Chapter 1

LINK PREDICTION IN SOCIAL NETWORKS

Link Prediction

Mohammad Al Hasan
eBay Research Labs
San Jose, CA 95125
mhasan@ebay.com

Mohammed J. Zaki
Department of Computer Science, Rensselaer Polytechnic Institute
Troy, NY 12180
zaki@cs.rpi.edu

Abstract Link prediction is an important task for analying social networks which also has applications in other domains like, information retrieval, bioinformatics and e-commerce. There exist a variety of techniques for link prediction, ranging from feature-based classification and kernel-based method to matrix factorization and probabilistic graphical models. These methods differ from each other with respect to model complexity, prediction performance, scalability, and generalization ability. In this article, we survey some representative link prediction methods by categorizing them by the type of the models. We largely consider three types of models: first, the traditional (non-Bayesian) models which extract a set of features to train a binary classification model. Second, the probabilistic approaches which model the joint-probability among the entities in a network by Bayesian graphical models. And, finally the linear algebraic approach which computes the similarity between the nodes in a network by rank-reduced similarity matrices. We discuss various existing link prediction models that fall in these broad categories and analyze their strength and weakness. We conclude the survey with a discussion on recent developments and future research direction.

Keywords: Link prediction, network evolution model, social network analysis, probabilistic model, local probabilistic model

1. Introduction

Social networks are a popular way to model the interactions among the people in a group or community. They can be visualized as graphs, where a vertex corresponds to a person in some group and an edge represents some form of association between the corresponding persons. The associations are usually driven by mutual interests that are intrinsic to a group. However, social networks are very dynamic, since new edges and vertices are added to the graph over time. Understanding the dynamics that drive the evolution of social network is a complex problem due to a large number of variable parameters. But, a comparatively easier problem is to understand the association between two specific nodes. For instance, some of the interesting questions that can be posed are: How does the association pattern change over time? What are the factors that drive the associations? How is the association between two nodes affected by other nodes? The specific problem instance that we address in this article is to predict the likelihood of a future association between two nodes, knowing that there is no association between the nodes in the current state of the graph. This problem is commonly known as the *Link Prediction* problem.

More formally, the link prediction task can be formulated as followed (based upon the definition in Liben-Nowell and Kleinberg [36]): Given a social network $G(V, E)$ in which an edge $e = (u, v) \in E$ represents some form of interactions between its endpoints at a particular time $t(e)$. We can record multiple interactions by parallel edges or by using a complex timestamp for an edge. For time $t \leq t'$ we assume that $G[t,\ t']$ denotes the subgraph of G restricted to the the edges with time-stamps between t and t'. In a supervised training setup for link prediction, we can choose a training interval $[t_0,\ t'_0]$ and a test interval $[t_1,\ t'_1]$ where $t'_0 < t_1$. Now the link prediction task is to output a list of edges not present in $G[t_0, t'_0]$, but are predicted to appear in the network $G[t_1, t'_1]$.

Link prediction is applicable to a wide variety of application areas. In the area of Internet and web science, it can be used in tasks like automatic web hyper-link creation [3] and web site hyper-link prediction [65]. In e-commerce, one of the most prominent usages of link prediction is to build recommendation systems [25, 37, 35]. It also has various applications in other scientific disciplines. For instance, in bibliography and library science, it can be used for deduplication [39] and record linkage [4]; in Bioinformatics, it has been used in protein-protein interaction (PPI) prediction [6] or to annotate the PPI graph [18]. In security related applications, it can be used to identify hidden groups of terrorists and criminals. In many of the above applications, the graphs that we work on are not necessarily social network graphs, rather they can be Internet, information networks, biological entity networks, and so on.

In this article, we present a survey of existing approaches to link prediction, with focus mainly on social network graphs. We classify the extant approaches into several groups. One group of the algorithms computes a similarity score between a pair of nodes so that a supervised learning method can be employed. In this class we also include methods that use a kernel matrix, and then employ a maximum margin classifier. Another class of algorithms consists of those based on Bayesian probabilistic models, and probabilistic relational models. Beside these, there are algorithms that are based on graph evolution models or on linear algebraic formulations. Several methods span multiple classes in the above classification scheme. After a brief overview, we discuss each group of methods in more detail below.

2. Background

Liben-Nowell and Kleinberg [36] proposed one of the earliest link prediction models that works explicitly on a social network. Every vertex in the graph represents a person and an edge between two vertices represents the interaction between the persons. Multiplicity of interactions can be modeled explicitly by allowing parallel edges or by adopting a suitable weighting scheme for the edges. The learning paradigm in this setup typically extracts the similarity between a pair of vertices by various graph-based similarity metrics and uses the ranking on the similarity scores to predict the link between two vertices. They concentrated mostly on the performance of various graph-based similarity metrics for the link prediction task. Later, Hasan et. al. [22] extended this work in two ways. First, they showed that using external data outside the scope of graph topology can significantly improve the prediction result. Second, they used various similarity metric as features in a supervised learning setup where the link prediction problem is posed as a binary classification task. Since then, the supervised classification approach has been popular in various other works in link prediction [10, 58, 15].

The link prediction problem has also been studied previously in the context of relational data [53, 46, 47] and also in the Internet domain [50], where explicit graph representations were not used. The prediction system proposed in these works can accept any relational dataset, where the objects in the dataset are related to each other in any complex manners and the task of the system is to predict the *existence* and the *type* of links between a pair of objects in the dataset. Probabilistic relational models [21], graphical models [40], stochastic relational models [6, 61, 20], and different variants of these are the main modeling paradigm used in these works. The advantages of these approaches include the genericity and ease with which they can incorporate the attributes of the entities in the model. On the down side, they are usually complex, and have too many parameters, many of which may not be that intuitive to the user.

The research on social network evolution [7, 32, 34] closely resembles the link prediction problem. An evolution model predicts the future edges of a network, taking into account some well known attributes of social networks, such as the power law degree distribution [7] and the small world phenomenon [32]. This remains the main difference between evolution models and the link prediction models. The former concentrate on the global properties of the network and the latter model the local states of the network to predict the probability of the existence of a link between a specific pair of nodes in the network. Nevertheless, the ideas from these models have been instrumental for some research works [29] that directly addressed the task of link prediction.

One of the main challenges of link prediction concerns the evolution of Internet scale social networks like facebook, mySpace, flickr, and so on. These networks are huge in size and highly dynamic in nature for which earlier algorithms may not scale and adapt well—more direct approaches are required to address these limitations. For instance, Tylenda et. al. [56] shows that utilizing the time stamps of past interactions, which explicitly utilize the lineage of interactions, can significantly improve the link prediction performance. Recently, Song et. al. [52] used matrix factorization to estimate similarity between the nodes in a real life social network having approximately 2 millions nodes and 90 millions edges. Any traditional algorithm that aims to compute pair-wise similarities between vertices of such a big graph is doomed to fail. Recently, the matrix based factorization works have been extended to the more richer higher-order models such as tensors [1].

Having outlined the background methods, we now review the existing methods to link prediction. We begin with feature-based methods that construct pair-wise features to use in a classification task. Next we consider Bayesian approaches, followed by the probabilistic relational models. After reviewing methods based on linear algebra, we present some recent trends and directions for future work.

Notation. Typically, we will use small letters, like x, y, z to denote a node in a social network, the edges are represented by the letter e. For a node x, $\Gamma(x)$ represents the set of neighbors of x. $degree(x)$ is the size of the $\Gamma(x)$. We use the letter A for the adjacency matrix of the graph.

3. Feature based Link Prediction

We can model the link prediction problem as a *supervised classification* task, where each data point corresponds to a pair of vertices in the social network graph. To train the learning model, we can use the link information from the training interval ($[t_0,\ t'_0]$). ¿From this model, predictions of future links in the test interval ($[t_1,\ t'_1]$) can be made. More formally, assume $u, v \in V$ are two vertices in the graph $G(V, E)$ and the label of the data point $\langle u, v\rangle$ is $y^{\langle u,v\rangle}$. Note that we assume that the interactions between u and v are symmetric, so the pair $\langle u, v\rangle$ and $\langle v, u\rangle$ represent the same data point, hence, $y^{\langle u,v\rangle} = y^{\langle v,u\rangle}$. Now,

$$y^{\langle u,v\rangle} = \begin{cases} +1, & \text{if } \langle u,v\rangle \in E \\ -1, & \text{if } \langle u,v\rangle \notin E \end{cases}$$

Using the above labeling for a set of training data points, we build a classification model that can predict the unknown labels of a pair of vertices $\langle u, v\rangle$ where $\langle u, v\rangle \notin E$ in the graph $G[t_1, t'_1]$.

This is a typical binary classification task and any of the popular supervised classification tools, such as naive Bayes, neural networks, support vector machines (SVM) and k nearest neighbors, can be used. But, the major challenge in this approach is to choose a set of features for the classification task. Next we will discuss the set of features that have been used successfully for supervised link prediction tasks.

3.1 Feature Set Construction

Choosing an appropriate feature set is the most critical part of any machine learning algorithm. For link prediction, each data point corresponds to a pair of vertices with the label denoting their link status, so the chosen features should represent some form of proximity between the pair of vertices. In existing research works on link prediction, majority of the features are extracted from the graph topology. Also, some works develop a feature set constructed from a graph evolution model. Besides these, the attributes of vertices and edges can also be very good features for many application domains.

The features that are based on graph topology are the most natural for link prediction. Here we call them graph-topological feature. In fact, many works [36, 29] on link prediction concentrated only on the graph topological feature-set. Typically, they compute the similarity based on the node neighborhoods or based on the ensembles of paths between a pair of nodes. The advantage of these features are that they are generic and are applicable for graphs from any domain. Thus, no domain knowledge is necessary to compute the values of these features from the social network. However, for large social networks, some of these features may be computationally expensive. Below we explain some of the popular graph topological features under two categories: (1) Node neighborhood based and (2) Path based. Majority of these features are adapted from [36, 22]. Following that we discuss a set of features that are extracted from the vertex or edge properties of the graph.

