First, the Edge-Clustering method is adopted to construct the initial social dimensions. We use cosine similarity while performing the clustering; the dimensionality of the edge-based social features is set to 1000 for both BlogCatalog and YouTube dataset. Then, the embedded social features are extracted from the raw social dimensions using the procedure described in Section 3.2. Finally, a set of one-vs-all support vector machines (SVMs) are employed for classification.

Since our classification problem is essentially a multi-label task, during the prediction procedure, we assume that the number of labels for the unlabeled nodes is already known and assign the labels according to the top-ranking class. Such a scheme has been adopted for multi-label evaluation in social network datasets [109, 108]. In our work, we sample a small portion of nodes uniformly from the network as training instances. The fraction of the training data is from 5% to 30% for BlogCatalog dataset, and 1% to 10% for the YouTube datast. Two commonly used measures Micro-F1 and Macro-F1 are adopted to evaluate the classification performance.

3.3.4.3 Evaluation Measures

In this section, we explain the details of the evaluation criteria: Macro-F1 and Micro-F1 [32]. Given the dataset $X \in R^{N \times M}$, let $y_i, \hat{y}_i \in \{0,1\}^K$ be the true label set and the predicted label set for instance x_i.

• **Macro-F1** is the averaged F1 score over categories.

$$\text{Macro-F1} = \frac{1}{K} \sum_{k=1}^{K} F_1^k \tag{3.7}$$

For a category C_k, if P^k and R^k denote the precision and the recall respectively, the Macro-

F1 measure then is defined as the harmonic mean of precision and recall as follows:

$$F_1^k = \frac{2P^k R^k}{P^k + R^k} = \frac{2\sum_{i=1}^{N} y_i^k \hat{y}_i^k}{\sum_{i=1}^{N} y_i^k + \sum_{i=1}^{N} \hat{y}_i^k} \tag{3.8}$$

• **Micro-F1** is computed using F_1^k and while considering the precision as a whole. Specifically, it is defined as follows:

$$\text{Micro-F1} = \frac{2\sum_{k=1}^{k}\sum_{i=1}^{N} y_i^k \hat{y}_i^k}{\sum_{k=1}^{K}\sum_{i=1}^{N} y_i^k + \sum_{k=1}^{K}\sum_{i=1}^{N} \hat{y}_i^k} \tag{3.9}$$

According to the definition, Macro-F1 is more sensitive to the performance of rare categories while Micro-F1 is affected more by the major categories. In our experiments, we conduct 10 trials for each training condition, and report the average results in terms of Macro-F1 and Micro-F1.

3.3.5 Results

We perform three studies to evaluate the performance of the Fiedler embedding representation. First, we compare the classification results of our proposed framework against five baseline methods in BlogCatalog and YouTube dataset. We then study the connectivity between similar users in the embedded space. Finally a query experiment is conducted to examine the correlation between users and their connections.

3.3.5.1 Classification Study

Tables 3.3 and 3.4 list the classification performance of all methods in the BlogCatalog dataset and YouTube dataset, respectively. The best classification rates under each training condition are shown in bold. As we can see from the table, the methods based on edge-based social features; *EdgeCluster*, *EdgeCluster+Co-Clustering* and *EdgeCluster+Fiedler*, generally outperform

the other non-edge based methods. The baseline method *Random* only achieves around 4% for Macro-F1 while the other methods reach above 10% on the BlogCatalog dataset. *wvRN* method clearly outperforms *Random* method by utilizing correlation between linked nodes in the social networks for prediction. However, without differentiating the connections, *wvRN* shows poor performance when the links in network are heterogeneous. This is especially noticeable when we compare the results in YouTube dataset. By assuming each user is involved in one affiliation, the *NodeCluster* method performs worse than *EdgeCluster*.

Clearly, our proposed method *EdgeCluster+Fiedler* consistently outperforms *EdgeCluster* method in both BlogCatalog and YouTube datasets. By exploiting the correlations between different potential affiliations (edge clusters), Fiedler embedding shows its advantages in identifying the similar instances especially when the amount of training data is small. In the YouTube dataset, social features based on Fiedler embedding improves the classification results of *EdgeCluster* by 7% for Micro-F1 when only 1% of the instances are sampled as training data. Moreover, the higher results for Macro-F1 also demonstrate that the embedded social features has better performance in distinguishing different types of connections among the network. Fiedler embedding boosted the Macro-F1 results by around 2% and 5% in BlogCatalog and YouTube datasets respectively. Unfortunately, the comparison method, Co-Clustering, which is not able to capture the correlations between nodes and edge clusters, performs poorly in grouping multi-label instances and even undermines the performance of *EdgeCluster*.

Table 3.3: Classification Performance for BlogCatalog Dataset

Proportion of Labeled Nodes		5%	10%	15%	20%	25%	30%
Micro-F1(%)	*EdgeCLuster+Fiedler*	**23.72**	**24.25**	**26.15**	**26.85**	**27.43**	**27.45**
	EdgeCluster+Co-Clustering	18.55	19.92	21.30	21.94	22.34	22.67
	EdgeCluster	19.44	22.72	24.92	26.02	26.98	27.35
	wvRN	14.28	17.80	21.04	22.55	24.66	25.42
	NodeCluster	10.04	17.25	18.29	19.00	19.51	19.79
	Random	4.84	4.67	4.76	4.71	4.67	4.76
Macro-F1(%)	*EdgeCLuster+Fiedler*	**14.12**	**16.05**	**16.97**	**17.90**	**18.11**	**18.87**
	EdgeCluster+Co-Clustering	10.90	11.86	12.60	13.25	13.52	13.96
	EdgeCluster	12.12	14.63	15.89	17.30	17.75	18.55
	wvRN	8.59	10.54	12.29	12.99	14.20	14.48
	NodeCluster	9.47	11.08	12.06	12.74	13.31	13.66
	Random	4.07	3.97	4.04	4.01	3.97	4.07

Table 3.4: Classification Performance for YouTube Dataset

Proportion of Labeled Nodes		1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	*EdgeCLuster+Fiedler*	**34.11**	**34.80**	**35.58**	**36.28**	**37.08**	**37.94**	**38.02**	**38.10**	**38.16**	**38.21**
	EdgeCluster+Co-Clustering	23.00	25.40	26.86	27.60	28.92	29.09	29.11	30.53	31.19	31.52
	EdgeCluster	27.52	27.97	28.12	28.91	29.94	31.01	31.28	31.34	32.95	32.98
	wvRN	13.61	15.10	16.21	17.04	17.79	18.72	19.31	19.87	20.44	21.04
	NodeCluster	24.44	24.12	23.77	24.48	25.11	24.90	25.29	25.57	26.06	26.57
	Random	9.36	9.64	9.70	9.94	9.96	9.88	9.71	9.77	9.99	9.76
Macro-F1(%)	*EdgeCLuster+Fiedler*	**22.13**	**23.73**	**25.43**	**26.38**	**27.07**	**28.37**	**28.71**	**29.18**	**29.60**	**29.91**
	EdgeCluster+Co-Clustering	16.22	18.26	19.68	20.50	21.40	22.13	22.56	23.22	23.85	24.31
	EdgeCluster	17.22	19.10	20.45	21.33	22.30	23.29	23.50	23.93	24.93	25.03
	wvRN	11.38	13.10	14.51	15.41	16.34	17.41	17.99	18.75	19.18	19.66
	NodeCluster	19.52	20.15	20.93	21.65	22.38	23.17	23.32	23.78	24.10	24.49
	Random	8.69	8.98	9.03	9.27	9.18	9.32	9.13	9.14	9.29	9.08

3.3.5.2 *Connectivity Study*

Fiedler embedding is able to shorten the distance between similar instances in the embedded space even if they do not share any links. To verify this, we study the connectivity of the most similar users in the embedded space. We calculate the pairwise Euclidean distance between users based on their embedded social features and sort them in descending order. We then examine the probability of the most similar users being directly connected. As shown in Figure 3.2, the probability of being connected decreases steadily as the number of selected similar users increases. Only 60% of the most similar users are directly connected in the BlogCatalog dataset, and this probability is even lower in YouTube dataset, about 30%. This phenomenon suggests that the majority of users in close affinity do not share direct interactions in real social networks. People mainly maintain the activities with their current contacts, and neglect other potential friends in the network. From this point of view, Fiedler embedding also provides a way of recommending connections between similar users on social media websites.

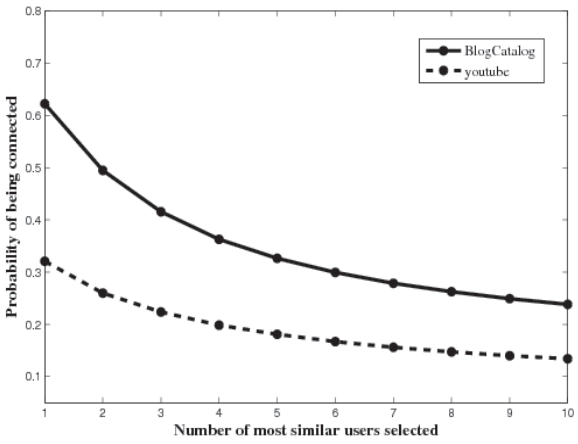


Figure 3.2: The majority of the most similar users are not connected in the embedded space. Fiedler embedding is able to explore the semantically similar users in the embedded space even if they are not connected with each other.

Table 3.5: Cluster Disagreement Scores

	Embedded space	Network structure
BlogCatalog	5.015	5.568
YouTube	2.245	3.155

3.3.5.3 Cluster Experiment

In this section, we explore the Fiedler embedding's capability to group the semantically similar entities (e.g., users and connections) together in the embedded space. Specifically, we query each edge cluster in the embedded space for its most similar users, and evaluate the overall grouping quality by the average disagreement score of the queries. The disagreement score is defined as the entropy of the selected users' labels according to the class distribution. Therefore, the more diverse the labels of the selected users, the greater the disagreement about the edge cluster's query.

We compare the query results using embedded edge cluster features with the one obtained directly from the network structure, the output of the edge cluster query and its associated nodes. Table 3.5 shows the cluster disagreement results for the BlogCatalog dataset and YouTube dataset respectively. In both datasets, Fiedler embedding demonstrates better performance in grouping similar entities together as shown by the lower disagreement scores. When the connections in the network are heterogeneous, directly relying on the link structure to detect similar instances could yield unsatisfactory results. Instead, the Fiedler embedding discovers the semantically similar entities, which are normally disconnected in the network, in the constructed embedded space that retains their relationships.

3.4 Proposed Method: Relational Neighbor Classification using Social Context Features

Relational learning [38] can learn models of this data structure by utilizing the correlation between labels of linked objects; networks resulting from social processes often possess a high amount of homophily, such that nodes with similar labels are more likely to be connected [72]. Many of the algorithms developed for relational classification are heuristic methods that do not necessarily correspond to formal probabilistic semantics [77]. In other approaches, during the inference process the probability distribution is structured as a graphical model based on the assumption that the structure of the network corresponds at least partially to the structure of the network of probabilistic dependencies [69]. Relational learning enhances the tractability of estimating the full joint probability distribution of the data by making a first-order Markov assumption that the label of one node is dependent on that of its immediate neighbors in the graph. Collective inference in relational classification makes simultaneous statistical estimations of the unknown labels for interrelated entities, and finds a equilibrium state such that the inconsistency between neighboring nodes is minimized. By exploiting network connectivity information, relational classification models have been shown to outperform traditional classifiers [80, 112].

The heterogeneous link structure in multi-relational networks limits the performance of conventional relational learning. Here, we present a multi-label relational classifier that accounts for this inhomogeneity in connections and is designed for classification problems on multi-label networked datasets. Our proposed method, *SCRN*, extends *RN* by introducing a node class-propagation probability that modulates the amount of propagation that occurs in a class specific way based on the node's similarity to each class. Although the class propagation probability can be determined by the node's intrinsic features, it can also be based on node's social features. These features capture link patterns between a node and its neighbors and can be extracted from network topology in instances when the node lacks intrinsic features. *SCRN*'s ability to differentiate between classes during the inference procedure allows it to outperform previous methods in a

real-world multi-label relational dataset and a set of synthetically generated sets designed with challenging network parameters.

In multi-label classification problems, a popular approach is to decompose the multi-label classification problem into multiple binary classification problems (one for each class). Conventional multi-label classification approaches (e.g., the ones used on non-networked data), usually assume the instances are i.i.d., and that the inference for each instance is performed independently:

$$P(L|V) \propto \prod_{v_i \in V^U} P(L_i|v_i). \tag{3.10}$$

We propose a multi-label relational classifier that models the correlations between interrelated instances in the network. We start by constructing a social feature space, an edge-based representation of social dimensions using the network topology to capture the node's potential affiliations as described in [109]. A class-propagation probability is assigned to each node to describe its intrinsic correlation to each class. The class-propagation probability is calculated from the similarity between the node's social features and the class reference vector. The multi-label relational classifier estimates a node's label set based on its neighbors' class labels, the similarity between connected nodes, and its class propagation probability. *SCRN* iteratively classifies the labels of the unlabeled nodes until all the label predictions are fixed or the maximum number of iterations is reached. In the next section, we describe the basic idea behind relational neighbor classifiers before describing our proposed method.