3.1.1 Node Neighborhood based Features.

Common Neighbors. For two nodes, x and y, the size of their common neighbors is defined as $|\Gamma(x) \cap \Gamma(y)|$. The idea of using the size of common neighbors is just an attestation to the network transitivity property. In simple words, it means that in social networks if vertex x is connected to vertex z and vertex y is connected to vertex z, then there is a heightened probability that vertex x will also be connected to vertex y. So, as the number of common neighbors grows higher, the chance that x and y will have a link between them increases. Newman [41] has computed this quantity in the context of collaboration networks to show that a positive correlation exists between the number of common neighbors of x and y at time t, and the probability that they will collaborate in the future.

Jaccard Coefficient. The common neighbors metric is not normalized, so one can use the Jaccard Coefficient, which normalizes the size of common neighbors as below:

$$\textbf{Jaccard-coefficient}(x{,}y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \tag{1.1}$$

Conceptually, it defines the probability that a common neighbor of a pair of vertices x and y would be selected if the selection is made randomly from the union of the neighbor-sets of x and y. So, for high number of common neighbors, the score would be higher. However, from the experimental results of four different collaboration networks, Liben-Nowell et. al. [36] showed that the performance of Jaccard coefficient is worse in comparison to the number of common neighbors.

Adamic/Adar. Adamic and Adar [2] proposed this score as a metric of similarity between two web pages. For a set of features z, it is defined as below.

$$\sum_{z\,:\,\text{feature shared by }x{,}y} \frac{1}{\log(frequency(z))} \tag{1.2}$$

For link prediction, [36] customized this metric as below, where the common neighbors are considered as features.

$$\textbf{adamic/adar}(x,\, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|} \tag{1.3}$$

In this way, Adamic/Adar weighs the common neighbors with smaller degree more heavily. From the reported results of the existing works on link prediction, Adamic/Adar works better than the previous two metrics.

3.1.2 Path based Features.

Shortest Path Distance. The fact that the friends of a friend can become a friend suggests that the path distance between two nodes in a social network can influence the formation of a link between them. The shorter the distance, the higher the chance that it could happen. But, also note that, due to the small world [59] phenomenon, mostly every pair of nodes is separated by a small number of vertices. So, this feature sometimes does not work that well. Hasan et. al. [22] found this feature to have an average rank of 4 among 9 features that they used in their work on link prediction in a biological co-authorship network. Similar finding of poor performance by this feature was also reported in [36].

Katz. Leo Katz proposed this metric in [31]. It is a variant of shortest path distance, but generally works better for link prediction. It directly sums over all the paths that exist between a pair of vertices x and y. But, to penalize the contribution of longer paths in the similarity computation it exponentially damps the contribution of a path by a factor of β^l, where l is the path length. The exact equation to compute the Katz value is as below:

$$\textbf{katz}(x,y) = \sum_{l=1}^{\infty} \beta^l \ . \ |\text{paths}_{x,y}^{\langle l \rangle}| \tag{1.4}$$

where $|\text{p}aths_{x,y}^{\langle l \rangle}|$ is the set of all paths of length l from x to y. Katz generally works much better than the shortest path since it is based on the ensemble of all paths between the nodes x and y. The parameter $\beta(\leq 1)$ can be used to regularize this feature. A small value of β considers only the shorter paths for which this feature very much behaves like features that are based on the node neighborhood. One problem with this feature is that it is computationally expensive. It can be shown that the Katz score between all the pairs of vertices can be computed by finding $(I - \beta A)^{-1} - I$, where A is the adjacency matrix and I is an identity matrix of proper size. This task has roughly cubic complexity which could be infeasible for large social networks.

Hitting Time. The concept of hitting time comes from random walks on a graph. For two vertices, x and y in a graph, the hitting time, $H_{x,y}$ defines the expected number of steps required for a random walk starting at x to reach y. Shorter hitting time denotes that the nodes are similar to each other, so they have a higher chance of linking in the future. Since this metric is not symmetric, for undirected graphs the commute time, $C_{x,y} = H_{x,y} + H_{y,x}$, can be used. The benefit of this metric is that it is easy to compute by performing some trial random walks. On the downside, its value can have high variance; hence, prediction by this feature can be poor [36]. For instance, the hitting time between x and y can be affected by a vertex z, which is far away from x and y; for instance, if z has high stationary probability, then it could be hard for a random walk to escape from the neighborhood of z. To protect against this problem we can use random walks with restart, where we periodically reset the random walk by returning to x with a fixed probability α in each step. Due to the scale free nature of a social network some of the vertices may have very high stationary probability (π) in a random walk; to safeguard against it, the hitting time can be normalized by multiplying it with the stationary probability of the respective node, as shown below:

$$\textbf{normalized-hitting-time}(x,y) = H_{x,y} \cdot \pi_y + H_{y,x} \cdot \pi_x \qquad (1.5)$$

Rooted Pagerank. Chung and Zhao [13] showed that the Pagerank [11] measures that is used for web-page ranking has inherent relationship with the hitting time. So, pagerank value can also be used as a feature for link prediction. However, since pagerank is an attribute of a single vertex, it requires to be modified so that it can represent a similarity between a pair of vertices x and y. The original definition of pagerank denotes the importance of a vertex under two assumptions: for some fixed probability α, a surfer at a web-page jumps to a random web-page with probability α and follows a linked hyperlink with probability $1 - \alpha$. Under this random walk, the importance of an web-page v is the expected sum of the importance of all the web-pages u that link to v. In random walk terminology, one can replace the term *importance* by the term *stationary distribution*. For link prediction, the random walk assumption of the original pagerank can be altered as below: similarity score between two vertices x and y can be measured as the stationary probability of y in a random walk that returns to x with probability

$1 - \beta$ in each step, moving to a random neighbor with probability β. This metric is assymetric and can be made symmetric by summing with the counterpart where the role of x and y are reversed. In [36], it is named as *rooted pagerank*. The rooted pagerank between all node pairs (represented as RPR) can be derived as follows. Let D be a diagonal *degree matrix* with $D[i,i] = \sum_j A[i,j]$. Let, $N = D^{-1}A$ be the adjacency matrix with row sums normalized to 1. Then,

$$RPR = (1-\beta)(I-\beta N)^{-1}$$

3.1.3 Features based on Vertex and Edge Attributes.

Vertex and edge attributes play an important role for link prediction. Note that, in a social network the links are directly motivated by the utility of the individual representing the nodes and the utility is a function of vertex and edge attributes. Many studies [22, 15] showed that vertex or edge attributes as proximity features can significantly increase the performance of link prediction tasks. For example, Hasan et. al. [22] showed that for link prediction in a co-authorship social network, attributes such as the degree of overlap among the research keywords used by a pair of authors is the top ranked attribute for some datasets. Here the vertex attribute is the research keyword set and the assumption is that a pair of authors are close (in the sense of a social network) to each other, if their research work evolves around a larger set of common keywords. Similarly, the Katz metric computed the similarity between two web-pages by the degree to which they have a larger set of common words where the words in the web-page are the vertex attributes. The advantage of such a feature set is that it is generally cheap to compute. On the down-side, the features are very tightly tied with the domain, so, it requires good domain knowledge to identify them. Below, we will provide a generic approach to show how these features can be incorporated in a link prediction task.

Vertex Feature Aggregation. Once we identify an attribute a of a node in a social network, we need to devise some meaningful aggregation function, f. To compute the similarity value between the vertices x and y, f accepts the corresponding attribute values of these vertices to produce a similarity score. The choice of function entirely depends on the type of the attribute. In the followings we show two examples where we aggregated some local metric of a vertex.

- *Preferential Attachment Score:* The concept of preferential attachment [8] is akin to the well known *rich gets richer* model. In short, it proposes that a vertex connect to other vertices in the network based on the probability of their degree. So, if we consider the neighborhood size as feature value, then multiplication can be an aggregation function, which is named as preferential attachment score:

$$\textbf{preferential attachmend score}(x,\ y) = \Gamma(x)\ .\ \Gamma(y) \qquad (1.6)$$

Actually, the summation function can also be used to aggregate the feature values. In Hasan et. al. [22], the authors show that the summation of the neighbor-count of a pair of vertices is a very good attribute, which stands out as the second ranked feature in the link prediction task in a co-authorship network.

- *Clustering Coefficient Score:* Clustering coefficient of a vertex v is defined as below.

$$\textbf{clustering coef.}(v) = \frac{3 \times \#\text{ triangles adjacent to } u}{\#\text{ possible triples adjacent to } u} \qquad (1.7)$$

To compute a score for link prediction between the vertex x and y, one can sum or multiply the clustering coefficient score of x and y.

Kernel feature Conjunction. In many domains, there could be numerous vertex attributes or the attributes could be complex or attribute values between a pair of instances may have no apparent match between them, hence direct application of aggregation function to each such attributes could be either cumbersome or misleading. In such a scenario, one can use pairwise kernel based feature conjunction [43, 9]. The basic idea is to obtain a kernel function that computes the similarity between two pairs of instances from the feature space which is expanded through Cartesian product. More details on this approach will be given below in Section 3.2.

Extended Graph Formulation. For a categorical vertex attribute, we can make an extended graph where the social network is extended by additional vertices where each additional vertex represents a specific attribute. The additional vertices can have a link among themselves based on co-existence of other similarity properties. Moreover, an original vertex can also be connected to an attribute vertex if that vertex shares that attribute value. This process can be repeated for any number of vertex attributes. Now, all the graph topological metrics can be deployed in the extended graph to compute a similarity score which considers both attributes and graph topology. For example, for link prediction in a co-authorship network, Hasan et. al. [22] considered an author-keyword extended graph where an additional vertex is added for each keyword. Each keyword node is connected to an author node, if that keyword is used by the authors in any of his papers. Moreover, two keywords that appear together in any paper are also connected by an edge. In this way, if two vertices do not have any matching values for an attribute, they can still be similar through the similarity link among the attributes values. Say, an author x is connected to a keyword node, named *machine learning* and the author y is connected to another keyword node, named *information retrieval* and if *machine learning* and *information retrieval* are connected to each other in this extended graph, attribute based similarity between node x and y can be inferred through the extended graph.

Generic simRank. In the above extended graph, we use the concept that "two objects are similar if they are similar to two similar objects". Jeh and Widom [27] suggested a generic metric called *SimRank* which captures this notion recursively. The simRank score is the fixed point of the following recursive equation.

$$\textbf{simRank(x,y)} = \begin{cases} 1 & \text{if } x = y \\ \gamma \ . \ \frac{\sum_{a \in \Gamma(x)} \sum_{b \in \Gamma(y)} simRank(a,b)}{|\Gamma(x)| \ . \ |\Gamma(y)|} & \text{otherwise} \end{cases}$$

Note that, if we apply simRank in the extended graph, the similarity score considers both the graph topological and attribute based similarity.

3.2 Classification Models

There exist a plethora of classification models for supervised learning, such as decision trees, naive Bayes, neural networks, SVMs, k nearest neighbors, and ensemble methods like bagging and boosting. Also regression models like logistic regression can also be used for this task [38]. Although their performances are comparable, some usually work better than others for a specific dataset or domain. In [22], the authors found that for a co-authorship social network, bagging and support vector machines have marginal competitive edge. However, learning a model for a link prediction task has some specific challenges that may make some models more attractive than others.

In this section we first discuss the specific challenges when modeling link prediction as a classification task. We then discuss supervised learning models that are custom-made to cope with some of these challenges.

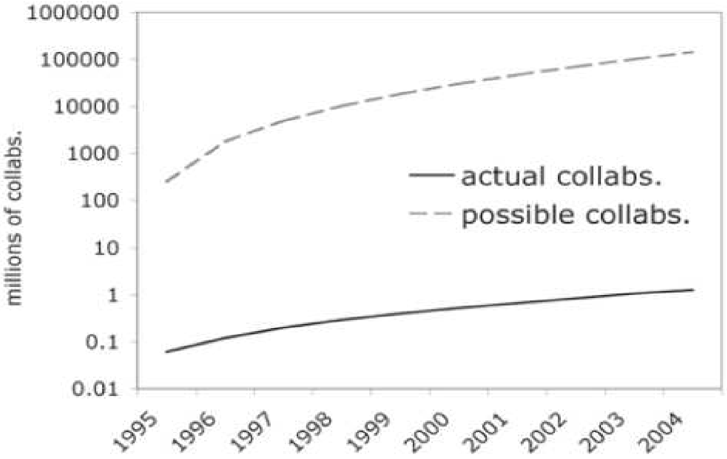

Figure 1.1. Logarithmic plot of actual and possible collaborations between DBLP authors, 1995-2004 [49].

Challenges for Link Prediction as Classification. The first challenge in supervised link prediction is extreme *class skewness*. The number of possible links is quadratic in the number of vertices in a social network, however the number of actual links (the edges in the graph) added to the graph is only a tiny fraction of this number. This results in large class skewness, causing training and inference to become difficult tasks. Hasan et. al. [22] reported very good performance of link prediction on DBLP and BIOBASE datasets, but they ignored the class distribution and reported cross validation performance from a dataset where the population is balanced. It is fair to say that the performance would drop (sometimes significantly) if the original class distribution were used. Rattigan an Jensen [49] studied this problem closely. As illustrated in Figure 1.1, they showed that in the DBLP dataset, in the year 2000, the ratio of actual and possible link is as low as 2×10^{-5}. So, in a uniformly sampled dataset with one million training instances, we can expect only 20 positive instances. Even worse, the ratio between the number of positive links and the number of possible links also slowly decreases over time, since the negative links grow quadratically whereas positive links grow only linearly with a new node. As reported in [49], for a period of 10 years, from 1995 to 2004 the number of authors in DBLP increased from 22 thousand to 286 thousand, thus the possible collaborations increased by a factor of 169, whereas the actual collaborations increased by only a factor of 21.

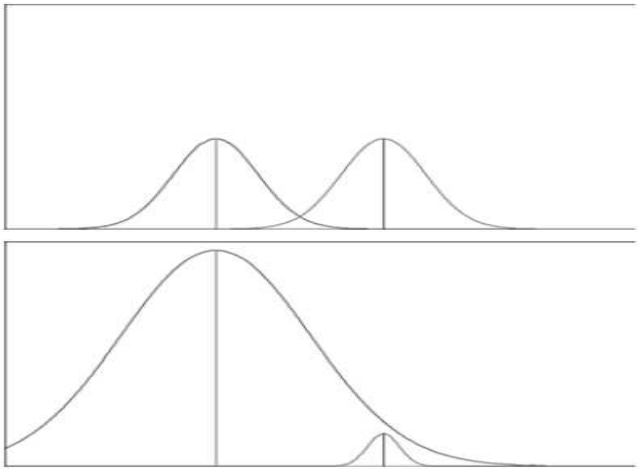

Figure 1.2. Schematic of the effect of large class skew on a model's ability to discriminate between classes. In first case (top), the two distributions are easily distinguished. In the second case (bottom), large class skew makes the discrimination really difficult. Image taken from [49].

The problem of class skewness in supervised learning is well known in machine learning. The poor performance of a learning algorithm in this case results from both the variance in the models estimates and the imbalance in the class distribution. Even if a low proportion of negative instances have the predictor value similar to the positive instances, the model will end up with a large raw number of false positives. We borrowed the following schematic explanation (see Figure 1.2) from [49]. For a hypothetical dataset, let us consider a predictor s measured on the instance pairs. Also assume that the values of s are drawn from a normal distribution with different means for positive (linked) and negative (not-linked) object pairs. In presence of large class skew, the entirety of the positive class distribution is “swallowed” by the tail of the negative class, as shown in Figure 1.2.

To cope with class skewness, existing research suggests several different approaches, such as altering the training sample by up-sampling or down-sampling [12], altering the learning method by making the process active [16] or cost-sensitive [28], and also more generally by treating the classifier score with different thresholds [48]. For kernel based classification, there exist some specific methods [63, 57] to cope with this problem. In general, learning from imbalanced datasets is a very important research consideration and we like to refer the reader to [60], which has a good discussion of various techniques to solve this.

The second challenge in supervised link prediction is model calibration [42], which is somewhat related to the class imbalance problem. However, model calibration is worth mentioning in its own merit because in the application domain of link prediction, calibrating the model is sometimes much more crucial than finding the right algorithm to build the classification model. Model calibration is the process to find the function that transforms the output score value of the model to a label. By varying (or biasing) the function we can control the ratio of false positive error and false negative error. In many application domains of link prediction, such as for detecting social network links in a terrorist network, the cost of missing a true link could be a catastrophic. One the other hand, in online social networks, recommending (predicting) a wrong link could be considered a more serious mistake than missing a true link. Based on these, the system designer needs to calibrate the model carefully. For some classifiers, calibration is very easy as the model predicts a score which can be thresholded to convert to a +1/-1 decision. For others, it may requires some alteration in the output of the model.

Another problem of link prediction is the training cost in terms of time complexity. Most of the social networks are large and also due to the class imbalances, a model's training dataset needs to consists of a large number of samples so that the rare cases [60] of the positive class are represented in the model. In such a scenario, classification cost may also become a consideration while choosing the model. For instance, running an SVM with millions of training instances could be quite costly in terms of time and resources, whereas Bayesian classification is comparably much cheaper.

Another important model consideration is the availability of dynamic updating options for the model. This is important for social networks because they are changing constantly and a trade off between completely rebuilding and updating the model may be worth considering. Recently, some models have been proposed that consider dynamic updates explicitly.

Above we also discussed how vertex attributes can be used for the task of link prediction. In supervised classification of link prediction, this is sometimes tricky because an instance in the training data represents a pair of vertices, rather than a single vertex. If the proposed model provides some options to map vertex attributes to pair attributes smoothly, that also makes the model an excellent choice for the link prediction task. Below we discuss a couple of supervised models that address some of the above limitations more explicitly.

3.2.1 Chance-Constrained with Second-Order Cone Programming. To explicitly handle imbalanced datasets, Doppa et. al. [15] proposed a second order cone programming (SOCP) formulation for the link prediction task. SOCP can be solved efficiently with methods for semi-definite programs, such as interior point methods. The complexity of SOCP is moderately higher than linear programs but they can be solved using general purpose SOCP solvers. The authors discussed two algorithms, named CBSOCP (Cluster-based SOCP formulation) and LBSOCP (Specified lower-bound SOCP).

In CBSOCP, the class conditional densities of positive and negative points are modeled as mixture models with component distribution having spherical covariances. If k_1 and k_2 denotes the number of components in the mixtures models for the positive and negative class, CBSOCP first finds k_1 positive clusters and k_2 negative clusters by estimating the second order moment (μ, σ^2) of all the clusters. Given these positive and negative clusters, it obtains a discriminating hyperplane $(w^T x - b = 0)$, like in SVM, that separates the positive and negative clusters. The following two chance-constraints are used.

$$\begin{array}{l} Pr(w^T X_i - b \geq 1) \geq \eta_1 : \forall i \in \{1 \dots k_1\} \\ Pr(w^T X_j - b \leq -1) \geq \eta_2 : \forall j \in \{1 \dots k_2\} \end{array}$$

Here X_i and X_j are random variables corresponding to the components of the mixture models for positive and negative classes, and η_1 and η_2 are the lower bound of the classification accuracy of these two classes. The chance-constraints can be replaced by deterministic constraints by using multinomial Chevyshev inequality (also known as Chevishev-Cantelli inequality) as below:

$$\begin{array}{lll} \min_{w,b,\xi_i} & \sum_{i=1}^{k} \xi_i & \\ \text{s.t.} & y_i(w^T \mu_i - b) \geq 1 - \xi_i + \kappa_1 \sigma_i W, & \forall i = 1, \dots, k_1 \\ & y_j(w^T \mu_j - b) \geq 1 - \xi_j + \kappa_2 \sigma_j W, & \forall j = 1, \dots, k_2 \\ & \xi_i \geq 0 & \forall i = 1, \dots, k_1 + k_2 \\ & W \geq \|w\|_2 & \end{array}$$

where, $k = k_1 + k_2$, $\kappa_i = \sqrt{\frac{\eta_i}{1-\eta_i}}$ and W is a user-defined parameter which lower bounds the margin between the two classes. By solving the above SOCP problem, we get the optimum values of w and b, and a new data point x can be classified as $sign(w^T x - b)$.