3.4.1 Proposed method: SCRN

The Relational Neighbor (*RN*) classifier proposed by [68], is a simple relational probabilistic model that makes predictions for a given node based solely on the class labels of its neighbors, without machine learning or additional features. RN estimates class-membership probabilities by assuming the existence of homophily in the dataset, entities connected to each other are similar

and likely belong to the same class. Suppose each instance in the network only belongs to a single class $c \in C$. Given $v_i \in V^U$, the relational-neighbor classifier estimates $P(L_i = c|v_i)$, the class-membership probability of a node v_i belonging to class c, as the (weighted) proportion of nodes in the neighborhood that belong to the same class. We define neighbors N_i as the set of *labeled* nodes that are linked to v_i. Thus:

$$P(L_i = c|v_i) = \frac{1}{Z} \sum_{v_j \in N_i} w(v_i, v_j) \times I(L_j = c), \tag{3.11}$$

where $Z = \sum_{v_j \in N_i} w(v_i, v_j)$. $w(v_i, v_j)$ is the weight of the link between node v_i and v_j and $I(.)$ is an indicator variable.

Instead of making a hard labeling during the inference procedure, the weighted-vote relational neighbor classifier (*wvRN)* extends *RN* by tracking changes in the class membership probabilities. *wvRN* estimates $P(L_i|v_i)$ as the (weighted) mean of the class membership probabilities of the entities in the neighborhood (N_i):

$$P(L_i = c|v_i) = \frac{1}{Z} \sum_{v_j \in N_i} w(v_i, v_j) \times P(L_j = c|N_j), \tag{3.12}$$

where Z is the usual normalization factor.

In both *RN* and *wvRN*, entities whose class labels are unknown are either ignored or are assigned a prior probability, depending on the choice of the local classifier. Since only a small portion of the nodes in G have known labels, a collective inference procedure is needed to propagate the label information through the network to related instances, using either the *RN* classifier or *wvRN* classifier in its inner loops.

To address this problem, instead of uniformly aggregating the neighbor's labels along each class like *wvRN* does, we propose to assign each node a class propagation probability distribution, which represents its likelihood of maintaining the neighbor's class label set. A node will be more

likely to share a class with neighbors that have a high class-propagation probability. Take the toy graph for example, when inferring the labels of node 2 from node 1, we want to keep its estimation of class 2's probability much higher than class 1 to make a more accurate prediction. Therefore, a node's class-propagation probability can be regarded as its prior probability for each class. Learning the class-propagation probability distribution is critical in order to achieve better discrimination during the inference procedure. Fortunately the structure of the network can be highly informative, and we capture this information through using the network topology to construct *social features*.

In the proposed method, we first extract the social features (SF) from the network topology using the Edge-Clustering method described in Section 3.2. These social features capture the nodes' involvement patterns in different potential affiliations, and the node's class propagation probability can be constructed from the social features in the following way. An initial set of reference features for class c can be defined as the weighted sum of social feature vectors for nodes known to be in class c:

$$RV(c) = \frac{1}{|V_c^K|} \sum_{v_i \in V_c^K} P(l_i^c = 1) \times SF(v_i) \tag{3.13}$$

where $V_c^K = \{v_i | v_i \in V^K\}$, which represents the nodes whose labels are known as class c.

Then node v_i's class propagation probability for class c conditioned on its social features, $P_{CP}(l_i^c | SF(v_i))$, can be calculated by the normalized vector similarity between $SF(v_i)$ and class c's reference feature vector, $RV(c)$:

$$P_{CP}(l_i^c | SF(v_i)) = \text{sim}(SF(v_i), RV(c)), \tag{3.14}$$

where $\text{sim}(a, b)$ is any normalized vector similarity function (e.g., cosine or inner product).

Our proposed multi-label relational classifier then estimates the class-membership proba-

bility of node v_i belonging to class c:

$$P(l_i^c|N_i, SF(v_i)),$$

based on the class labels of its neighbors, $\{L_j|v_j \in N_i\}$, the weight between v_i and its directed neighbors v_j, $w(v_i, v_j)$, and its conditional class propagation probability, $P_{CP}(l_i^c|SF(v_i))$. The multi-label relational classifier model is defined as follows:

$$\begin{aligned} P(l_i^c|N_i, SF(v_i)) &= \frac{1}{Z}\sum_{v_j \in N_i} P_{CP}(l_i^c|SF(v_i)) \\ &\quad \times w(v_i, v_j) \times P(l_j^c|N_j), \end{aligned} \tag{3.15}$$

where Z is the normalization factor. Similar to the *RN* and *wvRN* classifiers, our multi-label relational classifier iteratively classifies the nodes in V^U using the model defined in Equation 3.15 in its inner loop. Since the label predictions change in each iteration, the class reference feature vector is updated based on the feature vectors of nodes (both training and testing nodes) whose labels belong to class c in the current iteration. Here we adopt the Relaxation Labeling (RL) approach [125, 20] in the collective inference framework. During each iteration, RL updates the prediction probability by taking account of probability estimations in the last iteration. The general update procedure for relaxation labeling is shown in Equation 3.16 [69]:

$$P_i^{(t+1)} = \beta^{(t+1)} \cdot \mathcal{M}_{\mathcal{R}}(v_i^{(t)}) + (1 - \beta^{(t+1)}) \cdot P_i^{(t)}, \tag{3.16}$$

where $\beta^{(0)} = k$ and $\beta^{(t+1)} = \beta^{(t)} \cdot \alpha$. Both k and α are constants that fall in the range 0 and 1. t is the iteration count and $\mathcal{M}_{\mathcal{R}}(\cdot)$ denotes the relational model. The inference procedure in *SCRN* terminates when it meets the stopping criteria; possible stopping criteria include the stability of all label predictions between iterations or reaching a fixed budget of iterations. A summary of the *SCRN* framework is shown in Table 3.6.

Table 3.6: Overview of *SCRN* Algorithm

Input: $\{G, V, E, C, L_K\}$, *Max_Iter*
Output: L_U for nodes in V^U

1. Construct the social feature space using scalable K-means Edge-Clustering.
2. Initialize the class reference vectors, RV, for each class based on Equation 3.13.
3. Calculate the class-propagation probability for each test node using the similarity between the node's social feature and class reference vectors using the *GHI* kernel.
4. Repeat until # iterations>*Max_Iter* or predictions converge to stable values:
   - Estimate the test nodes class membership probability according to Equation 3.15.
   - Update the test node class membership probability based on the prediction in the last iteration according to Equation 3.16.
   - Update the class reference vectors according to the labels of the nodes in current iteration.
   - Re-calculate each node's class-propagation probability using the present class reference vectors.

By using the scalable Edge-Clustering method proposed in [109], we construct the node's social feature space. Figure 3.3 shows a result of the Edge-Clustering method on a small sample of DBLP dataset. Edges are clustered into 10 separate groups, and each edge group is marked in one color. As we can see, the Edge-Clustering method is able to maintain the correlation between connected nodes; nodes and their neighbors usually share the same type of edge. Also, nodes with high degree are more likely to associate with different types of edges since they are usually involved in multiple affiliations.

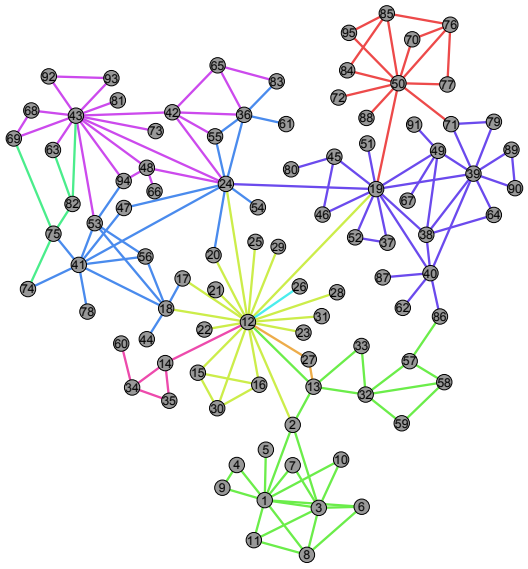


Figure 3.3: Visualization of Edge-Clustering using subset of DBLP with 95 instances. Edges are clustered into 10 groups, with each shown in a different color.

3.4.2 Experimental Setup

To evaluate the classification performance of our proposed multi-label relational classifier, we apply our model on three real-world multi-label relational datasets, DBLP, IMDb and YouTube, whose properties are summarized in Table 3.7.

Table 3.7: Dataset Summary

Data	**DBLP**	**IMDb**	**YouTube**
# of nodes	8,865	11,476	15,000
# of links	12,989	323,892	136,218
# of categories	15	27	47
Network Density	3.3×10^{-4}	4.7×10^{-3}	1.2×10^{-3}
Maximum Degree	86	290	14,999
Average Degree	3	55	9
Average Category	2.3	1.5	2.1

3.4.2.1 DBLP Dataset

The first real-world dataset we studied in this dissertation is extracted from the DBLP dataset.[1] The DBLP dataset provides bibliographic information for millions of computer science references. Here we construct a weighted collaboration network for authors who have at least published 2 papers during the 2000 to 2010 timeframe. In this network, the author is represented by the node, and two authors are linked together if they have collaborated at least twice. The weight of the link is defined as the number of times two authors have co-authored papers. Each author can have multiple research interests. For our dataset, we selected 15 representative conferences in 6 research areas. A author is interested in a research area if he/she has published a paper in any conferences listed under that area, and our classification task is to associate each author with a set of conferences he is interested in. The selected conferences are listed below:

- Database: ICDE, VLDB, PODS, EDBT
- Data Mining: KDD, ICDM, SDM, PAKDD
- Artificial Intelligence: IJCAI, AAAI
- Information Retrieval: SIGIR, ECIR

[1] `http://www.informatik.uni-trier.de/~ley/db/`

- Computer Vision: CVPR
- Machine Learning: ICML, ECML

3.4.2.2 IMDb Dataset

The second dataset studied in this research is the IMDb dataset.[2] The Internet Movie Database (IMDb) is an online database of information related to movies, television programs, and video games, including information about directors, actors, plots of a movie, etc. In this dissertation our classification task focuses on predicting the movie's genres in the movie network extracted from IMDb. Each movie can be assigned to a subset of 27 different candidate movie genres in the database such as "Drama", "Comedy", "Documentary" and "Action". In our experiment, we extract movies and TV shows released between 2000 and 2010, and those directed by the same director are linked together. We only retain movie and TV programs with more than 5 links.

3.4.2.3 YouTube Dataset

This dissertation also presents results on networks extracted from YouTube. YouTube is a popular website for sharing videos. Each user in YouTube can subscribe to different interest groups and add other users as his/her contacts. Here we select a small subset of data (15000 nodes) from the original YouTube dataset in [109] using *snowball sampling*, and retain 47 interest groups as our class label. The dataset is available from the Data Mining and Machine Learning Lab website.[3]

3.4.2.4 Baseline Methods

In this dissertation, we compare our proposed multi-label relational classifier to four related methods: *EdgeCluster*, *wvRN*, *Prior* and *Random*. A short description of these methods follows:

[2] `http://www.imdb.com/interfaces`

[3] `http://www.public.asu.edu/~ltang9/social_dimension.html`

• *EdgeCluster* captures the node's correlation to different classes by extracting social dimensions from network structure using the Edge-Clustering representation [109]. The edge-based social features are constructed using the *count* operator on the edge cluster IDs, and a linear SVM is used as the classifier.

• *wvRN*, weighted-vote Relational Neighbor Classifier [68], makes predictions based solely on the class labels of the given node's linked neighbors; the node's predicted class memberships are constructed as the weighted mean of its neighbors. To compare fairly with *SCRN*, we also include the relaxation labeling method in the collective inference procedure in *wvRN*. The results show that our proposed *SCRN* improves on the performance of *wvRN* by refining the class propagation model.

• *Prior* generates a class membership estimate according to the fraction of instances in the labeled training data with the given class label. Thus, all nodes (regardless of network connectivity) share the same class estimates which are assigned to multi-label nodes in rank order.

• *Random* generates class membership estimates randomly for each node in the network using neither network nor label information.

In our proposed method, the Edge-Clustering method is initially adopted to construct the social features. We use cosine similarity while performing the clustering; the dimensionality of the edge-based social features is set to 1000 for DBLP and YouTube datasets and 10000 for IMDb dataset. The selection of the dimensionality is based on the one which offers the best performance of *EdgeCluster* method. In *SCRN*, the class-propagation probability is calculated by the similarity between the node's social feature and class reference features. We evaluated several similarity measures, including *Cosine*, *Inner Product* and *Generalized Histogram Intersection Kernel*, and we observe that the Generalized Histogram Intersection Kernel (GHI) [16] outperforms the other measures in grouping similar instances. Therefore, we adopt GHI in our *SCRN* classifier.

Since our problem is essentially a multi-label classification task, we assume that the number of labels for the unlabeled nodes is already known and assign the labels according to the top-

ranking set of classes at the conclusion of the inference process. Such a scheme has been adopted for multi-label evaluation in social network datasets [109, 117]. In our work, we sample a small portion of nodes uniformly from the network as training instances. The fraction of the training data ranges from 5% to 30% for DBLP dataset, 1% to 20% for IMDb dataset, and 1% to 9% for YouTube dataset. Here we adopted the "network cross-validation" (NCV) method proposed in [76] to reduces the overlap between test samples and make more precise comparison evaluation between different classification approaches. NCV method start with creating k disjoint test sets. Then for each test set fold, the remaining folds are merged together, and the training set is randomly sampled from the merged set. The collective inference will be executed over the full set of unlabeled nodes (the inference set), but model performance will only be evaluated on the nodes assigned to the test set. The classification performance is evaluated on three standard measures: Macro-F1, Micro-F1, Hamming Loss. Table 3.8 outlines the NCV procedure [76].