LBSOCP imposes lower bounds on the desired accuracy in each class, thus controlling the false positive and false-negative rates. It considers the following formulation:

$$\begin{array}{ll} min_{w,b} & \frac{1}{2}\|w\|_2 \\ \text{s.t.} & Pr(X \in \mathcal{H}_2) \leq 1 - \eta_1 \\ & Pr(X \in \mathcal{H}_1) \leq 1 - \eta_2 \\ & X_1 \sim (\mu_1, \Sigma_1), X_2 \sim (\mu_2, \Sigma_2) \end{array}$$

where $\mathcal{H}_1$ and $\mathcal{H}_2$ denote the positive and negative half-spaces, respectively. The chance constraints specify that the probability that false-negative and false-positive rate should not exceed $1 - \eta_1$ and $1 - \eta_2$, respectively. Like before, using Chevyshev inequality, this can be formulated using a SOCP problem as below:

$$\begin{array}{ll} \min_{w,b,t} & t \\ \text{s.t.} & t \geq \|w\|_2 \\ & w^T\mu_1 - b \geq 1 + \kappa_1\|C_1^T w\|_2 \\ & b - w^T\mu_2 \geq 1 + \kappa_2\|C_2^T w\|_2 \end{array}$$

where, $\kappa_i = \sqrt{\frac{\eta_i}{1-\eta_i}}$, and C_1 and C_2 are square matrices such that $\Sigma_1 = C_1C_1^T$ and $\Sigma_2 = C_2C_2^T$. Note that such matrices exist since Σ_1 and Σ_2 are positive semi-definite. After solving this above problem, the optimal value of w and b can be obtained which can be used to classify new data point x as $sign(w^T x - b)$.

The strength of above two SOCP formulations is that they allow an explicit mechanism to control the false positive and false negative in link prediction. So, they are well suited for the case of imbalanced classification. Also they are scalable. Authors in [15] show that they perform significantly better than a traditional SVM classifier.

3.2.2 Pairwise Kernel Approach. In Section 3.1, we discussed the pairwise kernel technique for automatically converting the vertex attributes to pair attributes; this technique has been used to build kernel based classifiers for link prediction [30]. The main objective is to build a pair-wise classifier [43, 30]. Standard binary classification problem aims to learn a function $f : V \to \{+1, -1\}$, where V indicates the set of all possible instances. On the other hand, in the (binary) pairwise classification, the goal is to learn a function $f : V^{(1)} \times V^{(2)} \to \{+1, -1\}$, where $V^{(1)}$ and $V^{(2)}$ are two sets of possible instances. There also exists a matrix $\mathbf{F}$ of size $|V^{(1)}| \times |V^{(2)}|$ whose elements are +1 (link exist) and -1 (link does not exist). For link prediction task, $V^{(1)} = V^{(2)} = V$ the vertex set of the social network $G(V, E)$ and the matrix $\mathbf{F}$ is just the adjacency matrix of the graph G. For pairwise classification using kernels, we also have two positive semi-definite kernel matrices, $\mathbf{K}^{(1)}$ and $\mathbf{K}^{(2)}$ for $V^{(1)}$ and $V^{(2)}$ respectively. For link prediction task, $\mathbf{K}^{(1)} = \mathbf{K}^{(2)} = \mathbf{K}$. $\mathbf{K}$ is a kernel matrix of size $|V| \times |V|$, in which each entry denotes the similarity between the corresponding pair of nodes in the social network. To compute $\mathbf{K}$, any function that maps a pair of nodes to a real number can be used as long as $\mathbf{K}$ remains semi-definite.

Generally, the assumption that drives pair-wise classification is that the similarity score between a pair of instances (an instance itself is a pair) is higher if the first elements from both the instances are similar and also the second elements from both the pairs are similar. So, if $v_{i_1}^{(1)}, v_{j_1}^{(1)} \in V^{(1)}$, $v_{i_2}^{(2)}, v_{j_2}^{(2)} \in V^{(2)}$ and $(v_{i_1}^{(1)}, v_{i_2}^{(2)})$ and $(v_{j_1}^{(1)}, v_{j_2}^{(2)})$ are similar, we expect $v_{i_1}^{(1)}$ and $v_{j_1}^{(1)}$ are similar and $v_{i_2}^{(2)}$ and $v_{j_2}^{(2)}$ are similar. To model this expectation in the kernel framework, [43] proposed to consider the pairwise similarity to be the product of two instance-wise similarities, i.e.,

$$k_{\otimes}((v_{i_1}^{(1)}, v_{i_2}^{(2)}), (v_{j_1}^{(1)}, v_{j_2}^{(2)})) = [\mathbf{K}^{(1)}]_{i_1,j_1}\ [\mathbf{K}^{(2)}]_{i_2,j_2}$$

Since the product of Mercer kernels is also a Mercer kernel [51], the above similarity measure is also a Mercer kernel if the element-wise kernels are Mercer kernels. Using the above formulation, the kernel for the pair-wise classifier is just the Kronecker product of the instance kernel matrices: $\mathbf{K}_{\otimes} = \mathbf{K}^{(1)} \otimes \mathbf{K}^{(2)}$. This pairwise kernel matrix can be interpreted as a weighted adjacency matrix of the Kronecker product graph [26] of the two graphs whose weighted adjacency matrices are the instance-wise kernel matrices. [30] named it as *Kronecker Kernel* and proposed an alternative that is based on Cartesian product graph, hence named *Cartesian kernel*. The difference between them is just the way how these two product graphs are formed. In case of Kronecker product, if $(v_{i_1}^{(1)}, v_{i_2}^{(2)})$ and $(v_{j_1}^{(1)}, v_{j_2}^{(2)})$ are node pairs in the product graph, there exist a link between the pair $v_{i_1}^{(1)}$ and $v_{j_1}^{(1)}$ and also a link between the pair $v_{i_2}^{(2)}$ and $v_{j_2}^{(2)}$. On the other hand, for the case of Cartesian product a link between these two pairs in the product graph exists if and only if $v_{i_1}^{(1)} = v_{j_1}^{(1)}$ in the first graph and there is a link between $v_{i_2}^{(2)}$ and $v_{j_2}^{(2)}$ in the second graph or a link exists between $v_{i_1}^{(1)}$and $v_{j_1}^{(1)}$ in the first graph and $v_{i_2}^{(2)} = v_{j_2}^{(2)}$ in the second graph. Based on this, Cartesian kernel is defined as below:

$$k_{\otimes}((v_{i_1}^{(1)}, v_{i_2}^{(2)}), (v_{j_1}^{(1)}, v_{j_2}^{(2)})) = \delta(i_2 = j_2)[\mathbf{K}^{(1)}]_{i_1,j_1} + \delta(i_1 = j_1)[\mathbf{K}^{(2)}]_{i_2,j_2}$$

For link prediction on undirected graphs, both the instance matrices are the same and also the element pairs in an instance are exchangeable.

The Kronecker kernel can be made symmetric as below:

$$k_{\otimes}^{\text{SYM}}((v_{i_1}, v_{i_2}), (v_{j_1}, v_{j_2})) = [\mathbf{K}]_{i_1,j_1} \ [\mathbf{K}]_{i_2,j_2} + [\mathbf{K}]_{i_1,j_2} \ [\mathbf{K}]_{i_2,j_1}$$

And for Cartesian kernel it is as shown below:

$$\begin{aligned} k_{\otimes}^{\text{SYM}}((v_{i_1}, v_{i_2}), (v_{j_1}, v_{j_2})) = \ & \delta(i_2 = j_2)[\mathbf{K}]_{i_1,j_1} + \delta(i_1 = j_1)[\mathbf{K}]_{i_2,j_2} \\ & + \delta(i_2 = j_1)[\mathbf{K}]_{i_1,j_2} + \delta(i_1 = j_2)[\mathbf{K}]_{i_2,j_1} \end{aligned}$$

The advantage of Cartesian kernel over the Kronecker kernel is that it has many more zero entries (an entry is zero if the two pairs do not share at least one instance). So, the training time is much faster. [30] showed via experiments that its performance is comparable with respect to the Kronecker kernel.

4. Bayesian Probabilistic Models

In this section, we will discuss supervised models that use Bayesian concepts. The main idea here is to obtain a posterior probability that denotes the chance of co-occurrence of the vertex pairs we are interested in. An advantage of such model is that the score itself can be used as a feature in classification, as we discussed in section 3.2. Contenders in this category are the algorithms proposed by Wang, Satulur and Parthasarathy [58] and by Kashima and Abe [29]. The former uses a MRF based local probabilistic model and the later uses a parameterized probabilistic model. [58] also takes the output from the probabilistic method and uses it as a feature in a subsequent steps that employs several other features (Katz, vertex attribute similarity) to predict a binary value.

4.1 Link Prediction by Local Probabilistic Model

Wang et. al. [58] proposed a local probabilistic model for link prediction that uses Markov Random Field (MRF), an undirected graphical model. To predict the link between a pair of nodes x and y, it introduces the concept of *central neighborhood set*, which consists of other nodes that appear in the local neighborhood of x or y. Let $\{w, x, y, z\}$ be *one* such set, then the main objective of this model is to compute the joint probability $P(\{w, x, y, z\})$, which represents the probability of co-occurrence of the objects in this set. This probability can be marginalized (in this example, over all possible w and z) to find the co-occurrence probability between x and y. There can be many such central neighborhood sets (of varying size) for the pair x and y, which makes learning the marginal probability $(p(x, y))$ tricky. The authors exploited MRFs to solve the learning problem; their approach has three steps, as described below.

The first step is to find a collection of central neighborhood sets. Given two nodes x and y, their central neighborhood sets can be found in many ways. The most natural way is to find a shortest path between x and y and then all the nodes along this path can belong to one central neighborhood set. If there exist many shortest paths of the same length, all of them can be included in the collection. Finding shortest path of arbitrary length can be costly for very large graphs. So in [58] the authors only considered shortest paths up to length 4. Let us assume that the set Q contains all the objects that are present in any of the central neighborhood set.

The second step is to obtain the training data for the MRF model, which is taken from the event log of the social network. Typically a social network is formed by a chronological set of events where two or more actors in the network participate. In case of co-authorship network, co-authoring an article by two or more persons in the network is an event. Given an event-list, [58] forms a transaction dataset, where each transaction includes the set of actors participates in that event. On this dataset, they perform a variation of itemset mining, named non-derivable itemset mining, which outputs all the non-redundant itemsets (along with their frequencies) in the transaction data. This collection is further refined to include only those itemsets that contain only the objects belonging to the set Q. Assume this collection is the set $\mathcal{V}_Q$.

In the final step, an MRF model (say, M) is trained from the training data. This training process is translated to a maximum entropy optimization problem which is solved by *iterative scaling* algorithm. If $P_M(Q)$ is the probability distribution over the power set of Q, we have $\sum_{q \in \wp(Q)} P_M(q) = 1$, where $\wp(Q)$ denotes the power-set of Q. Each itemset along with its associated count in the set $\mathcal{V}_Q$ imposes a constraint on this distribution by specifying a value for that specific subset (of Q). Together, all these counts restrict the distribution to a feasible set of probability distributions, say $\mathcal{P}$. Since, the itemset counts come from empirical data, $\mathcal{P}$ is non-empty. But, the set of constraints coming through $\mathcal{V}_Q$ typically under-constrains the target distribution, for which we adopt the maximum entropy principle so that a unique (and unbiased) estimate of $P_M(Q)$ can be obtained from the feasible distribution set $\mathcal{P}$. Thus, we are trying to solve the following optimization problem,

$$P_M(Q) = \arg\max_{p \in \mathcal{P}} \; H(p)$$

where, $H(p) = -\sum_x p(x) \; \log p(x)$. The optimization problem is feasible and a unique target distribution exists only if the constraints are consistent (in this case, the frequency constraints are consistent since they are taken from the itemset support value). The solution has the following product form:

$$P_M(Q) = \mu_0 \prod_{j:V_j \in \mathcal{V}_Q} \mu_j^{I(\text{constraint } V_j \text{ satisfies})}$$

Here, $\mu_j : j \in \{1 \ldots |\mathcal{V}_Q|\}$ are parameters associated with each constraint, I is an indicator function which ensures that a constraint is considered in the model only if it is satisfied and μ_0 is a normalizing constant to ensure $\sum_{q \in \wp(Q)} P_M(q) = 1$. The value of the parameters can be obtained by an iterative scaling algorithm; for details, see [44].

Once the model $P_M(Q)$ is built, one can use inference to estimate the joint probability between the vertex x and y. The advantage of a local mode is that the number of variables in the set $\mathcal{V}_Q$ is small, so exact inference is feasible. [58] used the Junction Tree algorithm as an inference mechanism.

4.2 Network Evolution based Probabilistic Model

Kashima et. al. [29] proposed an interesting probabilistic model of network evolution which can be used for link prediction. The connection between these two problems is emphasized in [36] that we quote here: "*a network model is useful to the extent that it can support meaningful inference from observed network data*". Motivated from this statement, the authors in [29] showed that by having tunable parameters in an evolution model naturally gives rise to a learning algorithm for link prediction. First we discuss the network evolution model and later show how they use the model to perform link prediction.

The proposed evolution model considers only the topological (structural) properties of the network. For a graph $G(V, \phi)$, where V is the set of nodes and $\phi : V \times V \to [0, 1]$ is an edge label function, $\phi(x, y)$ denotes the probability that an edge exists between node x and y in G. In particular, $\phi(x, y) = 1$ denotes that an edge exists and $\phi(x, y) = 0$ denotes that an edge does not exist. $\phi^{(t)}$ denotes the edge label function at time t, which changes over time; further, the model is Markovian, i.e., $\phi^{(t+1)}$ depends only on $\phi^{(t)}$. In this model V remains fixed. The model evolves over the time as below: An edge label is copied from node l to node m randomly with probability w_{lm}. First, the model decides on l and m, then chooses an edge label uniformly from one of l's $|V| - 1$ edge labels (excluding $\phi(l, m)$) to copy as m's edge label. The model satisfies the following probability constraints.

$$\sum_{lm} w_{lm} = 1,\ w_{lm} > 0, w_{ll} = 0$$

The above idea closely resembles the transitivity property of social network – a friend of a friend becomes a friend. Through the edge label copying process, l can become friend of one of m's friend. The learning task in the above model is to compute the weights w_{ij} and the edge labels $\phi^{(t+1)}$ given the edge label $\phi^{(t)}$ from training dataset.

There are two possible ways for $\phi^{(t)}(i,j)$ to assume a particular edge label. The first is that node k copied one of its edge label to either node i or to node j. The other is that, copying happened elsewhere and $\phi^{(t+1)}(i,j) = \phi^{(t)}$. Following this, we have:

$$\begin{aligned} \phi^{(t+1)}(i,j) = \quad & \tfrac{1}{|V|-1} \sum_{k \neq i,j} w_{kj} \phi^{(t)}(k,i)\ I\left(\phi^{(t)}(k,j) = 1\right) \\ & + \tfrac{1}{|V|-1} \sum_{k \neq i,j} w_{ki} \phi^{(t)}(k,j)\ I\left(\phi^{(t)}(k,i) = 1\right) \\ & + \left(1 - \tfrac{1}{|V|-1} \sum_{k \neq i,j} (w_{kj} + w_{ki})\right) \phi^{(t)}(i,j) \end{aligned} \tag{1.8}$$

Note that, for the case when the copy happens if k copies its label to node i, then k should already have an edge with j and if k copies its label to node j, it should already have an edge with i. This requirement is manifested by the indicator function I, which assumes a value 0 if the condition inside the parenthesis is not satisfied. By iterative application of this equation on the edge labels, the network structure evolves over time.

For the task of link prediction, the model considers that the current network is in an stationary state, i.e., $\phi^{(\infty)}(k,i) = \phi^{(t+1)}(k,i) = \phi^{(t)}(k,i)$; by plugging this assumption in Equation 1.8, we obtain the following equation

$$\phi^{(\infty)}(i,j) = \frac{\sum_{k \neq i,j} \left(w_{kj} \phi^{(\infty)}(k,i) + w_{ki} \phi^{(\infty)}(k,j)\right)}{\sum_{k \neq i,j} (w_{kj} + w_{ki})} \tag{1.9}$$

The log-likelihood for the edge label $\phi(i,j)$ can be written as

$$\begin{aligned} L_{ij} \quad & = \phi^{(\infty)}(i,j) \log \tfrac{\sum_{k \neq i,j} \left(w_{kj} \phi^{(\infty)}(k,i) + w_{ki} \phi^{(\infty)}(k,j)\right)}{\sum_{k \neq i,j} \left(w_{kj} + w_{ki}\right)} \\ & \left(1 - \phi^{(\infty)}(i,j)\right) \log \left(1 - \tfrac{\sum_{k \neq i,j} \left(w_{kj} \phi^{(\infty)}(k,i) + w_{ki} \phi^{(\infty)}(k,j)\right)}{\sum_{k \neq i,j} \left(w_{kj} + w_{ki}\right)}\right) \end{aligned} \tag{1.10}$$

Total log-likelihood for the known edge labels is defined as:

$$L(W) = \sum_{(i,j)\in E^{\text{train}}} L_{ij} \tag{1.11}$$

Now, the parameter estimation process is mapped to the following constrained optimization problem:

$$\text{Maximize}_{w,\phi^{(\infty)}(i,j) \text{ for } (i,j)\in E^{\text{train}}} \;\; L(W)$$
$$\text{s. t.}$$
$$\phi^{(\infty)}(i,j) = \frac{\sum\limits_{k\neq i,j} \Big(w_{kj}\phi^{(\infty)}(k,i) + w_{ki}\phi^{(\infty)}(k,j) \Big)}{\sum\limits_{k\neq i,j} (w_{kj} + w_{ki})}, \forall (i,j) \in$$
$$E^{\text{train}}, \quad \text{and} \sum_{l,m} w_{lm} = 1, \quad w_{lm} \geq 0$$

The above optimization problem can be solved by an Expectation Maximization type transductive learning; for details, see [29].

The benefit of this model is that it is very generic and can be applied to any social network. Further, the EM based learning yields an efficient algorithm. However, the performance of the algorithm entirely depends on the degree to which the network agree to the proposed graph evolution model.

4.3 Hierarchical Probabilistic Model

Clauset et. al. [14] proposed a probabilistic model which considers the hierarchical organization in the network, where vertices divide into groups that further subdivide into groups of groups and so forth over multiple scales. The model infers hierarchical structure from network data and can be used for prediction of missing links. It is proposed as a probabilistic model for hierarchical random graphs. The learning task is to use the observed network data to fit the most likely hierarchical structure through statistical inference – a combination of the maximum likelihood approach and a Monte Carlo sampling algorithm.

Let G be a graph with n vertices. A *dendogram* D is a binary tree with n leaves corresponding to the vertices of G. Each of the $n-1$ internal nodes of D corresponds to the group of vertices that are descended from it. A probability p_r is associated with each internal node r. Then, given two vertices i,j of G, the probability p_{ij} that they are connected by an edge is $p_{ij} = p_r$ where r is the lowest common ancestor in D. The combination, $(D, \{p_r\})$ of the dendogram and the set of probabilities then defines a *hierarchical random graph.*

The learning task is to find the hierarchical random graph or graphs that best fits the observed real world network data. Assuming all hierarchical graphs are *a priori* equally likely, the probability that a given model, $(D, \{p_r\})$ is the correct explanation of the data is, by Bayes theorem, proportional to the posterior probability or likelihood, $\mathcal{L}$ with which the model generates the observed network. The goal is to maximize $\mathcal{L}$.