Table 3.8: Network cross-validation procedure

Input: *G*, *propLabeled*, *k*,
S = total number of instances in *G*
F = ∅
Split data into *k* disjoint folds
for *fold* 1 to *k*
  current *fold* becomes *testSet*
  remaining *fold* are merged and become *trainPool*
  trainSet = uniform random sample of (*propLabeled* × *S*) nodes from *trainPool*
  inferenceSet = *G* - *trainSet*
  $F = F \bigcup <trainSet, testSet, inferenceSet>$
end for
output: F

3.4.2.5 Evaluation Measures

For this experiment, we choose three evaluation criteria: Macro-F1, Micro-F1 and Hamming Loss. The first two are explained in Section 3.3. *Hamming Loss* [128] is one of the most frequently used criteria, which counts the number of labels whose relevance is incorrectly predicted.

$$\text{HammingLoss} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}||y_i \otimes \hat{y}_i||_1. \tag{3.17}$$

where $\otimes$ stands for the Hamming distance of two sets (XOR operation), and $||\cdot||_1$ denotes the $l1$ -norm. The smaller the value, the better the performance of the classifier.

3.4.3 Results

We perform two studies to evaluate the performance of multi-label relational classifier. First, we study the performance of *SCRN* under different measures of calculating the node similarity, $w(v_i, v_j)$. Then we compare the classification results of *SCRN* against four baseline methods on the DBLP, YouTube and IMDb datasets.

3.4.3.1 Node Similarity Measures

Both the *wvRN* and *SCRN* classifier consider the similarity of linked nodes, $w(v_i, v_j)$, when estimating the label of node v_i. $w(v_i, v_j)$ measures the similarity between linked nodes; note that the weight matrix W is not necessarily symmetric (i.e., $w(v_i, v_j)$ can be different from $w(v_j, v_i)$). In this experiment we compare three different approaches for determining the node similarity using the information contained in the network structure.

• *Degree* calculates the weight $w(v_i, v_j)$ by the normalized fraction of connections between v_i and v_j among all of v_i's connections. In our weighted DBLP dataset, we normalize the original weight of the link, $w_0(v_i, v_j)$, by the total weight summed over the neighbors of node v_i, $\sum_{j\in N_i} w_0(v_i, v_j)$, and use it as an estimate of the node's similarity to v_j.

Table 3.9: *SCRN* results using different node similarity measures on DBLP (10% training data)

	Degree	**Cosine**	**Pearson**
Micro-F1 (%)	**56.51**	42.96	54.39
Macro-F1 (%)	**49.35**	36.99	47.33

• *Cosine Similarity* uses the cosine function to normalize the number of common neighbors between two nodes in the graph.

• *Pearson Correlation Coefficient* is an alternative way to normalize the count of common neighbors by comparing it with the expected value that the count would have in a network in which nodes select their neighbors at random [84].

Table 3.9 shows the performance of *SCRN* using different node similarity measures. The *Degree* similarity measure clearly achieves the highest accuracy rate (Macro-F1 score of 49.35%); the *Pearson Correlation Coefficient* performs slightly worse than *Degree* by around 2%; and *Cosine Similarity* is poorest at capturing the relationship between two nodes. Based on this experiment, we select the *Degree* method to measure node similarity for the remaining experiments.

3.4.3.2 Classification Results

Table 3.10 shows the classification performance under Macro-F1 and Micro-F1 of our method vs. the related methods on the DBLP dataset averaged over 10 cross-validation folds. We make several observations. First, we confirm that the relational approaches, which consider the correlations between linked nodes (*SCRN*, *EdgeCluster* and *wvRN*) all outperform the two baseline methods, *Random* and *Prior*. *wvRN*, which takes advantage of the correlation between the labels of linked nodes significantly outperforms the baselines. *EdgeCluster*, which aims to differentiate different types of connections in the network by extracting social dimension from network

topology, performs worse than *wvRN* in this case. This might because the links in DBLP dataset are explicitly driven by common conference interests (labels) of collaborating authors. In this scenario, by directly making use of the dependencies between the labels, *wvRN* offers better classification results than *EdgeCluster*. Our proposed method *SCRN* consistently outperforms all of the other methods in all training conditions. The class-propagation probability in *SCRN* captures the node's intrinsic likelihood of belonging to each class, and provides more accurate estimation in the inference procedure. Under all training conditions, *SCRN* demonstrates a higher accuracy improvement over *wvRN*. *SCRN* boosts the results by 3% for Macro-F1 and 4% for Micro-F1.

Table 3.11 shows the result on the IMDb dataset. We observe that *SCRN*'s performance is consistently superior than other baseline methods on Micro-F1 and most cases on Macro-F1. The baseline methods *Prior* and *Random* achieve only 5% and 6% for Macro-F1 respectively while other methods reach more than 20%. We further evaluate the classification performance of *SCRN*, *wvRN* and *EdgeCluster* using the *Hamming Loss* measure. These results, shown in Figure 3.4 and Figure 3.5, respectively, demonstrate that *SCRN* achieves better performance over *wvRN*, and *EdgeCluster* under all cases. *SCRN* decreases the value of *Hamming Loss* by 2.6% on DBLP (5% training samples) and 1.4% on IMDb (1% training samples).

We observe that *EdgeCluster* performs surprisingly well on the YouTube dataset as shown in 3.12. *SCRN* performs worse than *wvRN* and *EdgeCluster* for Macro-F1, but shows slightly better performance than *EdgeCluster* methods for Micro-F1 in most conditions. *SCRN*, *EdgeCluster* and *wvRN* perform quite closely on *Hamming Loss* as shown in Figure 3.6. As has been discussed in [110], by differentiating heterogeneous connections, *SocioDim* achieves better performance than collective inference when a network has high heterogeneity and there is only a small amount of labeled data. Compared to the DBLP and IMDb datasets, YouTube has a much larger number of classes (47 categories), and the link structure is more heterogenous, and much less informative. The high link heterogeneity reduces the correlations between linked nodes, which directly limits performance of relational classifiers, such as *SCRN* and *wvRN*.

Table 3.10: Classification Performance for DBLP Dataset (Macro-F1 and Micro-F1)

Labeled	5%	10%	15%	20%	25%	30%
Micro-F1 (%)						
SCRN	**51.06±1.08**	**56.51±1.18**	**60.31±0.70**	**62.80±0.84**	**65.03±1.13**	**66.58±0.95**
Edge	38.41±2.39	43.65±1.69	47.53±2.54	50.29±1.55	52.50±2.42	54.00±1.26
wvRN	47.59±1.22	52.78±1.29	56.67±0.87	59.45±0.59	61.51±1.36	63.24±0.97
Prior	32.22±1.48	33.06±1.60	32.43±1.85	33.45±1.22	33.23±0.94	33.50±1.04
Random	20.32±0.75	20.65±0.83	20.59±0.97	20.06±0.96	19.84±1.05	20.40±0.78
Macro-F1 (%)						
SCRN	**44.35±1.45**	**49.35±1.51**	**53.65±1.37**	**56.38±1.26**	**58.62±1.21**	**59.54±0.94**
Edge	33.83±1.62	40.26±2.03	43.47±1.14	46.12±1.18	47.36±2.11	49.29±0.98
wvRN	41.85±1.48	46.61±1.49	51.05±1.53	53.88±1.18	55.83±1.70	57.08±1.51
Prior	15.31±1.11	15.76±1.52	15.42±1.41	16.06±0.97	16.13±0.72	16.48±0.81
Random	18.66±0.74	18.97±0.70	18.97±0.91	18.42±0.94	18.20±0.98	18.69±0.67

We confirm that in most of the multi-labeled collaborative networks, such as DBLP and IMDb, the correlation between connected nodes can be a great asset for relational learning. In this dissertation, we proposed a new relational classifier that combine social features. The way to combine the social features can be tricky and will directly affect the performance of the classifier. In [108], they implemented a variant of link-based classifier with relaxation labeling that is based on the combination of the labeled nodes' social features and relational features aggregated from their neighbors. Unfortunately they found the collective inference reduced the performance of social features. Here, rather than simply concatenate two types of features, we compare the similarity between node's social features, and translate it into a class propagation probability that boosts the performance of the collective classifier.

Table 3.11: Classification Performance for IMDb Dataset (Macro-F1 and Micro-F1)

Labeled	1%	3%	5%	10%	15%	20%
Micro-F1 (%)						
SCRN	**45.62±2.03**	**58.58±1.39**	**63.65±1.07**	**68.90±1.69**	**71.01±0.70**	**71.98±1.24**
Edge	40.08±1.51	52.17±0.85	57.31±1.56	62.03±1.89	64.50±0.85	65.27±1.32
wvRN	44.72±1.91	56.98±1.35	62.44±1.18	67.05±1.89	70.76±0.92	71.78±1.36
Prior	39.67±1.49	39.48±1.20	39.37±1.23	39.27±1.11	39.28±1.30	39.20±1.14
Random	7.58±0.61	7.23±0.72	7.77±0.60	7.43±0.99	7.38±0.85	7.75±0.59
Macro-F1 (%)						
SCRN	18.46±2.35	27.19±2.31	**33.22±1.39**	**39.40±3.10**	**42.67±2.57**	**43.31±1.66**
Edge	17.64±1.59	24.24±1.83	29.66±2.13	34.17±2.45	36.61±1.59	37.50±1.54
wvRN	**18.53±2.28**	**27.41±2.06**	33.02±1.76	39.08±2.90	42.10±2.54	43.28±2.11
Prior	5.58±0.49	5.57±0.52	5.49±0.46	5.43±0.41	5.40±0.40	5.34±0.42
Random	6.22±0.53	6.04±0.67	6.40±0.61	6.21±0.94	6.24±0.62	6.31±0.58

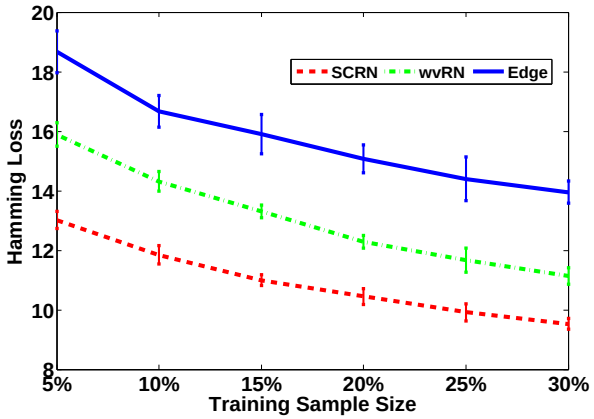


Figure 3.4: Classification Performance for DBLP Dataset (Hamming Loss), lower score shows better performance.

Table 3.12: Classification Performance for YouTube Dataset (Macro-F1 and Micro-F1)

Labeled	1%	3%	5%	7%	9%
Micro-F1 (%)					
SCRN	**35.67±3.54**	40.69±3.35	**43.15±1.35**	**43.76±3.11**	43.93±3.27
Edge	35.44±3.89	**40.92±3.87**	41.76±2.60	43.20±3.88	**44.09±2.89**
wvRN	33.18±5.39	40.08±4.23	42.57±2.56	43.40±3.45	43.87±3.90
Prior	34.32±2.74	37.21±1.94	37.24±2.22	37.83±2.38	37.54±2.04
Random	9.77±2.92	9.91±2.56	9.05±2.92	9.52±2.37	9.66±2.69
Macro-F1 (%)					
SCRN	15.20±4.51	21.43±4.98	23.93±3.75	25.84±4.90	26.00±4.28
Edge	**21.64±3.33**	**25.46±4.23**	**26.73±3.67**	**30.08±3.76**	**30.65±4.08**
wvRN	14.80±4.40	23.53±4.99	24.26±4.03	26.54±4.82	27.57±4.27
Prior	10.58±4.80	11.10±4.64	10.78±4.65	11.04±4.53	11.10±4.42
Random	9.07±3.22	9.08±2.70	8.24±3.06	8.65±2.55	8.78±2.91

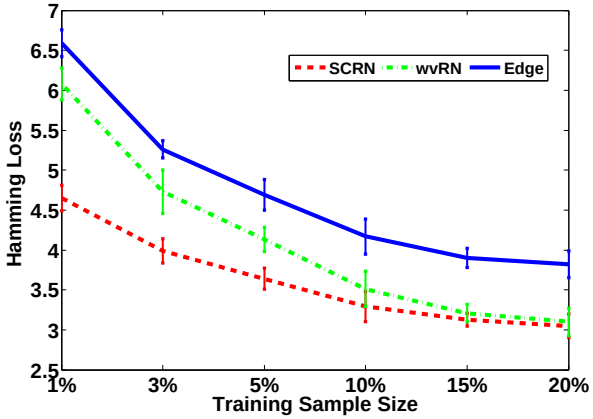


Figure 3.5: Classification Performance for IMDB Dataset (Hamming Loss), lower score shows better performance.

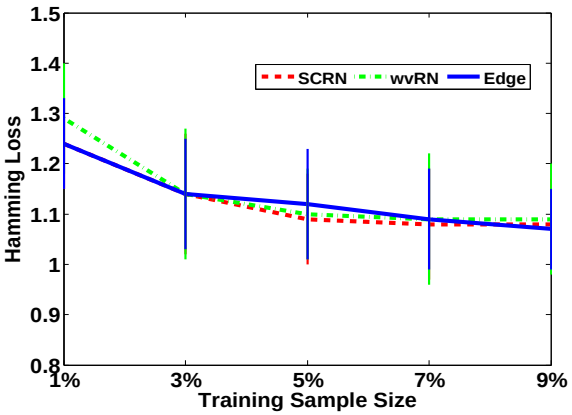


Figure 3.6: Classification Performance for YouTube Dataset (Hamming Loss), lower score shows better performance.

CHAPTER 4: LINK PREDICTION IN MULTI-RELATIONAL NETWORKS

4.1 Problem Formulation

In many social media tools, link prediction is used to detect the existence of unacknowledged linkages in order to relieve the users of the onerous chore of populating their personal networks. The problem can be broadly formulated as follows: given a disjoint node pair (x, y), predict if the node pair has a relationship, or in the case of dynamic interactions, will form one in the near future [121]. Often, the value of the participant's experience is proportional to the size of their personal network so bootstrapping the creation of social networks with link prediction can lead to increased user adoption. Conversely, poor link prediction can irritate users and detract from their initial formative experiences.

Although in some cases link predictors leverage external information from the user's profile or other documents, the most popular link predictors focus on modeling the network using features intrinsic to the network itself, and measure the likelihood of connection by checking the proximity in the network [52, 99]. Generally, the similarity between node pairs can be directly measured by neighborhood methods such as the number of shared neighbors [83] or subtly measured by path methods [66].

One weakness with network-based link prediction techniques is that the links are often treated as having a homogeneous semantic meaning, when in reality the underlying relationship represented by a given link could have been engendered by different causal factors. In some cases, these causal factors are easily deduced using user-supplied meta-information such as tags or circles, but in other cases the provenance of the link is not readily apparent. In particular, the meaning of links created from overlapping communities are difficult to interpret, necessitating the development of heterogeneous link prediction techniques.

In the familiar example of scientific collaboration networks, authors usually have multiple research interests and seek to collaborate with different sets of co-authors for specific research areas. For instance, Author A cooperates with author B on publishing papers in machine learning conferences whereas his/her interaction with author C is mainly due to shared work in parallel computation. The heterogeneity in connection causality makes the problem of predicting whether a link exists between authors B and C more complicated. Additionally, Author A might collaborate with author D on data mining; since data mining is an academic discipline closely related to machine learning, there is overlap between the two research communities which indicates that the linkage between B and D is more likely than a connection between B and C. In this dissertation, we detect and leverage the structure of overlapping communities toward this problem of link prediction in networks with multiple distinct types of relationships.

Community detection utilizes the notion of "structural equivalence" which refers to the property that two actors are similar to one another if they participate in equivalent relationships [85]. Inspired by the connection between structural equivalence and community detection, Soundarajan and Hopcroft proposed a link prediction model for non-overlapping communities; they showed that including community information can improve the accuracy of similarity-based link prediction methods [102]. Since community information is not always readily available, community detection techniques can be applied to partition the network into separate groups [3]. In this dissertation, we present a new link prediction framework for networks with overlapping communities that accounts for the hidden community information embedded in a set of heterogeneous connections.

When a person's true affiliations are unknown, our proposed method, *LPSF*, models link heterogeneity by adding weights to the links to express the similarities between node pairs based on their social features. These social features are calculated from the network topology using Edge-Clustering [109] and implicitly encode the diversity of the nodes' involvements in potential affiliations. The weights calculated from the social features provide valuable information about the true closeness of connected people, and can also be leveraged to predict the existence of the

unobserved connections. In this dissertation, we proposed a new link prediction framework *LPSF* that captures nodes' intrinsic interaction patterns from network topology and embeds the similarities between connected nodes as link weights. The nodes' similarity is calculated based on social features extracted using Edge-Clustering to detect overlapping communities in the network. Experiments on the DBLP collaboration network demonstrate that the judicious choice of weight measure in conjunction with supervised link prediction enables us to significantly outperform existing methods. Our proposed method is better able to capture the true proximity between node pairs based on link group information and improve the performance of supervised link prediction methods.

The strength of our approach is that it extracts communities in an unsupervised way, and thus can be used to study informal patterns of contact between researchers. These informal patterns, described as the "invisible college" in bibliometric research, can be a powerful but difficult to quantify force behind the process of scientific collaboration [134]. The proposed method is very practical: it can be employed on any unweighted or weighted network in conjunction with any existing link prediction classifier. Moreover, the social features are themselves complementary to node-based approaches.

4.2 Problems of Heterogeneity

Unsupervised link prediction methods mainly fall into two categories: neighborhood methods, such as *Common Neighbors* (CN) and *Jaccard's Coefficient* (JC), which make predictions based on structural scores that are calculated from the connections in the node's immediate neighbors, and path methods, such as *PageRank*, which predict the links based on the paths between nodes [66]. Essentially, the prediction score represents the similarity between the given pair of nodes: the higher the score, the more likely that there exists a connection between them. Using the *Common Neighbors* (CN) scoring method, two nodes with 10 common neighbors are more likely

to be linked than nodes with only a single common neighbor.

However, these neighborhood approaches intrinsically assume that the connections in the network are homogeneous: each node's connections are the outcome of one relationship. Directly applying homogeneous link predictors to overlapping communities can cause prediction errors. A simple example is shown in Figure 3.1, where two types of relationships co-exist within the same network. The solid line represents the coauthorship of a paper in a data mining conference and the dashed line represents the activity of collaborating on a machine learning paper (the link types are hidden from the method — only the presence of a link is known). Author 1 is associated with 2 affiliations since he/she participates in both activities. If all interactions were considered homogeneously, the prediction score for linking authors 2 and 6, $CN(2,6)$ and that for authors 2 and 3, $CN(2,3)$ under the *Common Neighbors* scoring method would be the same, since both node pairs share only one common neighbor; yet this is clearly wrong. The question now becomes how can we capture type correlations between edges to avoid being misled by connection heterogeneity? In the next section, we describe how edges in the network can be analyzed using Edge-Clustering [109] to construct a social feature space that makes this possible.

4.3 Proposed Link Prediction Scheme: Reweighting the Network

Most of previous work in link prediction focuses on node-similarity metrics computed for unweighted networks, where the strength of relationships is not taken into account. However, proximities between nodes can be estimated better by using both graph proximity measures and the weights of existing links [26, 74]. Most of this prior work uses the number of encounters between users as the link weights. However, as the structure of the network can be highly informative, social dimensions provide an effective way of differentiating the nodes in collaborative networks [109, 117].

In this dissertation, the weights of the link are evaluated based on the user's social fea-

tures extracted from the network topology under different similarity measures. For our domain, we evaluated several commonly used metrics including inner product, cosine similarity, and *Histogram Intersection Kernel* (HIK), which is used to compare color histograms in image classification tasks [6]. Since our social features can be regarded as the histogram of person's involvement in different potential groups, HIK can be also adopted to measure the similarity between two people. Given the social features of person v_i and person v_j, $(SF_i, SF_j) \in \mathcal{X} \times \mathcal{X}$, the HIK is defined as follows:

$$K_{\text{HIK}}(v_i, v_j) = \sum_{i=1}^{m} \min\{SF_i, SF_j\}, \tag{4.1}$$

where m is the length of the feature vector.

The closeness of users can also be evaluated by the total number of common link clusters they associate with. We call this measure "*Common Link Clusters*" (CLC). Section 4.3.5.1 compares the classification performance of these similarity metrics.

4.3.1 Unsupervised Proximity Metrics

In order to investigate the impact of link weights for link prediction in collaboration networks, we compare the performances of eight benchmark unsupervised metrics for unweighted networks and their extensions for weighted networks. The prediction scores from these unsupervised metrics can further be used as the attributes for learning supervised prediction models. We detail the unsupervised prediction metrics for both unweighted and weighted networks in the following sections.

Let $\mathcal{N}(x)$ be the set of neighbors of node x in the social network and let D_x be the degree (the total number of neighbors) of node x. Obviously, in an unweighted network, $D_x = |\mathcal{N}(x)|$. Let $w(x, y)$ be the link weight between nodes x and y in a weighted network. Note that in our generated weighted network, the weight matrix W is symmetric, i.e. $w(x, y) = w(y, x)$.

•**Number of Common Neighbors (CN)**: the *CN* measure for unweighted networks is de-

fined as the number of nodes with direct connections to the given nodes nodes x and y:

$$CN(x, y) = |\mathcal{N}(x) \cap \mathcal{N}(y)|. \tag{4.2}$$

The *CN* measure is one the most widespread metrics adopted in link prediction, mainly due to its simplicity. Intuitively, the measure simply states that two nodes that share a high number of common neighbors should be directly linked [83]. For weighted networks, the *CN* measure can be extended as:

$$CN(x, y) = \sum_{z \in \mathcal{N}(x) \cap \mathcal{N}(y)} w(x, z) + w(y, z). \tag{4.3}$$

•**Jaccard's Coefficient (JC)**: the *JC* measure assumes that the node pairs that share a higher proportion of common neighbors relative to their total number of neighbors are more likely to be linked. From this point of view, *JC* can be regarded as a normalized variant of *CN*. For unweighted networks, the *JC* measure is defined as:

$$JC(x, y) = \frac{|\mathcal{N}(x) \cap \mathcal{N}(y)|}{|\mathcal{N}(x) \cup \mathcal{N}(y)|}. \tag{4.4}$$

For weighted networks, the *JC* measure can be extended as:

$$JC(x, y) = \frac{\sum_{z \in \mathcal{N}(x) \cap \mathcal{N}(y)} w(x, z) + w(y, z)}{\sum_{a \in \mathcal{N}(x)} w(x, a) + \sum_{b \in \mathcal{N}(x)} w(y, b)}. \tag{4.5}$$

•**Preferential Attachment (PA)**: the *PA* measure assumes that the probability that a new link is created from a node x is proportional to the node degree D_x (i.e., nodes that currently have a high number of relationships tend to create more links in the future). Newman proposed that the product of a node pair's number of neighbors should be used as a measure for the probability of a

future link between those two [83]. The *PA* measure for an unweighted network is defined by:

$$PA(x, y) = |\mathcal{N}(x)| \times |\mathcal{N}(y)|. \tag{4.6}$$

The *PA* measure extended for a weighted network can be defined as:

$$PA(x, y) = \sum_{z_1 \in \mathcal{N}(x)} w(x, z_1) \times \sum_{z_2 \in \mathcal{N}(y)} w(y, z_2). \tag{4.7}$$

•**Adamic/Adar Coefficient (AA)**: the *AA* measure is related to *Jaccard's coefficient* with additional emphasis on the importance of the common neighbors [1]. *AA* defines higher weights for the common neighbors that have fewer neighbors. The *AA* measure for unweighted networks is defined as:

$$AA(x, y) = \sum_{z \in \mathcal{N}(x) \cap \mathcal{N}(y)} \frac{1}{\log(\mathcal{N}(z))}. \tag{4.8}$$

The *AA* measure extended for a weighted network can be defined as:

$$AA(x, y) = \sum_{z \in \mathcal{N}(x) \cap \mathcal{N}(y)} \frac{w(x, z) + w(y, z)}{\log(1 + \sum_{c \in \mathcal{N}(z)} w(z, c))}. \tag{4.9}$$

•**Resource Allocation Index (RA)**: the Resource Allocation Index has a similar formula as the Adamic-Adar Coefficient, but with a different underlying motivation. *RA* is based on physical processes of resource allocation [87] and can be applied on networks formed by airports (for example, flow of aircraft and passengers) or networks formed by electric power stations such as power distribution. The *RA* measure was first proposed in [130] and for unweighted networks it is expressed as follows:

$$RA(x, y) = \sum_{z \in \mathcal{N}(x) \cap \mathcal{N}(y)} \frac{1}{|\mathcal{N}(z)|}. \tag{4.10}$$

The *RA* measure for weighted networks can be defined as:

$$RA(x,y) = \sum_{z \in \mathcal{N}(x) \cap \mathcal{N}(y)} \frac{w(x,z) + w(y,z)}{\sum_{c \in \mathcal{N}(z)} w(z,c)}. \tag{4.11}$$

•**Inverse Path Distance (IPD)**: the *Path Distance* measure for unweighted networks simply counts the number of nodes along the shortest path between x and y in the graph. Thus, when two nodes x and y share at least one common neighbor, then $PD(x,y) = 1$. In this dissertation, we adopt the *Inverse Path Distance* to measure the proximity between two nodes, where

$$IPD(x,y) = 1/PD(x,y) \tag{4.12}$$

IPD is based on the intuition that nearby nodes are likely to be connected. In a weighted network, IPD is defined by the inverse of the shortest weighted distance between two nodes. Since IPD quickly approaches 0 as path lengths increase, for computational efficiency, we terminate the shortest path search once the distance exceeds a threshold L and approximate IPD for more distant node pairs as 0.

•**PropFlow**: is a new unsupervised link prediction method which calculates the probability that a restricted random walk starting at x ends at y in L steps or fewer using link weights as transition probabilities [61] . The walk terminates when reaching node y or revisiting any nodes including node x. By restricting its search within the threshold L, *PropFlow* is a local measure that is insensitive to noise in network topology far from the source node and can be computed quite efficiently. The algorithm for unweighted networks is identical to that for weighted networks, except that all link weights are set equal to 1.

•**PageRank**: the PageRank (PR) algorithm of Google fame was first introduced in [18]; it aims to represent the significance of a node in a network based on the significance of other nodes that link to it. Inspired by the same assumption as made by*Preferential Attachment*, we assume

that the links between nodes are driven by the importance of the node, hence the PageRank score of the target node represents a useful statistic. Essentially, PageRank outputs the ranking scores (or probability) of visiting the target node during a random walk from a source. A parameter α, the probability of suffering to a random node, is considered in the implementation. In our experiment, we set $\alpha = 0.85$ and perform an unoptimized PageRank calculation iteratively until the vector that represents PageRank scores converges.

For weighted network, we adopted the weighted PageRank algorithm proposed in [28].

$$PR_w(x) = \alpha \sum_{k \in \mathcal{N}(x)} \frac{PR_w(x)}{L(k)} + (1 - \alpha) \frac{w(x)}{\sum_{y=1}^{N} w(y)}. \tag{4.13}$$

where $L(x)$ is the sum of weights of outgoing links from node x, and $\sum_{y=1}^{N} w(y)$ is the total node weights across the whole network.