Let E_r be the number of edges in G whose endpoints have r as their lowest common ancestor in D, and let L_r and R_r, respectively, be the numbers of leaves in the left and right subtrees rooted at r. Then, the likelihood of the hierarchical random graph is $\mathcal{L}(D, \{p_r\}) = \prod_{r \in D} p_r^{E_r}(1-p_r)^{L_r R_r - E_r}$, with the convention that $0^0 = 1$. If we fix the dendogram D, it is easy to find the probabilities $\{\bar{p}_r\}$ that maximize $\mathcal{L}(D, \{p_r\})$, which is:

$$\bar{p}_r = \frac{E_r}{L_r R_r} \tag{1.12}$$

the fraction of the potential edges between the two subtrees of r that actually appear in the graph G. The logarithm of the likelihood is:

$$\log \mathcal{L}(D) = -\sum_{r \in D} L_r R_r h(\bar{p}_r) \tag{1.13}$$

where, $h(p) = -p \log p - (1-p) \log(1-p)$. Note that each term $-L_r R_r h(\bar{p}_r)$ is maximized when $\bar{p}_r$ is close to 0 or close to 1. In other words, high-likelihood dendograms are those that partition the vertices into groups between which connections are either very common or very rare.

The choice among the dendograms are made by a Markov chain Monte Carlo sampling method with probabilities proportional to their likelihood. To create the Markov chain, the method first creates a set of transitions between possible dendograms through rearrangement. For rearrangement, the method chooses an internal node of a dendogram and then chooses uniformly among various configuration of the subtree at that node; for details, see [14]. Once the transition criteria is known the sampling process initiates a random walk. A new rearrangement is accepted according to the Metropolis-Hastings sampling rule, i.e., for a transition from a dendogram D to another rearranged dendogram D', the transition is accepted if $\Delta \log \mathcal{L} = \log \mathcal{L}(D') - \log \mathcal{L}(D)$ is nonnegative, otherwise it is accepted with a probability $\mathcal{L}(D')/\mathcal{L}(D)$. Authors proved that the random walk is ergodic and at stationary distribution the dendogram are sampled according to their probability of likelihood.

For the task of link prediction, a set of sample dendograms are obtained at regular intervals once the MCMC random walk reaches an equilibrium. Then, for the pair of vertices x and y for which no connection exists, the model computes a mean probability p_{xy} that they are connected by averaging over the corresponding probability p_{xy} in each of the sampled dendograms. For a binary decision, a model calibration can be made through a calibration dataset. The unique nature of the hierarchical random graph model is that it allows an hierarchy in the model. Also, it allows to sample over the set of hierarchical structures to obtain a consensus probability. On the downside, it may not be that accurate unless MCMC converges to the stationary distribution in a reasonable number of steps. Also for large graphs the entire process could be very costly.

5. Probabilistic Relational Models

In earlier sections, we discussed that the vertex attributes play a significant role in link prediction task. We also showed how different link prediction methods try to incorporate the vertex attributes in the prediction model to obtain better performance. However, in most of the cases, these approaches are not generic, and thus, are not applicable in all possible scenarios. Probabilistic Relational model (PRM) is a concrete modeling tool that provides a systematic way to incorporate both vertex and edge attributes to model the joint probability distribution of a set of entities and the links that associate them. The benefit of a PRM is that it considers the object-relational nature of structured data by capturing probabilistic interactions between entities and the links themselves. So, it is better than a flat model which discards such relational information. There are two pioneering approach of PRM, one based on Bayesian networks, which consider the relation links to be directed [21], and the other based on relational Markov networks, which consider the relation links to be undirected [53]. Though both are suitable for link prediction task, for most networks an undirected model seems to be more appropriate due to its flexibility.

As an example consider the link prediction problem in a co-authorship network. The only entities that other (non-relational) models consider is the *person*. However, in PRM we can mix heterogeneous entities in the model. So it is entirely possible to include other relevant objects in this model, such as *article*, *conferenceVenue*, and *institution*. Similar to a database schema, each of these objects can have attributes. For example, a person may have attributes like *name*, *affiliationInstitute*, *status* (whether (s)he is a student, an employee or a professor); an article may have *publicationYear*, *conferenceVenue*; an institute may have *location*, and a conference venue may have attributes like *ResearchKeywords* and so on. Then there can be relational links between these entities. Two person can be related by an *advisor/advisee* relationship. A person can be related to a paper by an *author* relationship. A paper can be related to a conference venue by *publish* relationship. In this way, the model can include a complete relational schema similar to an object relational database.

PRM was originally designed for the attribute prediction problem in relational data. For link prediction task, it was extended [21, 53] so that the links are first-class citizens in the model, so additional objects, named *link objects* are added in the relational schema. Any link object, l, is associated with a tuple of entity objects $(o_1, \ldots o_k)$ that participate in the relation (for most of the cases, links will be between a tuple of two entity objects). Following the example in the previous paragraph, one of the link object can be *advisor/advisee* object that relates two persons. The model also allows the link objects to have attributes. Now, consider a object named *potentialLink* that relates two persons. It has a binary attribute named *exist* which is *true* if there exists a link between the associated objects, and false otherwise. The link prediction task now reduces to the problem of predicting the existence attribute of these link objects.

In the training step of the model, a single probabilistic model is defined over the entire link graph, including both object labels and links between the objects. The model parameters are trained discriminatively, to maximize the probability of the (object) and the link labels given the known attributes. The learned model is then applied using probabilistic inference, to predict and classify links using observed attributes and links.

5.1 Relational Bayesian Network

Relational Bayesian Network (RBN) is the relational counterpart of a Bayesian network (BN). Hence, the model graph of RBN $G_M^D = (V_M, E_M)$ is a directed acyclic graph with a set of conditional probability distribution (CPD) to represent a joint distribution over the attributes of the item types. Each CPD corresponding to an attribute X represents $P(X|pa(X))$, where $pa(X)$ are the parents of X in the network. In RBN, like BN, the joint probability distribution can be factorized according to the dependencies in the acyclic graph structure. RBN has closed-form parameter estimation techniques, which make the learning of the model parameters very efficient. The learning process is almost identical to BN. As for inference, RBN adopts belief propagation [21], which could perform poorly in many cases.

5.2 Relational Markov Network

Relational Markov Network (RMN) is the relational counterpart of undirected graphical models or Markov Networks [45]. Let V denotes a set of discrete random variables, and v is an instantiation of the variables in V. A Markov network for V defines a joint distribution over V through an undirected dependency network and a set of parameters. For a graph G, if $C(G)$ is the set of cliques (not necessarily maximal), the Markov network defines the distribution $p(v) = \frac{1}{Z}\prod_{c \in C(G)} \phi_c(v_c)$, where Z is the standard normalizing factor, v_c is the vertex set of the clique c, and ϕ_c is a clique potential function. RMN specifies the cliques using the notion of a *relational clique template*, which specifies tuples of variables in the instantiation using a relational query language.

Given a particular instantiation $\mathcal{I}$ of the schema, the RMN $\mathcal{M}$ produces an *unrolled* Markov network over the attributes of entities in $\mathcal{I}$ (see [55] for details). The cliques in the unrolled network are determined by the clique template C. There exists one clique for each $c \in C(\mathcal{I})$, and all of these cliques are associated with the same clique potential ϕ_C. Tasker et. al. [54] show how the parameters of a RMN over a fixed set of cliques can be learned from data. In a large network with a lot of relational attributes, the network is typically large, so exact inference is typically infeasible. So, like RBN, RMN also uses belief propagation for inference.

Besides the above two, their exists several other relational models that can be used for link prediction. These are Bayesian relational models such as DAPER (Directed Acyclic Probabilistic Entity Relationship) [24], relational dependency network [23], parametric hierarchical Bayesian relational model [62], non-parametric hierarchical Bayesian relational model [61] and stochastic relational model [64]. For details on these, we encourage the readers to read the respective references.

6. Linear Algebraic Methods

Kunegis et. al. [33] proposed a very general method that generalizes several graph kernels and dimensionality reduction methods to solve the link prediction problem. This method is unique in the sense that it is the only method that proposes to learn a function F which works directly on the graph adjacency or the graph Laplacian matrix.

Let $\mathbf{A}$ and $\mathbf{B}$ be two adjacency matrices of the training and test set for the link prediction. We assume that they have the same vertex set. Now, consider a spectral transformation function F that maps $\mathbf{A}$ to $\mathbf{B}$ with minimal error given by the solution to the following optimization problem:

$$\begin{aligned} \min_F \quad & \| \, F(\mathbf{A}) - \mathbf{B} \|_F \\ s.t. \quad & F \in \mathcal{S} \end{aligned} \tag{1.14}$$

where $\| \, . \, \|_F$ *denotes the Frobenius norm.* Here, the constrain ensures that the function F belongs to the family of spectral transformation functions ($\mathcal{S}$). Given a symmetric matrix $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, for such an F, we have $F(\mathbf{A}) = \mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^T$, where $F(\mathbf{A})$ applies the corresponding function on reals to each eigenvalue separately. Note that the above formulation of link prediction is a form of transductive learning as the entire test data is available to learn the model parameters.

The optimization problem in (1.14) can be solved by computing the eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ and using the fact that the Frobenius norm is invariant under multiplication by an orthogonal matrix

$$\begin{aligned} & \| \, F(\mathbf{A}) - \mathbf{B} \|_F \\ = \; & \| \, \mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^T - \mathbf{B} \|_F \\ = \; & \| \, F(\mathbf{\Lambda}) - \mathbf{U}^T\mathbf{B}\mathbf{U} \|_F \end{aligned} \tag{1.15}$$

Since, the off-diagonal entries in the expression (1.15) are not dependent on the function F, the desired optimization function on the matrix can be transformed into an optimization function on real numbers as below:

$$\min_f \quad \sum_i \left(f(\mathbf{\Lambda}_{ii}) - \mathbf{U}^T_{.i}\mathbf{B}\mathbf{U}_{.i} \right)^2 \tag{1.16}$$

So, the link prediction problem thus reduces to a one-dimensional least square curve fitting problem.