4.3.2 Supervised Link Predictor LPSF

As mentioned in [74], unsupervised link prediction methods exhibit several drawbacks. First, they can only perform well if the network link topology conforms to the scoring function *a priori*. In other words, the assumption is both the links in the existing network and the predicted links score highly on the given measure. Second, the ranking of node pairs is performed using only a single metric, and hence the strategy may completely explore different structural patterns contained in the network. By contrast, supervised link prediction schemes can integrate information from multiple measures and can usually better model real-world networks. Most importantly, unlike in other domains where supervised algorithms require access to appropriate quantities of labeled data, in link prediction we can use the existing links in the network as the source of supervision. For these reasons, supervised approaches to link prediction are drawing increased attention in the community [46, 92, 61].

In this dissertation, we follow a standard approach: we treat the prediction scores from the

unsupervised measures as features for the supervised link predictor. We compare the accuracy of different classifiers on both unweighted and weighted collaboration networks.

4.3.3 Experimental Setup

4.3.3.1 Multi-relational Dataset

Our proposed method is evaluated on two real-world multi-relational collaboration networks extracted from the DBLP dataset[1]. The DBLP dataset provides bibliographic information for millions of computer science references. In this dissertation we only consider authors who have published papers between 2006 and 2008, and extract their publication history from 2000 to 2008. In the constructed network, authors correspond to nodes, and two authors are linked if they have collaborated at least once. The link prediction methods are tested on the new co-author links in the subsequent time period [2009, 2010]. For the weighted variant, the number of coauthored publications is used as the weight on each link. Link heterogeneity is induced by the broad research topic of the collaborative work.

•DBLP-A: In the first DBLP dataset, we select 15 representative conferences in 6 computer science research areas (Databases, Data Mining, Artificial Intelligence, Information Retrieval, Computer Vision and Machine Learning), and each paper is associated with a research area if it is published in any conferences listed under that area. The collaboration network is constructed only for authors who have publications in those areas.

•DBLP-B: In the second DBLP dataset, we select 6 different computer science research areas (Algorithms & Theory, Natural Language Processing, Bioinformatics, Networking, Operating Systems and Distributed & Parallel Computing), and choose 16 representative conferences in these areas.

Similar DBLP datasets have previously been employed by Kong et al. to evaluate collective

[1] http://www.informatik.uni-trier.de/~ley/db/

classification in multi-relational networks [54]. In this dissertation, we aim to predict the missing links (coauthorship) in the future based on the existing connection patterns in the network.

Table 4.1: Data Statistics

Data	DBLP-A	DBLP-B
Categories	6	6
# of Nodes	10,708	6,251
# of New Links	12,741	5,592
# of Existing Links	49,754	30,130
Network Density	9.78×10^{-4}	1.7×10^{-3}
Maximum Degree	115	72
Average Degree	5.2	5.3

4.3.4 Evaluation of LPSF

In this dissertation, the supervised link prediction models are learned from training links (all existing links) in the DBLP dataset extracted between 2000 and 2008, and the performance of the model is evaluated on the testing links, new co-author link generated between 2009 and 2010. Link prediction using supervised learning model can be regarded as a binary classification task, where the class label (0 or 1) represents the link existence of the node pair. When performing the supervised classification, we sample the same number of non-connected node pairs as that of the existing links to use as negative instances for training the supervised classifier.

In our proposed *LPSF* model, the Edge-Clustering method is adopted to construct the initial social dimensions. When conducting the link prediction experiment, we use *cosine* similarity while clustering the links in the training set. The edge-based social dimension in our proposed method, *LPSF*, is constructed based on the edge cluster IDs using the *count* aggregation operator, and varying numbers of edge clusters are tested in order to provide the best performance of *LPSF*. The weighted network is then constructed according to the similarity score of connected nodes'

social features under the weight measure selected from Section 4.3. The search distance L for unsupervised metrics *Inverse Path Distance* and *PropFlow* is set to 5. We evaluate the performance of four supervised learning models in this section, which are *Naive Bayes* (NB), *Logistic Regression* (LR), *Neural Network* (NN) and *Random Forest* (RF). All algorithms have been implemented in WEKA [43], and the performance of each classifiers are tested using their default parameter setting.

In DBLP dataset, the number of positive link examples for testing is very small compared to negative ones. In this dissertation, we sample an equivalent number of non-connected node pairs as links from the 2009 and 2010 period to use as the negative instances in the testing set. The evaluation measures for link prediction performance used in this section are precision, recall and F-Measure.

4.3.5 Results

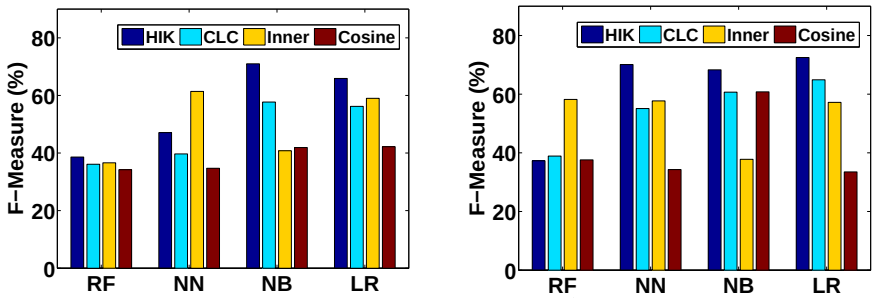
This section describes several experiments to study the benefits of augmenting link prediction methods using *LPSF*. First, we compare the performance of different weighting metrics used in *LPSF*. Second, we evaluate how the number of social features affects the performance of *LPSF*. Finally, we examine how several supervised link prediction models perform on unweighted and weighted networks, and the degree to which *LPSF* improves classification performance under different evaluation measures.

4.3.5.1 Choice of weighting metric for LPSF

A critical procedure in *LPSF* is reweighting the original networks according to the similarity of node pair's social features. Figure 4.1 shows the F-Measure performance of *LPSF* using different weighting metrics on DBLP datasets. Here the number of edge clusters is set to 1000 for all conditions, and different classifiers have been adopted for comparison purpose. We observe that in the DBLP-A dataset, even though the performance of each weighting metric is mainly dom-

inated by the choice of classifier, *Histogram Intersection Kernel* (HIK) and *Inner Product* perform better than *Hist* and *Cosine* in most cases. HIK dramatically outperforms *Cosine* in *Naive Bayes* by about 20% and *Inner* in *Logistic Regression* for 7%. The *Cosine* measure performs almost equally well for all classifiers but with a relatively low accuracy unfortunately.

In the DBLP-B dataset, while *Inner Product* performs well on *Random Forest*, HIK outperforms other weighting metrics using the other classifiers. Accordingly, we select HIK as our default weighting metric in *LPSF* for the remainder of the experiments.

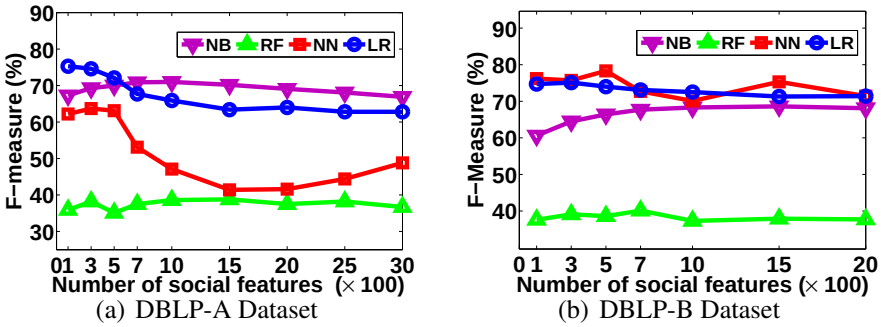


(a) DBLP-A Dataset (b) DBLP-B Dataset

Figure 4.1: Classification performance of *LPSF* on DBLP Dataset using different similarity measures on node's social features. The number of edge clusters is set to 1000, and *Histogram Intersection Kernel* (HIK) performs best in both datasets.

4.3.5.2 Choice of the number of social features

Here, we evaluate how the number of social features (edge clusters) affects the link prediction performance of *LPSF*, and Figure 4.2 shows the corresponding classification accuracy under the F-Measure metric. In the DBLP-A dataset, *Naive Bayes* and *Random Forest* are relatively robust to the number of social features while *Logistic Regression* and *Neural Network* perform better with a smaller number of social features (less than 500). Similarly in the DBLP-B dataset, *LPSF* demonstrates better performance with fewer social features. Therefore we set the number of social features to 300 and 500 for the DBLP-A and DBLP-B datasets respectively.



(a) DBLP-A Dataset (b) DBLP-B Dataset

Figure 4.2: Classification performance of *LPSF* using HIK on the DBLP Dataset with varying number of social features, using different supervised classifiers.

4.3.5.3 Supervised Link Prediction (LPSF Reweighting)

Figure 4.3 and 4.4 display the comparisons between *LPSF* and the baseline methods on DBLP dataset using a variety of supervised link classification techniques, against both the unweighted and weighted supervised baselines. The same features are used by all methods, with the only difference being the weights on the network links. In this dissertation, we compare the proposed method *LPSF* with alternate weighting schemes, such as the number of co-authored papers, as suggested in [26]. We see that in both DBLP datasets, *Unweighted*, *Weighted* and *LPSF* perform almost equally under Precision, though *LPSF* performs somewhat worse for some classifiers (*Random Forest* and *Naive Bayes*). When considering the number of collaborations between author pairs, the *Weighted* method slightly improves upon the performance of the *Unweighted* method. The *Weighted* approach receives the most improvements on *Naive Bayes* (3% on Recall and 5% on F-Measure) in the DBLP-A dataset and on *Neural Network* (5% on Recall and 10% on F-Measure).

The proposed reweighting (*LPSF*) offers substantial improvement over both the *Unweighted* and *Weighted* schemes on Recall and F-Measure in both datasets. In the DBLP-A dataset, *LPSF* outperforms the unweighted baseline the most dramatically on *Logistic Regression*, with about 23% improvement and 40% on Recall and F-Measure respectively. In the DBLP-B dataset, *LPSF* shows the best performance using *Neural Network* with accuracy improvements over baselines for

13% on Recall and 30% on F-Measure.

LPSF calculates the closeness between connected nodes according to their social dimensions, which captures the nodes' prominent interaction patterns embedded in the network and better addresses heterogeneity in link formation. By differentiating different types of links, *LPSF* is able to discover the possible link patterns between disconnected node pairs that may not be determined by the *Unweighted* and simple *Weighted* method, and hence exhibits great improvement on Recall and F-Measure. Since *LPSF* can be directly applied on the unweighted network, without considering any additional node information, it is thus broadly applicable to a variety of link prediction domains.

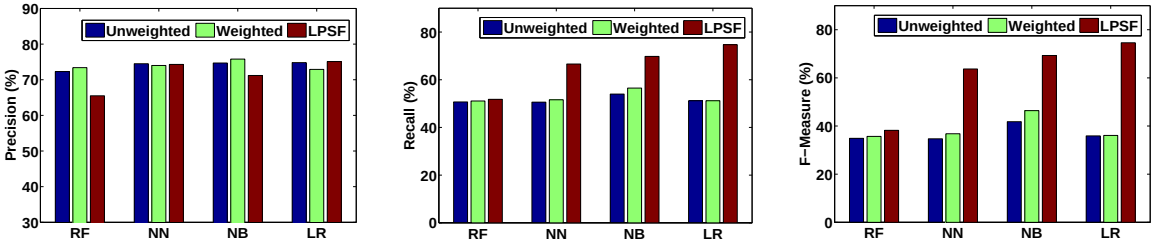


Figure 4.3: Comparing the classification performance of supervised link prediction models on unweighted and weighted DBLP-A networks using Precision, Recall and F-Measure. The proposed method (*LPSF*) is implemented using 300 edge clusters and the HIK reweighting scheme. Results show that *LPSF* significantly improves over both unweighted and weighted baselines, especially under Recall and F-Measures.

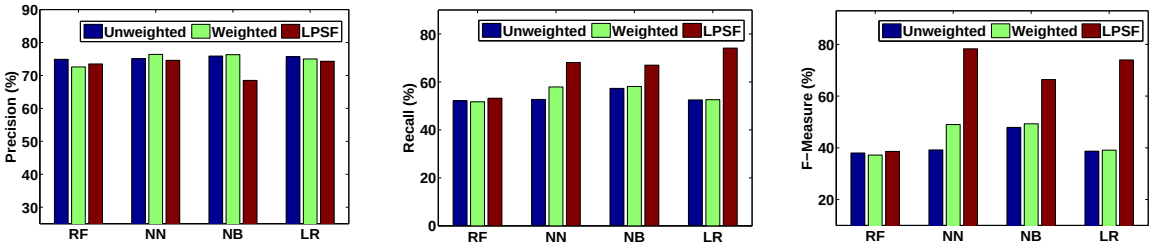


Figure 4.4: Comparing the classification performances of supervised link prediction models on unweighted and weighted DBLP-B networks using Precision, Recall and F-Measure. The proposed method (*LPSF*) is implemented using 500 edge clusters and the HIK reweighting scheme. Results show that *LPSF* significantly improves over both unweighted and weighted baselines, especially under Recall and F-Measures.

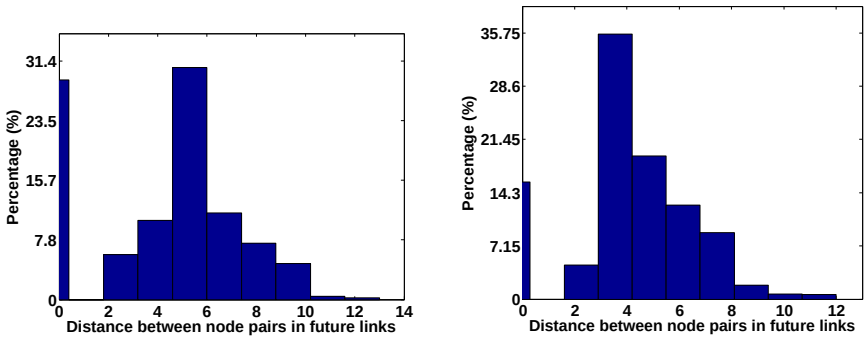
4.3.5.4 *Supervised Link Prediction: Choice of Classifier*

Figure 4.3 and 4.4 also enable us to compare different supervised classifiers for link prediction. We found that the performance of the classifiers varies from datasets. *Logistic Regression*, *Naive Bayes* and *Neural Network* exhibit comparable performance. Somewhat surprisingly, *Random Forest* does not perform well with *LPSF*. We also observe that *LPSF* using *Naive Bayes* will boost the Recall performance over baseline methods at the cost of lower Precision. Therefore *Logistic Regression* and *Neural Network* will be a better choice for *LPSF* in that they improve the Recall performance without decreasing the Precision. Using the traditional weighted features [26] does not help supervised classifiers for link prediction to a great extent. As discussed above, reweighting the unweighted collaboration network using our proposed technique, *LPSF*, performs the best.

4.4 Proposed Unsupervised Diffusion-based Link Prediction Models

Traditional unsupervised link prediction methods aim to measure the similarity for a node pair and use the affinity value to predict the existence of a link between them. The performance of link predictor is consequently highly dependent on the choice of pairwise similarity metrics. Most widely used unsupervised link predictors focus on the underlying local structural information of the data, which is usually extracted from the neighboring nodes within a short distance (usually 1-hop away) from the source. For instance, methods such as *Common Neighbors* and *Jaccard's Coefficient* calculate the prediction scores based on the number of directly shared neighbors between the given node pair. However, a recent study of coauthorship networks by Backstrom and Leskovec shows that researchers are more interested in establishing long-range weak ties (collaborations) rather than strengthening their well-founded interactions [4]. Figure 4.5 shows the distance distribution of newly collaborating authors between 2009 and 2010 in the DBLP datasets. We discover that in both datasets the majority of new links are generated by a node pair with a

minimal distance equal to or greater than two. This poses a problem for local link predictors which ignore information from the intermediate nodes along the path between the node pair.



(a)DBLP-A Dataset (b)DBLP-B Dataset

Figure 4.5: Probability distribution of the shortest distance between node pairs in future links (between 2009 and 2010) in the DBLP datasets. Distances marked as "0" are used to indicate that no path can be found that connects the given node pair.

In the past few years, the diffusion process (DP) model has attracted an increasing amount of interest for solving information retrieval problems in different domains [29, 124, 119]. DP aims to capture the geometry of the underlying manifold in a weighted graph that represents the proximity of the instances. First, the data are represented as a weighted graph, where each node represents an instance and edges are weighted according to their pairwise similarity values. Then the pairwise affinities are re-evaluated in the context of all connected instances, by diffusing the similarity values through the graph. The most common diffusion processes are based on random walks, where a transition matrix defines probabilities for walking from one node to a neighboring one, that are proportional to the provided affinities. By repeatedly making random walk steps on the graph, affinities are spread on the manifold, which in turn improves the obtainable retrieval scores. In the context of social network data, the data structure naturally leads to graph modeling, and graph-based methods have been proven to perform extremely well when combined with Markov chain techniques. In the following sections, we will explore the effectiveness of diffusion-based

methods on solving link prediction problems. The next section introduces the diffusion process model (DP) and an embedding method based on diffusion processes, diffusion maps (DM). Our proposed diffusion-based link prediction models (*LPDP* and *LPDM*) are discussed in 4.4.1 and 4.4.2.

4.4.1 Diffusion Process

We begin with the definition of a random walk on a graph $G = (V, E)$, which contains N nodes $v_i \in V$, and edges $e_{ij} \in E$ that link nodes to each other. The entries in the $N \times N$ affinity matrix A provide the edge weights between node pairs. The random walk transition matrix $\mathbf{P}$ can be defined as

$$\mathbf{P} = D^{-1}A \tag{4.14}$$

where D is a $N \times N$ diagonal matrix defined as:

$$d_{ij} = \begin{cases} \deg(i) & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

and $\deg(i)$ is the degree of the node i (i.e., the sum over its edge weights). The transition probability matrix $\mathbf{P}$ is a row-normalized matrix, where each row sums up to 1. Assuming $\mathbf{f}_0$, a $1 \times N$ dimensional vector of the initial distribution for a specific node, the single step of the diffusion process can be defined by the simple update rule:

$$\mathbf{f}_{t+1} = \mathbf{f}_t\mathbf{P} \tag{4.16}$$

Therefore, it is possible to calculate the probability vector $\mathbf{f}_t$ after t steps of random walks as

$$\mathbf{f}_t = \mathbf{f}_0\mathbf{P}^t \tag{4.17}$$

where $\mathbf{P}^t$ is the power of the matrix $\mathbf{P}$. The entry f_j^t in $\mathbf{f}_t$ measures the probability of going from the source node to node j in t time steps.

The PageRank algorithm described in Section 4.3.1 is one of the most successful webpage ranking methods and is constructed using a random walk model on the underlying hyperlink structures. In PageRank, the standard random walk is modified: at each time step t a node can walk to its outgoing neighbors with probability α or will jump to a random node with probability $(1-\alpha)$. The update strategy is as follows:

$$\mathbf{f}_{t+1} = \alpha \mathbf{f}_t \mathbf{P}^t + (1-\alpha)y \tag{4.18}$$

where y defines the probabilities of randomly jumping to the corresponding nodes. The PageRank algorithm iteratively updates the webpage's ranking distribution ($\mathbf{f}$) until it converges. One extension of the PageRank algorithm is *random walk with restart* (RWR) [88], which considers a random walker starting from node i, who will iteratively move to a random neighbor with probability α and return to itself with probability $1-\alpha$. In the RWR update, y in Equation 4.18 is simply a $1 \times N$ vector with the ith element equal to 1 and others to 0.

The diffusion process can further be extended to different independent instances by updating the probability matrix as follows:

$$\mathbf{W}_{t+1} = \alpha \mathbf{W}_t \mathbf{P}^t + (1-\alpha)\mathbf{Y} \tag{4.19}$$

where $\mathbf{W}$ is a $N \times N$ matrix that represents the local relationships (weights) between different instances. For networked data, the adjacency matrix A can be directly used as $\mathbf{W}$, and $\mathbf{P}$ can be formed by normalizing matrix W such that its rows add up to 1. Similarly, the $N \times N$ matrix $\mathbf{Y}$ consists of N personalized row vectors y.

In the literature, a number of diffusion models have been proposed by tuning the functions

for $\mathbf{W}$ for different application domains [88, 29, 119]. Our studies also reveal the choice of diffusion scheme has a substantial impact on the link prediction accuracy. In this dissertation, we adopt the updating scheme used for *Random Walk with Restart* in Equation 4.19. To apply the diffusion model on the link prediction problem, we calculate the prediction score for a given node pair (i, j) based on the corresponding entries in the final diffusion matrix:

$$LPDP(i, j) = W_{ij}^{(t)} \times W_{ji}^{(t)} \tag{4.20}$$

where $W_{ij}^{(t)}$ is the corresponding (i, j) entry in $\mathbf{W}_t$. Note that $\mathbf{W}_t$ is not necessarily a symmetric matrix, meaning $W_{ij}^t \neq W_{ji}^t$.

4.4.2 Diffusion Maps

The diffusion maps technique (DM), first introduced by Coifman and Lafon, applies the diffusion process model toward the problem of dimensionality reduction; it aims to embed the data manifold into a lower-dimensional space while preserving the intrinsic local geometric data structure [23]. Different from other dimensionality reduction methods such as principal component analysis (PCA) and multi-dimensional scaling (MDS), DM is a non-linear method that focuses on discovering the underlying manifold generating the sampled data. It has been successfully used on problems outside of social media analysis, including learning semantic visual features for action recognition [63].

As discussed in the previous section, in diffusion models, each entry $\mathbf{W}_{ij}^{(t)}$ indicates the probability of walking from i to j in t time steps. When we increase t, the diffusion process moves forward, and the local connectivity is integrated to reveal the global connectivity of the network. Increasing the value of t raises the likelihood that edge weights diffuse to nodes that are further away in the original graph. From this point of view, the $\mathbf{W}_t$ in the diffusion process reflects the intrinsic connectivity of the network, and the diffusion time t plays the role of a scaling factor for

data analysis.

Subsequently, the diffusion distance D is defined using the random walk forward probabilities p_{ij}^t to relate the spectral properties of a Markov chain (its matrix, eigenvalues, and eigenvectors) to the geometry of the data. The diffusion distance aims to measure the similarity of two points (N_i and N_j) using the diffusion matrix $\mathbf{W}_t$, which is in the form of:

$$[D^{(t)}(N_i, N_j)]^2 = \sum_{q \in \Omega} \frac{(W_{iq}^{(t)} - W_{jq}^{(t)})^2}{\varphi(N_q)^{(0)}} \tag{4.21}$$

where $\varphi(N_q)^{(0)}$ is the unique stationary distribution which measures the density of the data points.

Since calculating the diffusion distance is usually computationally expensive, spectral theory can be adopted to map the data point into a lower dimensional space such that the diffusion distance in the original data space now becomes the Euclidean distance in the new space. The diffusion distance can then be approximated with relative precision δ using the first k nontrivial eigenvectors and eigenvalues of $\mathbf{W}_t$ according to

$$[D^{(t)}(N_i, N_j)]^2 \simeq \sum_{s=1}^{k} (\lambda_s^t)^2 * (v_s(N_i) - v_s(N_j))^2 \tag{4.22}$$

where $\lambda_k^t > \delta \lambda_1^t$. If we use the eigenvectors weighted with λ as coordinates on the data, $D^{(t)}$ can be interpreted as the Euclidean distance in the low-dimensional space. Hence, the diffusion map embedding and the low-dimensional representation are given by

$$\Pi_t : N_i \Rightarrow \{\lambda_1^t v_1(N_i), \lambda_2^t v_2(N_i), \ldots, \lambda_k^t v_k(N_i)\}^T \tag{4.23}$$

The diffusion map Π_t embeds the data into a Euclidean space in which the distance is approxi-

mately the diffusion distance:

$$[D^{(t)}(N_i, N_j)]^2 \simeq \| \Pi_t(N_i) - \Pi_t(N_j) \|^2 \tag{4.24}$$

The diffusion maps framework for the proposed method *Link Prediction using Diffusion Maps* (LPDM) is summarized in Table 4.2. *LPDM* defines the link prediction score for a given node pair (N_i, N_j) by the diffusion distance, $D^{(t)}(N_i, N_j)$, between them.

Table 4.2: Algorithm: Diffusion maps on unweighted networked data

Objective: Given a weighted graph $\mathbf{W}$ with N nodes, embed all nodes into a k-dimensional space.

1. Create Markov transition matrix $\mathbf{P}$ by normalizing matrix $\mathbf{W}$ such that each row sums to 1.
2. Compute diffusion matrix $\mathbf{W}_t$ at diffusion time t using Equation 4.19.
3. Perform eigen-decomposition on $\mathbf{W}_t$, and obtain eigenvalue λs and eigenvectors v_s, such that $\mathbf{W}_t v_s = \lambda_s v_s$.
4. Embed data by DM using Equation 4.23.

4.4.3 Evaluation of LPDP and LPDM

In this dissertation, we evaluate the performance of our proposed diffusion-based link prediction models (*LPDP* and *LPDM*) on the same DBLP datasets mentioned in Section 4.3.3.1, and compare them with the eight unsupervised baselines listed in Section 4.3.1. Similar to the *LPSF* model, *LPDP* and *LPDM* can be applied on the weighted networks constructed with the Edge-Clustering method. In the later section, we compare the performance of *LPDP* and *LPDM* on both

unweighted and weighted DBLP networks. We use *cosine* similarity while clustering the links in the training set. Then the edge-based social dimension is constructed based on the edge cluster IDs using the *count* aggregation operator. We tested the algorithms with various numbers of edge clusters, and report the one offering the best performance of *LPDP* and *LPDM*. The similarity scores of the connected nodes' social features are measured using the *Histogram Intersection Kernel*, which are then used to construct the weighted network. The search distances L for unsupervised metrics *Inverse Path Distance* and *PropFlow* are set to 7 and 11 for the DBLP-A and DBLP-B datasets respectively.

We sample the same number of non-connected node pairs as that of the existing future links to be used as the negative training instances. The Area Under the Receiver Operating Characteristic curve (AUROC) is a standard measure of accuracy that relates the sensitivity (true positive rate) and specificity (true negative rate) of a classifier. In this dissertation, we report the performance of all unsupervised link prediction methods using AUROC.

4.4.4 Results

We conduct several experiments for evaluating the performance of the diffusion-based link predictors. First, we evaluate the link prediction performance of *LPDP* and *LPDM* on the unweighted DBLP datasets under different model parameter settings, such as the damping factor α and diffusion time t. For *LPDM*, we also examine how different sizes of the embedded diffusion spaces affect its link prediction performance. Additionally, we compare the diffusion-based link prediction models with other unsupervised benchmarks on both unweighted and weighted networks.

4.4.4.1 Effects of Diffusion Time on LPDP

As mentioned before, in diffusion processes, the diffusion time t controls the amount of weight likelihood that diffuses between long distance node pairs. The higher the value of t is, the

more likely the link weights are to diffuse to the nodes that are further away. Figure 4.6 shows the effect of varying diffusion time on the *LPDP* link prediction accuracy for the DBLP dataset. In this experiment, we fix the value of α to 0.9 which offers LPDP the best performance. We discover that setting t to a higher value does not guarantee higher link prediction accuracy. *LPDP* performs best when $t = 15$, yielding an AUROC accuracy 84.61% and 85.49% on DBLP-A and DBLP-B datasets respectively.

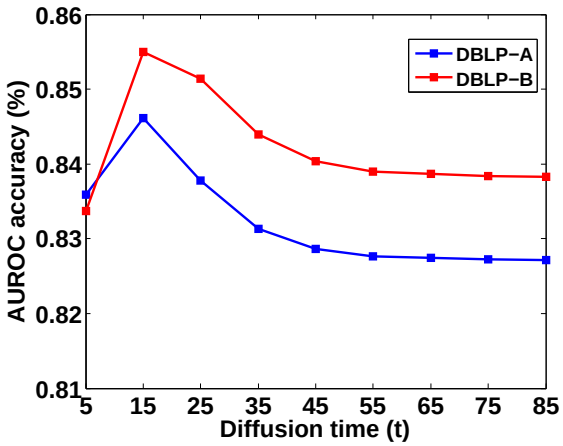
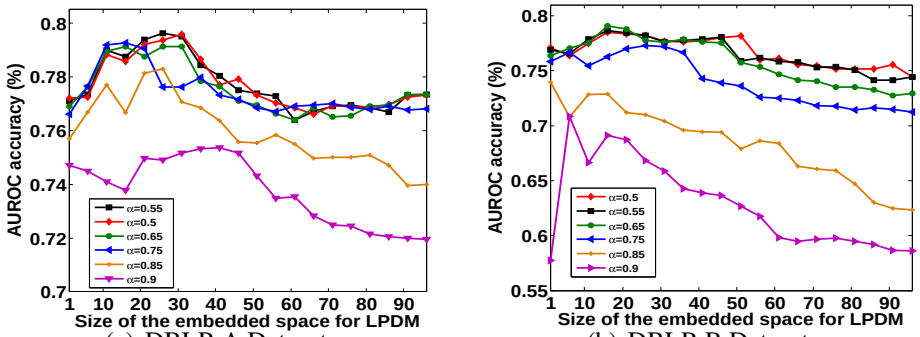


Figure 4.6: Link prediction performance (AUROC) of LPDP with fixed damping factor $\alpha = 0.9$ and varying diffusion time (t) on unweighted DBLP-A and DBLP-B datasets. LPDP performs best on both datasets when $t = 15$.

4.4.4.2 Effects of Damping Factor and Embedded Space Size on LPDM

Here, we evaluate how the size of the embedded space and the value of the damping factor affect the link prediction performance of *LPDM*. Figure 4.7 shows the corresponding classification accuracy measured by AUROC. The diffusion time t has an insignificant effect on the performance of *LPDM*, and the results we report here are based on setting t to 100 and 60 for DBLP-A and DBLP-B respectively. In both datasets, a lower damping factor α yields higher accuracy, and *LPDM* demonstrates the best performance when α equals 0.55 and 0.65 on DBLP-A and DBLP-B

respectively. Note that in Equation 4.19, a lower α results in a reduced probability of exchanges between a node and its connected neighbors. Our results reveal that the size of the embedded diffusion space greatly affects the performance of *LPDM*. Here we report experimental results for embedded diffusion space dimensions ranging from 1 and 100. As shown in Figure 4.7, the diffusion maps technique is able to identify semantically similar nodes by measuring distance on an embedded space with a much smaller dimensionality. *LPDM* exhibits the best performance (79.61% and 79.08%) when the size of the embedded space equals 25 and 15 on DBLP-A and DBLP-B respectively.



(a) DBLP-A Dataset (b) DBLP-B Dataset

Figure 4.7: AUROC accuracy of *LPDM* on DBLP datasets with varying damping factor α and embedded space size. The diffusion time t for $LPDM$ is set to 100 and 60 for DBLP-A and DBLP-B dataset respectively.

4.4.4.3 Comparing Unsupervised Link Prediction Methods

In Section 4.3.5, we evaluate our supervised link classifier *LPSF* which employs an ensemble of unsupervised measures as features. These unsupervised measures can themselves be used for classification, although we do not expect an individual feature to be competitive with the supervised combination. Here, we compare these unsupervised measures with our proposed diffusion-based measures *LPDP* and *LPDM* on unweighted and reweighted graphs. Table 4.3 and Table 4.4 summarize the link prediction performance (AUROC) of individual unsupervised fea-

tures on DBLP. We make several interesting observations.

First, we note that among the individual features, *PA* is by far the best performer. This is because *PA*'s model for link generation is a particularly good fit to the DBLP network structure and real-world academic publishing. It is true that highly published authors generate many more publications than their less prolific peers and will also seek to collaborate with other highly influential (high degree) authors in the future. Hence the "richer get richer" phenomenon definitely exists in coauthorship networks. Since the preferential attachment model is already a good match for the academic publishing domain, reweighting the links does not improve link prediction performance; in fact, performance drops slightly. This highlights the sensitivity of unsupervised classifiers to the link prediction domain.

Second, we observe that methods that rely on information gathered from node pairs' directly connected neighbors, such as *CN*, *JC*, *AA* and *RA*, perform poorly with accuracies only slightly above 50%. This result is not unexpected, given that the authorship distribution shown in Figure 4.5 reveals that DBLP authors are more likely to form future collaborations with authors with whom they share longer range ties. By collecting structural information from all nodes in the path, *IPD*, *PropFlow*, *PR*, *LPDP* and *LPDM* significantly improve the link prediction performance. Furthermore, in both the DBLP-A and DBLP-B datasets, the models that incorporate the random walk technique (*PR*, *LPDP* and *LPDM*) outperform the other two methods (*IPD* and *PropFlow*). *LPDP* performs the best among the three with an AUROC accuracy of 85.49% and 84.61% on DBLP-A and DBLP-B datasets respectively. Unfortunately the diffusion maps in *LPDM* are not able to capture the semantically similar nodes after the diffusion process which results in inferior performance to *LPDP*. *LPDM*'s performance is worse than *LPDP* by around 5%, while still performing better than *IPD* and *PropFlow*. This might be because the diffusion process after t diffusion time steps is good enough to capture the underlying similarity between nodes at farther distances using the node similarity extracted from the final diffusion matrix.

Third, Tables 4.3 and 4.4 also include the comparison results of different unsupervised link

predictors on weighted DBLP networks constructed using edge cluster information. On one hand, we found that in methods such as *CN*, *JC*, *AA* and *RA*, the weighting scheme does not affect the corresponding link prediction accuracy much. On the other hand, the weighting scheme helps to improve the performance of *IPD*, *PropFlow*, *PageRank* as well as *LPDM* by around 2%-3%. On both weighted datasets, *PageRank* performs best among all unsupervised features. It is also surprising that *LPDP* performs poorly on the weighted network, reducing the accuracy by 2% on the DBLP-A dataset and 4% on the DBLP-B dataset.

In summary, we observe that the reweighting scheme yields dramatic improvements in *LPSF* which integrates the first eight features listed in Table 4.3 in a supervised setting; however, it fails to boost the unsupervised performance of individual features. As mentioned in [65], the utility of using weights in link prediction is a somewhat controversial issue. Some case studies have shown that prediction accuracy can be significantly harmed when weights in the relationships were considered [65]. Our experiments reveal a more nuanced picture: although link weights (using the proposed approach) may not generate a large improvement for some individual unsupervised feature-level techniques, employing an appropriate choice of link weights (e.g., using *LPSF*) in conjunction with a supervised classifier enables us to achieve more accurate classification results on the DBLP datasets. The source code for *LPSF*, *LPDP* and *LPDM* is available at `https://github.com/jenniferwx/Link_Prediction_in_Multi-relational_Networks`.

Table 4.3: Link prediction accuracy of individual (unsupervised) classifiers on the DBLP-A dataset. Performance is evaluated on both unweighted networks and weighted networks constructed using social context features. Note that the reweighting scheme does not always improve accuracy at the individual feature level.

AUROC (%)	PA	AA	CN	JC	RA	IPD	PropFlow	PageRank	LPDP	LPDM
Unweighted	86.68	50.95	50.95	50.95	50.20	77.46	77.52	82.54	85.49	79.61
Weighted	85.16	50.95	50.95	50.95	50.20	80.06	79.71	85.61	83.08	80.43

Table 4.4: Link prediction accuracy of individual (unsupervised) classifiers on the DBLP-B dataset. Performances are evaluated on both unweighted networks and weighted networks constructed using social context features. Note that the reweighting scheme does not always improve accuracy at the individual feature level.

AUROC (%)	PA	AA	CN	JC	RA	IPD	PropFlow	PageRank	LPDP	LPDM
Unweighted	87.97	52.15	52.15	52.14	50.66	77.09	76.98	83.60	84.61	79.08
Weighted	87.11	52.15	52.15	52.15	50.66	76.23	76.66	87.14	80.11	80.09

CHAPTER 5: CONCLUSION

In this dissertation, we focus on the problem of learning collective behavior in multi-relational networks where the interaction links between actors are formed by different casual reasons. Traditional machine learning models on networked data leverage the class correlation between linked actors and treat all links in a homogeneous way. In multi-relational networks, users are naturally organized into overlapping communities. The heterogeneity of connection types in social media datasets creates issues when attempting to predict users' behaviors solely based on their neighbors.

We investigate two major problems in multi-relational networks in this dissertation: collective behavior classification and link prediction. We summarize our contributions in the following sections:

5.1 Collective Behavior Classification

We proposed two new frameworks to classify the collective behaviors in multi-relational networks. In both frameworks, we first adopted the unsupervised Edge-Clustering method to extract node's social features from network topology. The social feature is able to capture node's intrinsic involvements in different potential affiliations.

The proposed collective classification model using Fiedler embedding aims to construct an alternate social feature space from the initial social features to express the correlations between different entities (users and their connections). Also it is able to discover semantically similar users who are disconnected in the network.

The multi-label relational classifier (SCRN) addresses the issues that arise when directly applying the relational neighbor classifier (RN) on network data. SCRN combines the ability of relational neighbor classifiers to exploit label homophily while simultaneously leveraging social

feature similarity through the introduction of class propagation probabilities. The class propagation probability strives to capture, on a per class basis, how the given node resembles other nodes based upon the network structure, and modifies the probabilities of the node belonging to the different classes. Additionally, SCRN can be easily extended to employ content features constructed from node (e.g.,document) properties.

In this dissertation, we conducted a series of comparative experiments on various social media datasets (DBLP, IMDb, BlogCatalog and YouTube) to evaluate the classification performance of the proposed methods. We demonstrate that both methods outperform their baseline methods. The proposed embedded social feature space demonstrates its representational advantage for distinguishing different types of connections and predicting collective behavior. When the network exhibits high label homophily, SCRN is able to produce more accurate predictions by reducing incorrect class prediction probabilities during label propagation procedure.

5.2 Relationship Prediction

Most commonly-used link prediction methods assume that the network is in unweighted form, and treat each link equally. In this dissertation, we proposed a new link prediction framework LPSF that captures nodes' intrinsic interaction patterns from the network topology and embeds the similarities between connected nodes as link weights. The nodes' similarity is calculated based on social features extracted using Edge-Clustering to detect overlapping communities in the network. Experiments on the DBLP collaboration network demonstrate that a judicious choice of weight measure in conjunction with supervised link prediction enables us to significantly outperform existing methods. LPSF is better able to capture the true proximity between node pairs based on link group information and improves the performance of supervised link prediction methods.

We also found that the social features utilized effectively by the supervised version of LPSF are less useful in an unsupervised setting both with the raw proximity metrics and our two new

diffusion-based methods, (LPDP and LPDM). We observe that in the DBLP dataset researchers are more likely to collaborate with other highly published authors with whom they share weak ties which causes the random-walk based methods (PR, LPDP and LPDM) to generally outperform other benchmarks. Even though the reweighting scheme greatly boosts the performance of LPSF, it does not always have significant impact on its corresponding unsupervised features. In conclusion we note that any weighting strategy should be applied with caution when tackling the link prediction problem.

LIST OF REFERENCES

[1] L. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[2] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multi-scale complexity in networks, 2009. `http://arxiv.org/abs/0903.3178`.

[3] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multi-scale complexity in networks. *Nature*, 466:761–764, 2010.

[4] L. Backstrom and J. Leskovec. Supervised random walks: Predicting and recommending links in social networks. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 635–644, 2011.

[5] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations, 2002.

[6] A. Barla, F. Odone, and A. Verr. Histogram intersection kernel for image classification. In *Proceedings of International Conference on Image Processing*, volume 3, pages III–513–16, 2003.

[7] J. Baumes, M. Goldberg, and M. Magdon-ismail. Efficient identification of overlapping communities. In *Proceedings of IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 27–36, 2005.

[8] J. Baumes, M. K. Goldberg, M. S. Krishnamoorthy, M. M. Ismail, and N. Preston. Finding communities by clustering a graph into overlapping subgraphs. In *IADIS AC*, pages 97–104, 2005.

[9] N. Benchettara, R. Kanawati, and C. Rouveirol. Supervised machine learning applied to link prediction in bipartite social networks. In *Proceedings of International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 326–330, 2010.

[10] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, B-48:259–302, 1986.

[11] S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. *Computing Research Repository (CoRR)*, abs/1101.3291, 2011.

[12] S. Bhagat, I. Rozenbaum, and G. Cormode. Applying link-based classification to label blogs. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD, pages 92–101, 2007.

[13] M. Bilgic, G. M. Namata, and L. Getoor. Combining collective classification and link prediction. In *Proceedings of IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 381–386, 2007.

[14] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, (10), 2008.

[15] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan. Directed scale-free graphs. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 132–139, 2003.

[16] S. Boughorbely, J.-P. Tarel, and N. Boujemaa. Generalized histogram intersection kernel for image recognition. In *IEEE International Conference on Image Processing*, 2005.

[17] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of International Conference on World Wide Web (WWW)*, pages 107–117, 1998.

[18] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[19] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Community mining from multi-relational networks. In *Proceedings of European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2005.

[20] D. Chakrabarti, B. Dom, , P. Indyk, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of ACM International Conference on Management of Data*, pages 307–318, 1998.

[21] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

[22] D. Chen, M. Shang, Z. Lv, and Y. Fu. Detecting overlapping communities of weighted networks via a local algorithm. *Physica A: Statistical Mechanics and its Applications*, 389(19):4177–4187, 2010.

[23] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[24] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence (AAAI IAAI)*, pages 509–516, 1998.

[25] D. Davis, R. Lichtenwalter, and N. V. Chawla. Supervised methods for multi-relational link prediction. *Social Network Analysis and Mining*, (2):127–141, 2013.

[26] H. R. de Sá and R. B. C. Prudêncio. Supervised link prediction in weighted networks. In *Proceedigns of International Joint Conference on Neural Networks (IJCNN)*, pages 2281–2288, 2011.

[27] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 269–274, 2001.

[28] Y. Ding. Applying weighted pagerank to author citation networks. *Computing Research Repository (CoRR)*, abs/1102.1760, 2011.

[29] M. Donoser and H. Bischof. Diffusion processes for retrieval revisited. In *Proceedings of IEEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1320–1327, 2013.

[30] S. Džeroski. Multi-relational data mining: an introduction. *SIGKDD Exploration Newsletter*, 5(1):1–16, 2003.

[31] T. S. Evans. Clique graphs and overlapping communities. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(12), 2010.

[32] Y. Fan and C. R. Shelton. Learning continuous-time social network dynamics. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 161–168, 2009.

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

[34] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1300–1309, 1999.

[35] B. Gallagher, H. Tong, T. Eliassi-Rad, and C. Faloutsos. Using ghost edges for classification in sparsely labeled networks. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 256–264, 2008.

[36] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[37] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *Relational Data Mining*, 2001.

[38] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[39] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice (Chapman & Hall/CRC Interdisciplinary Statistics)*. 1 edition, Dec 1995.

[40] A. Globerson, G. Chechik, F. Pereira, and N. Tishby. Euclidean embedding of co-occurrence data. *The Journal of Machine Learning Research*, 8:2265–2295, 2007.

[41] A. Goldberg, X. Zhu, and S. Wright. Dissimilarity in graph-based semi-supervised classification. In *Proceedigns of International Conference on Artificial Intelligence and Statistics*, 2007.

[42] Y. Guo and S. Gu. Multi-label classification using conditional dependency networks. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1300–1305, 2011.

[43] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[44] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 2005.

[45] J. Han. Mining heterogeneous information networks by exploring the power of links. In *Proceedings of the 12th International Conference on Discovery Science*, pages 13–30, 2009.

[46] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning, 2006.

[47] R. Heatherly, M. Kantarcioglu, and X. Li. Social network classification incorporating link type. In *Proceedings of IEEE Intelligence and Security Informatics*, pages 19–24, 2009.

[48] D. Heckerman, C. Meek, and D. Koller. Probabilistic models for relational data. Technical report, Microsoft Research, 2004.

[49] B. Hendrickson. Latent semantic analysis and fiedler retrieval. In *Proceedings of SIAM Workshop On Text Ming*, 2006.

[50] B. Hendrickson. Latent semantic analysis and Fiedler retrieval. *Linear Algebra and Its Applications*, 421:345–355, 2007.

[51] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, pages 593–598, 2004.

[52] E. M. Jin, M. Girvan, and M. E. J. Newman. The structure of growing social networks. *Physical Review E*, 2001.

[53] T. Kato, H. Kashima, and M. Sugiyama. Integration of multiple networks for robust label propagation. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 716–726, 2008.

[54] X. Kong, X. Shi, and P. S. Yu. Multi-label collective classification. In *Proceedings of SIAM International Conference on Data Mining*, pages 618–629, 2011.

[55] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1), 2009.

[56] A. Lancichinetti and S. Fortunato. Consensus clustering in complex networks. *Scientific reports*, 2, 2012.

[57] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, (25):259–284, 1998.

[58] J. B. Lee and H. Adorna. Link prediction in a modified heterogeneous bibliographic network. In *Proceedings of International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 442–449, 2012.

[59] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

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

[61] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 243–252, 2010.

[62] J. Liu, S. Ali, and M. Shah. Recognizing human actions using multiple features. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[63] J. Liu, Y. Yang, and M. Shah. Learning semantic visual vocabularies using diffusion distance. In *Proceedings of IEEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 461–468, 2009.

[64] W. Liu and L. Lu. Link prediction based on local random walk. *EPL (Europhysics Letters)*, 85(5), 2010.

[65] L. Lü and T. Zhou. Role of weak ties in link prediction of complex networks. In *Proceedings of the ACM International Workshop on Complex Networks Meet Information and Knowledge Management*, pages 55–58, 2009.

[66] L. Lu and T. Zhou. Link prediction in complex networks: A survey. *Physica A*, 390(6):1150–1170, 2011.

[67] Q. Lu and L. Getoor. Link-based classification. In *Proceedings of 20th International Conference on Machine Learning (ICDM)*, pages 496–503, 2003.

[68] S. A. Macskassy and F. Provost. A simple relational classifier. In *Proceedings of the Second Workshop on Multi-Relational Data Mining*, pages 64–76, 2003.

[69] S. A. Macskassy and F. Provost. Classification in networked data: a toolkit and a univariate case study. *Macnine Learning*, 8:935–983, 2007.

[70] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrival*, 3(2):127–163, 2000.

[71] L. K. Mcdowell, K. M. Gupta, and D. W. Aha. Cautious inference in collective classification. *Machince Learning Research*, 10:2777–2836, 2009.

[72] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[73] M. Molloy and B. A. Reed. A critical point for random graphs with a given degree sequence. *Random Struct. Algorithms*, 6(2/3):161–180, 1995.

[74] T. Murata and S. Moriyasu. Link prediction of social networks based on weighted proximity measures. In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 85–88, 2007.

[75] J. Neville. *Statistical Models and Analysis Techniques for Learning in Relational Data*. PhD thesis, 2006.

[76] J. Neville, B. Gallagher, T. Eliassi-Rad, and T. Wang. Correcting evaluation bias of relational classifiers with network cross validation. *Knowledge and Information Systems*, pages 1–25, 2011.

[77] J. Neville and D. Jensen. Iterative classification in relational data. In *Proceedings of the AAAI Workshop on Learning Statistical Models from Relational Data*, pages 42–49, 2000.

[78] J. Neville and D. Jensen. Leveraging relational autocorrelation with latent group models. In *Proceedings of International Workshop on Multi-relational Mining*, pages 49–55, 2005.

[79] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.

[80] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 625–630, 2003.

[81] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 625–630, 2003.

[82] J. Neville, D. Jensen, and B. Gallagher. Simple estimators for relational bayesian classifiers. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM)*, pages 609–612, 2003.

[83] M. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2), 2001.

[84] M. Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[85] M. E. J. Newman. Detecting community structure in networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[86] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[87] Q. Ou, Y. D. Jin, T. Zhou, B. H. Wang, and B. Q. Yin. Power-law strength-degree correlation from resource-allocation dynamics on weighted networks. *Physical Review E*, page 021102. rXiv:physics/0603081.

[88] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 653–658, 2004.

[89] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[90] C. Perlich and F. Provost. Aggregation-based feature invention and relational concept classes. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 167–176, 2003.

[91] S. Peters, Y. Jacob, L. Denoyer, and P. Gallinari. Iterative multi-label multi-relational classification algorithm for complex social networks. *Social Network Analysis and Mining*, 2:17–29, 2012.

[92] A. Popescul, R. Popescul, and L. H. Ungar. Statistical relational learning for link prediction. In *Proceedigns of Workshop on Learning Statistical Models from Relational Data at the International Joint Conference on Articial Intelligence*, 2003.

[93] C. Pring. 216 social media and internet statistics. `http://thesocialskinny.com/216-social-media-and-internet-statistics-september-2012/`, 2012.

[94] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 2007.

[95] M. A. Rodriguez and J. Shinavier. Exposing multi-relational networks to single-relational network analysis algorithms. *Journal of Informetrics*, 4(1):29–41, 2010.

[96] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, (6):420–433, 1976.

[97] M. Rosvall and C. T. Bergstrom. Maps of information flow reveal community structure in complex networks. (arXiv:0707.0609), 2007.

[98] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[99] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[100] P. Sen and L. Getoor. Link-based classification. 2007.

[101] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, pages 93–106, 2008.

[102] S. Soundarajan and J. Hopcroft. Using community information to improve the precision of link prediction methods. In *Proceedings of International Conference Companion on World Wide Web (WWW)*, pages 607–608, 2012.

[103] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han. Co-author relationship prediction in heterogeneous bibliographic networks. In *Proceedings of International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 121–128, 2011.

[104] J. Tang, T. Lou, and J. Kleinberg. Inferring social ties across heterogenous networks. In *Proceedings of ACM International Conference on Web Search and Data Mining (WSDM)*, pages 743–752, 2012.

[105] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 990–998, 2008.

[106] L. Tang. Learning with large-scale social media networks. *Arizona State University*, 2010.

[107] L. Tang and H. Liu. *Community Detection and Mining in Social Media*. Synthesis Lectures on Data Mining and Knowledge Discovery.

[108] L. Tang and H. Liu. Relational learning via latent social dimensions. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, pages 817–826, 2009.

[109] L. Tang and H. Liu. Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of International Conference on Information and Knowledge Management (CIKM)*, pages 1107–1116, 2009.

[110] L. Tang and H. Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23, 2011.

[111] L. Tang, X. Wang, H. Liu, and L. Wang. A multi-resolution approach to learning with overlapping communities. In *Proceedings of the First Workshop on Social Media Analytics (SOMA)*, pages 14–22, 2010.

[112] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of Conference on Uncertainty in Artificial Intelligence*, pages 895–902, 2002.

[113] B. Taskar, M. fai Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2003.

[114] H. Tong. Fast random walk with restart and its applications. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*, pages 613–622, 2006.

[115] F. Wang, C. Zhang, H. C. Shen, and J. Wang. Semi-supervised classification using linear neighborhood propagation. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 160–167, 2006.

[116] X. Wang, M. Maghami, and G. Sukthankar. Leveraging network properties for trust evaluation in multi-agent systems. In *Proceedings of the IEEE International Conference on Intelligent Agent Technologies (IAT)*, pages 288–295, 2011.

[117] X. Wang and G. Sukthankar. Extracting social dimensions using Fiedler embedding. In *Proceedings of IEEE International Confernece on Social Computing*, pages 824–829, 2011.

[118] X. Wang, L. Tang, H. Gao, and H. Liu. Discovering overlapping groups in social media. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*, pages 569–578, 2010.

[119] J. Wanga, Y. Lia, X. Baib, Y. Zhanga, C. Wangc, and N. Tang. Learning context-sensitive similarity by shortest path propagation. *Pattern Recognition*, 44(10-11):2367 – 2374, 2011.

[120] Y. Weiss. Comparing the mean field method and belief propagation for approximate inference in mrfs. *Advanced Mean Field Methods*, 2001.

[121] E. W. Xiang. A survey on link prediction models for social network data, 2008.

[122] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: the state of the art and comparative study. *Computing Research Repository (CoRR)*, abs/1110.5813, 2011.

[123] J. Yang and J. Leskovec. Structure and overlaps of communities in networks. *arXiv preprint arXiv:1205.6228*, 2012.

[124] X. Yang, S. K"okar-tezel, and L. J. Latecki. Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. In *Proceedings of IEEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[125] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51:2282–2312, 2005.

[126] Z. Yin, M. Gupta, T. Weninger, and J. Han. A unified framework for link recommendation using random walks. In *2010 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 152–159, 2010.

[127] M. Zhang and Z. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[128] X. Zhang, Q. Yuan, S. Zhao, W. Fan, W. Zheng, and Z. Wang. Multi-label classification without the multi-label cost. In *Proceedings of SIAM International Conference on Data Mining*, 2010.

[129] E. Zheleva, L. Getoor, J. Golbeck, and U. Kuter. Using friendship ties and family circles for link prediction. In *Advances in Social Network Mining and Analysis, Second International Workshop (SNAKDD)*, pages 97–113, 2010.

[130] T. Zhou, L. Lü, and Y.-C. Zhang. Predicting missing links via local information. *The European Physical Journal B - Condensed Matter and Complex Systems*, 71(4):623–630, 2009.

[131] X. Zhu. *Semi-supervised Learning with Graphs*. PhD thesis, 2005.

[132] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, CMU CALD tech report, 2002.

[133] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. pages 912–919, 2003.

[134] A. Zuccala. Modeling the invisible college. *Journal of the American Society for Information Science and Technology*, 57(2):152–168, 2005.