Now, the above general method can be used to fit many possible spectral transformation functions. In particular, we are looking for a function F that accepts a matrix and return another matrix which is suitable for link prediction, i.e., the entries in the returned matrix should encode the similarity between the corresponding vertex pairs. There are many graph kernels which can be used for the function F.

Exponential Kernel. For an adjacency matrix of an unweighted graph, $\mathbf{A}$, the powers, $\mathbf{A}^n$ denotes the number of paths of length n connecting all node pairs. On the basis that the nodes connected by many paths should be consider nearer to each other than nodes connected by few paths, a function F for link prediction can be as below:

$$F_P(\mathbf{A}) = \sum_{i=0}^{d} \alpha_i \mathbf{A}^i \tag{1.17}$$

The constant α_i should be decreasing as α grows larger to penalize longer paths. Now an exponential kernel can be expressed as below which models the above path counting.

$$exp(\alpha\mathbf{A}) = \sum_{i=0}^{\infty} \frac{\alpha^i}{i!}\mathbf{A}^i \tag{1.18}$$

Von-Neumann Kernel. It is defined similar to the exponential kernel

$$(\mathbf{I} - \alpha\mathbf{A})^{-1} = \sum_{i=0}^{\infty} \alpha^i\mathbf{A}^i \tag{1.19}$$

it also models a path counting kernel.

Laplacian kernels. The generic idea proposed in this method is not confined to use functions on adjacency matrix. In fact, one is also allowed to use functions that apply on the Laplacian matrix, $\mathbf{L}$ which is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal degree matrix. The normalized Laplacian matrix, $\mathcal{L}$ is defined as $\mathcal{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. While using Laplacian matrices, the entire formulation proposed in this method remains the same except that instead of an adjacency matrix we use a Laplacian matrix. Many graph kernels are defined on the Laplacian matrix. For example, by taking the Moore-Penrose pseudo-inverse of the Laplacian we can obtain the commute time kernel:

$$\begin{aligned} F_{COM}(\mathbf{L}) &= \mathbf{L}^{+} \\ F_{COM}(\mathcal{L}) &= \mathcal{L}^{+} \end{aligned}$$

by applying regualrization, we can obtain regularized commute time kernels:

$$\begin{aligned} F_{COMR}(\mathbf{L}) &= (\mathbf{I} + \alpha\mathbf{L})^{-1} \\ F_{COMR}(\mathcal{L}) &= (\mathbf{I} + \alpha\mathcal{L})^{-1} \end{aligned}$$

We can also obtain heat diffusion kernels as below:

$$\begin{aligned} f_{HEAT}(\mathbf{L}) &= exp(-\alpha\mathbf{L}) \\ f_{HEAT}(\mathcal{L}) &= exp(-\alpha\mathcal{L}) \end{aligned}$$

For some of the above functions, the corresponding one dimensional function on reals is shown in Table 1.1.

The advantage of this method is its genericity and simplicity. The number of parameters to learn in this model is much less compared to

Link Prediction Function	Real Function
$F_P(\mathbf{A}) = \sum_{i=0}^{d} \alpha_i \mathbf{A}^i$	$f(x) = \sum_{i=0}^{d} \alpha_i x^i$
$F_{EXP}(\mathbf{A}) = \exp(\alpha\mathbf{A})$	$f(x) = e^{\alpha x}$
$F_{NEU}(\mathbf{A}) = ((I) - \alpha\mathbf{A})^{-1}$	$f(x) = \frac{1}{1-\alpha x}$
$F_{COM}(\mathbf{L}) = \mathbf{L}^{+}$	$f(x) = x^{(} - 1)$ when $x > 0$, $f(x) = 0$, otherwise
$F_{COMR}(\mathbf{L}) = ((I) + \alpha\mathbf{L})^{-1}$	$f(x) = \frac{1}{1+\alpha x}$
$F_{HEAT}(\mathbf{L}) = (exp)(-\alpha\mathbf{L})$	$f(x) = e^{-\alpha x}$

Table 1.1. One dimensional link prediction functions

many other models that we discussed. On the downside, this model cannot incorporate vertex based attributes. Morevoer, The computational cost of this method mostly depends on the cost of eigen-decomposition of **A**, which could be costly for large matrices. However, efficient methods for this task are available [19].

7. Recent development and Future Works

In recent years, the works on link prediction has evolved over various aspects. One of the main aspects among these is to consider the time in the model, which can be named as time-aware link prediciton [56, 5]. Some of the algorithms that we discussed in this survey can be extended to consider the temporal attribute of a link. For example, algorithms that perform supervised learning by using a set of features can directly consider the temporal properties in the feature value calculation. For instance, while computing Jaccard coefficient between two nodes, one can redefine the similariy metric so that recent association is weighted more than the past associations. But, the approach is somewhat ad-hoc because the desired (or optimal) temporal weighting mechanism is not available and for different metrics different weighting may apply. In case of relational model, we can always include time in the relational schema just as an edge attribute. However, in the context of link prediction, the model needs to accord special treatment for the time attribute, so that progression of the time can be captured in the model properly instead of just matching the time values. Tylenda et. al. [56] showed that considering the time stamp of the previous interactions significantly improves the accuracy of the link prediction model. Ahmed et. al. [5] proposed an scalable solution to a slightly different problem from link prediction, where they find how links in the network vary over time. They use a temporally smoothed l_1-regularized logistic regression formalism to solve this problem. Techniques like these can be borrowed to perform time-aware link prediction in a more principled manner.

Another important concern is the scalability of the proposed solutions for link prediction. Social networks are large and many of the proposed solutions, specifically, the probabilistic methods that consider the entire graph in one model is too large for most of the inference mechanisms to handle. Technique such as kernel based methods are also not scalable, because it is practically impossible to obtain a kernel matrix for such a large graph data. Note that a kernel matrix in this case is not of size $|V| \times |V|$, but of size $|V^2| \times |V^2|$. For most of the real-life social networks, $|V|$ is in the range of several millions to even billions, for which this approach is just not feasible. Even for the methods that perform feature based classification, computation of some of the features are very costly. Specially features such as Katz and rooted pagerank may require significant time to compute their values for a large number of vertex pairs. So, an approximate solution for these features can be a good research topic (see for example [52]).

Game theoretic concepts are very prominent in modeling various social problems, however these have surprisingly been ignored in the link prediction task. The closest work is the local connection game proposed by Fabrikant et. al.[17]. In this game the edges have constant cost and the players try to minimize their cost plus the sum of distances to all other pairs. However, such a local connection model may not be practical in the social network domain because the utility function partly considers a global objective which minimizes the distances to all pairs. So, it may not yield good result for the link prediciton task. An interesting alteration to this model that considers the utility of a person in the network from more subjective viewpoint is worth considering.

Acknowledgments

This work was supported in part by NSF Grants EMT-0829835 and EIA-0103708, and NIH Grant 1R01EB0080161-01A1.

References

[1] Acar, Evrim, and Dunlavy, Daniel M., Kolda, Tamara G. (2009). *Link Prediction on Evolving Data Using Matrix and Tensor Factorizations.* In Proceedings of the Workshop on Large Scale Data Mining Theory and Applications. ICDM Workshops:262-269

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

[3] Adafre, Sisay F., and Rijke, Maarten de. (2005). *Discovering missing links in Wikipedia.* LINK-KDD '05: Proceedings of the Third

International Workshop on Link Discovery.

[4] Ahmed, Elmagarmid, and Ipeirotis, Panagiotis G., and Verykios, Vassilios. (2007) *Duplicate Record Detection: A Survey.* In IEEE Transactions on Knowledge and Data Engineering 19 (1):116

[5] Ahmed, Amr, and Xing, Eric P. (2009). *Recovering time-varying network of dependencies in Social and biological studies.* PNAS 106(29):11878-11883.

[6] Airodi, Edoardo M., and Blei, David M., and Xing, Eric P., and Fienberg, Stephen E. (2006). *Mixed Membership stochastic block models for relational data, with applications to protein-protein interactions.* Proceedings of Ineterational Biometric Society-ENAR Annual Meetings.

[7] Barabasi, Albert-Laszlo, and Albert, Reka. (1999) *Emergence of Scaling in Random Networks*, Science, 286(5439):509.

[8] Barabasi, Albert-Laszlo, and Jeong, H., and Neda, Z. and Ravasz, E. (2002) *Evolution of the social network of scientific collaboration.* Physics A, 311(3-4):590-614.

[9] Basilico, J., and Hofmann, T. (2004) *Unifying Collaborative and Content-based filtering.* In Proceedings of European Conference on Machine Learning.

[10] Bilgic, Mustafa, and Namata, Galileo M., and Getoor, Lise. (2007). *Combining collective classification and link prediction.* In Proceedings of the Workshop on Mining Graphs and Complex Structures at ICDM Conference.

[11] Brin, Sergey, and Page, Lawrence. (1998). *The anatomy of a large-scale hypertextual Web search engine.* Computer Networks and ISDN Systems, 30(1-7):107-117.

[12] Chawla, Nitesh V, and Bowyer, Kevin W., and Hall, Lawrence O., and W. Kegelmeyer, Philip. (2002) *SMOTE: synthetic minority over-sampling technique.* Journal of Artificial Intelligence Research, 16(1):321-357.

[13] Chung, Fan, and Zhao, Wenbo, (2010). *PageRank and random walks on graphs.* Proceedings of the "Fete of Combinatorics" conference in honor of Lovasz.

[14] Clause, Aaron, and Moore, Christopher, and Newman, M. E. J. (2008). *Hierarchical structure and the prediction of missing links in network.* Nature 453:98-101.

[15] Doppa, Janardhan R., and Yu, Jun, and Tadepalli, Prasad, and Getoor, Lise. (2009). *Chance-Constrained Programs for Link Pre-*

diction. In Proceedings of Workshop on Analyzing Networks and Learning with Graphs at NIPS Conference.

[16] Erketin, Syeda, and Huang, Jian, and Giles, Lee. (2007). *Active learning for Class imbalance problem.* In Proceedings of the 30th ACM SIGIR Conference.

[17] Fabrikant, Alex, and Luthra, Ankur, and Maneva, Elitza, and Papadimitriou, Christos H., and Shenker, Scott. (2003). *On a Network Creation Game.* In Proc. of the twenty-second annual symposium on principles of distributed computing, pp:347-351.

[18] Freschi, Valerio. (2009). *A Graph-based Semi-Supervised Algorithm for Protein Function Prediction from Interaction Maps.* In Learning and Intelligent Optimization, Lecture Notes in Computer Science, 5851:249-258

[19] Frieze, A, and Kannan, R., and Vempala, S. (1998) *Fast monte-carlo algorithms for finding low-rank approximations.* in Journal of the ACM (JACM), 51(6):10251041.

[20] Fu, Wenjie, and Song, Le, and Xing, Eric P. (2009) . In Proc. of the 26th International Conference on Machine Learning.

[21] Getoor, Lise, and Friedman, Nir, and Koller, Dephne, and Taskar, Benjamin. (2002) *Learning Probabilistic Models of Link structure.* Journal of Machine Learning Research, 3:679-707.

[22] Hasan, Mohammad A., and Chaoji, Vineet, and Salem, Saeed and Zaki, Mohammed. (2006) *Link Prediction using Supervised Learning.* In Proceedings of SDM Workshop of Link Analysis, Counterterrorism and Security.

[23] Heckerman, David, and Chickering, David M., and Meek, Christopher, and Rounthwaite, Robert, and Kadie, Carl M. (2000) *Dependency Networks for inference, collaborative filtering, and data visualization.* Journal of Machine Learning Research, 1:49-75.

[24] Heckerman, David, and Meek, Christopher, and Koller, Daphne. (2004) *Probabilistic models for relational data.* Technical Report, Microsoft.

[25] Huang, Zan, and Li, Xin, and Chen Hsinchun. (2005) *Link Prediction approach to collaborative filtering.* Proceedings of the fifth ACM/IEEE Joint Conference on Digital Libraries.

[26] Imrich, W., Klavzar, S. (2000). *Product Graphs: Structure and Recognition.* Wiley.

[27] Jeh, Glen, and Widom, Jennifer. (2002) *SimRank: A measure of structural-context similarity.* In Proceedings of ACM SIGKDD International Conference of Knowledge Discovery and Data Mining.

[28] Karakoulas, Grigoris, and Shawe-Taylor, John. (1999). *Optimizing classifiers for imbalanced training sets.* Proceedings of NIPS, 253-259.

[29] Kashima, Hisashi, and Abe, Naoke. (2006) *A Parameterized Probabilistic Model of Network Evolution for Supervised Link Prediction.* ICDM '06: Proceedings of the Sixth IEEE International Conference on Data Mining. 340-349.

[30] Kashima, Hisashi, and Oyama, Satoshi, and Yamanishi, Yoshihiro, and Tsuda, Koji. (2009). *On Pairwise Kernels: An Efficient Alternative and Generalization Analysis*, Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pp.1030-1037.

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

[32] Kleinberg, Jon M. (2000). *Navigation in a small world.* Nature 406, (845).

[33] Kunegis, Jerome, and Lommatzsch, Andreas. (2009) *Learning Spectral Graph Transformations for Link Prediction.* In Proceedings of the International Conference on Machine Learning, pp 561-568.

[34] Leskovec, Jure, and Kleinberg, Jon M, and Faloutsos, Christos. (2005). *Graphs over time:densification laws, shrinking diameters and possible explanations.* KDD '05: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining.

[35] Li, Xin, Chen Hsinchun. (2009). *Recommendation as link prediction: a graph kernel-based machine learning approach.* Proceedings of the ninth ACM/IEEE Joint Conference on Digital Libraries.

[36] Liben-Nowell, David, and Kleinberg, Jon. (2007). *The Link Prediction Problem for Social Networks.* Journal of the American Society for Information Science and Technology, 58(7):1019-1031.

[37] Liu, Yan and Kou, Zhenzhen. (2007). *Predicting who rated what in large-scale datasets.* SIGKDD Exploration Newsletter, 9 (2).

[38] Madadhai, J., and Hutchins, J., and Smyth, P. (2005). *Prediction and Ranking algorithms for event-based Network Data.* SIGKDD Explorations Newsletter, 7(2):23-30.

[39] Malin, Bradley, and Airoldi, Edoardo, and Carley, Kathlee M. (2005). *A Network Analysis Model for Disambiguation of Names in Lists.* In Journal of Computational and Mathematical Organization Theory, 11(2):119-139.

[40] Nallapati, Ramesh, and Ahmed, Amr, and Xing, Eric P., and Cohen, William W. (2008). *Joint Latent Topic Models for Text and*

Citations. In Proc. of The Fourteen ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[41] Newman, M. E. J. (2001). *Clustering and Preferential attachment in growing networks.* PHysical Review Letters E, 64(025102).

[42] Niculescu-Mizil, and Alexandru, and Caruana, Rich. (2005). *Predicting Good Probabilities with Supervised Learning.* International Conference on Machine Learning.

[43] Oyama, Satoshi, and Manning, Christopher D., (2004). *Using feature conjunctions across examples for learning pairwise classifiers*, In The Proc. of European Conference on Machine Learning, pp. 323-333.

[44] Pavlov, Dmitry, and Mannila, Heikki, and Smyth, Phadraic. (2009) *Beyond Independence: Probabilistic Models for Query Approximation on Binary Transaction Data.* University of California, Irvine Technical Report UCI-ICS-TR-01-09.

[45] Pearl, Judea. (1988). *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Fransisco.

[46] Popescul, Alexandrin and Ungar, Lyle H. (2003). *Statistical Relational Learning for Link Prediction.* In Proceedings of Workshop on Learning Statistical Models from Relational Data at IJCAI Conference.

[47] Popescul, Alexandrin and Ungar, Lyle H. (2003). *Structural Logistic Regression for Link Analysis.* In Proceedings of Workshop on Multi-Relational Data Mining at KDD Conference.

[48] Provost, Foster, and Fawcell, Tom. (2001). *Robust Classification for Imprecise Environments.* Machine Learning, 42(3):203-231.

[49] Rattigan, Matthew J., and Jensen, David. (2005). *The case for anomalous link discovery.* SIGKDD Explorations Newsletter, 7 (2):41-47.

[50] Sarukkai, Ramesh R. (2000). *Link Prediction and Path Analysis using Markov Chain.* WWW '00: Proceedings of the Ninth World Wide Web Conference, 377-386.

[51] Shawe-taylor, J., and Cristianini, Nelo. (2004). *Kernel Methods for Pattern Analysis.* Cambridge University Press, NY.

[52] Song, Han H., and Cho Tae W., and Dave, Vacha, and Zhang, Yin, and Qiu, Lili. (2009). *Scalable proximity Estimation and Link Prediction in Online Social Networks*, IMC '09: In Proceedings of the Internet Measurement Conference.

[53] Tasker, Benjamin, and Wong, Ming F., and Abbeel, Pieter, and Koller, Daphne. (2003). Link Prediction in Relational Data. NIPS '03: In Proceedings of Neural Information Processing Systems.

[54] Tasker, Benjamin, and Abbeel, Pieter, and Koller, Daphne. (2002). Discriminative Probabilistic Models for Relational Data. In Proceedings of Uncertainty in Artificial Intelligence Conference.

[55] Taskar, Benjamin, and Abbeel, Pieter, and Wong, M.-F, and Koller, Daphne (2007). *Relational Markov Networks.* In L. Getoor and B. Taskar, editors, Introduction to Statistical Relational Learning.

[56] Tylenda, Tomasz, and Angelova, Ralitsa, and Bahadur, Srikanta. (2009). *Towards time-aware link prediction in evolving social network.* SNA-KDD '09: Proceedings of the third Workshop on Social Network Mining and Analysis.

[57] Campbell, Veropoulos, and Campbell, C.K., and Cristianini, N., *Controlling the sensitivity of support vector machines.* In: Dean, T. (Ed.), IJCAI: Proceedings of International Joint Conference on Artificial Intelligence. pp. 55-60.

[58] Wang, Chao, and Satuluri, Venu, and Parthasarathy, Srinivasan. (2007). *Local Probabilistic Models for Link Prediction.* ICDM '07: In Proceedings of International Conference on Data Mining.

[59] Watts, D, and Stogatz, S. (1998). *Small world.* Nature, 393:440-442.

[60] Weiss, Gary M. (2004) *Mining with rarity: a unifying framework*, In SIGKDD Explorations Newsletter, 6(1):7-19.

[61] Xu, Zhao, and Tresp, Volker, and Yu, Shipeng, and Yu, Kai. (2005). *Nonparametric Relational Learning for Social Network Analysis.* SNA-KDD '08: In Proceedings of the Second Workshop on Social Network Mining and Analysis.

[62] Xu, Zhao, and Tresp, Volker, and Yu, Kai and Kriegel, Hans-Peter. (2005). *Dirichlet Enhanced Relational Learning.* In Proceedings of International Conference on Machine Learning, pp 1004-1011.

[63] Yang, Chan-Yun, and Yang, Jr-Syu, and Wang Jian-Jun. (2009). *Margin Calibration in SVM class-imbalanced learning*, Neurocomputing, 73(1-3):397-411.

[64] Yu, Kai, and Chu, Wei, and Yu, Shipeng, and Tresp, Volker, and Xu, Zhao. (2006). *Stochastic relational models for discriminative link prediction.* In Proceedings of NIPS, pp-1553-1560

[65] Zhu, Jianhan, and Hong, Jun, and Hughes G. (2002). *Using Markov models for web site link prediction.* HYPERTEXT '02: Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia.