
REVIEW ON GRAPH FEATURE LEARNING AND FEATURE EXTRACTION TECHNIQUES FOR LINK PREDICTION

Ece C. Mutlu*

Department of Industrial Engineering
University of Central Florida
ece.mutlu@ucf.edu

Toktam A. Oghaz*

Department of Computer Science
University of Central Florida
toktam@cs.ucf.edu

Amirarsalan Rajabi*

Department of Computer Science
University of Central Florida
amirarsalan@knights.ucf.edu

Ivan Garibay✉

Department of Industrial Engineering
Department of Computer Science
University of Central Florida
igaribay@ucf.edu

ABSTRACT

The problem of link prediction has recently attracted considerable attention by research community. Given a graph, which is an abstraction of the relationships among entities, the task of link prediction is to anticipate future connections among entities in the graph, concerning its current state. Extensive studies have examined this problem from different aspects and proposed various methods, some of which might work very well for a specific application but not as a global solution. This work presents an extensive review of state-of-art methods and algorithms proposed on this subject and categorizes them into four main categories: similarity-based methods, probabilistic methods, relational models, and learning-based methods. Additionally, a collection of network data sets has been presented in this paper, which can be used to study link prediction. To the best of our knowledge, this survey is the first comprehensive study that considers all of the mentioned challenges and solutions for link prediction in graphs with the improvements in the recent years, including the unsupervised and supervised techniques and their evolution over the recent years.

Keywords Graph Analysis · Complex Networks · Proximity Measures · Supervised Link Prediction · Unsupervised Link Prediction

1 Introduction

The applications of graph theory are extensive in understanding the social, natural, and information systems. Particularly, graph analysis research been deployed in fields as diverse as sociology, biology, transportation networks, operation research, etc. [1, 2, 3, 4]. Moreover, the study of graphs plays a crucial role in modern science. Online social networks, biological networks such as protein-protein interactions and genetic interactions between organisms, ecological systems of species, and knowledge networks such as citation networks, are all instances of graphs of complex interactions, also referred to as complex networks. While these networks are almost always dynamic in nature, a vital query is how they change over time. More specifically, what are the future associations between entities in a graph under investigation.

The oldest studies in the field of network science are based upon random graphs [5] proposed by Erdős and Rényi, in which n edges are connected randomly out of $n(n-1)/2$ number of possible edges with probability p . Random graphs demonstrate the common properties and the probabilistic distributions of graphs, and thus, have fueled novel research ideas [6, 7, 8, 9, 10]. Investigating this emerging "connectedness", fascinated researchers to investigate graphs

*Represents equal contribution to this work.

thoroughly. Ever since, researchers have investigated the formation and evolution of real-world graphs. Studies on computational network analysis mainly comprise of statistical analysis of complex networks [11], community detection and node classification [12, 13, 14], evolution of the dynamics of graphs over time [15, 16, 17, 18], information diffusion and cascade analysis [19, 20, 21, 22], data mining [23, 24, 25] and graph visualization [26, 27, 28, 29]. One of the most interesting and long-standing challenges is the problem of link prediction in graphs.

Given a graph which is an abstraction of relationships among entities of a network, link prediction is to anticipate future connections among entities in the graph, with respect to its current state. Link prediction models might (i) exploit the similarity metrics as the input features (ii) embed the nodes into a low dimensional vector space while preserving the topological structure of the graph (iii) combine the information derived from the two aforementioned points, with the node attributes available from the data set. All these models rely on the hypothesis that higher similarity between nodes results in a higher probability of connection [30].

Link prediction has a diverse domain of applications. In online social networks it is being used for analyzing the user-user and user-content recommendations [31, 32, 33], in bioinformatics, it is being used for the reconstruction of the PPI (protein protein interaction) network and reducing the present noise [34, 35, 36], in hyper-link prediction [37], predicting transportation networks [38], and also in forecasting terrorism campaigns [39]. Link prediction in these applications have been mostly investigated through unsupervised graph representation and feature learning methods based on node (local) or path (global) similarity metrics that evaluate the neighboring nodes. Common neighbors, preferential attachment, Jaccard, and Katz and Adamic Adar are some of the most widely used similarity metrics that measure the likelihoods of edge associations in graphs. While these methods may seem dated, they are far from being obsolete. Despite the fact that these methods do not discover the graph attributes, they have remained popular for years due to their simplicity, interpretability and scalability [40]. Probabilistic models, on the other hand, aim predicting the likelihood of future connections between entities in an evolving dynamical graph with respect to the current state of the graph, e.g. the graph of an online social network. Another context under which the problem of link prediction is raised is relational data [41, 42, 43, 44]. In this context, considering the relational data set in which objects are related to each other, the task of link prediction is to predict the existence and the type of links between pairs of objects [45] (These models are explained in Section 5). However, the availability of labeled data allows the supervised machine learning algorithms to also provide solutions for the link prediction task.

In this study, we cover similarity-based (local, global and quasi-local approaches), probabilistic and relational methods as unsupervised solutions to the link prediction problem. In the case of existence of the labeled data set also, we give a detailed overview of learning-based methods.

2 Problem Description and Performance Evaluation

A graph (complex network) denoted as $G = \langle V, E \rangle$ can be defined as the set of vertices (nodes) V , and the interactions among pairs of nodes called links (edges) E , at a particular time t . The main idea behind applying feature extraction or feature learning-based methods for the link prediction problem is to use the present information regarding the existing edges to predict the future or missing link that will emerge at time $t' > t$. Types of graphs can be classified into two main categories according to the direction of the information flow between interacted nodes; directed and undirected graphs. Although many of the discussed methods in the next sections of this paper can provide solutions to the link prediction problem in directed graphs, the majority of the reviewed methods in this survey address the problem of link prediction for undirected graphs. The difference between the link prediction problem for these two graph categories arises from the additional information required for the directed graphs. This information refers to the origin of the associated link in directed graphs in which $\langle v_x, v_y \rangle$ conveys the existence of a directed edge from node v_x to v_y and $\langle v_x, v_y \rangle \neq \langle v_y, v_x \rangle$ [46]. However, edges in undirected graphs have no orientation, and the relations among node pairs are reciprocal. It should be noted that self-interactions of nodes are not allowed and, accordingly, are not taken into account in link prediction [47]. The set of nodes connected to node $v_x \in V$ are known as the “neighbors” of v_x , denoted as $\Gamma(v_x) \in V$, and the number of edges connected to the node v_x is referred to as $|\Gamma(v_x)|$. Link prediction algorithms necessitate training and test sets to be compared in the case of model performance, similar to other machine learning methods. However, one cannot know the future links of a graph at time t' given the current graph structure. Therefore, a fraction of links from the current graph structure is deleted (Figure 1), and taken as the test set (true positive); whereas, the remaining fraction of edges in the graph is used for the training purpose. A reliable link prediction approach should provide higher probabilities for the edges that belong to the set of true positives than the set of nonexistent edges (true negatives) [48]. The most common standard metric that is used to quantify the performance of the link prediction algorithms is “area under the receiver operating characteristic curve (AUC)” [49]. The AUC value represents

the probability that a randomly selected missing link between two nodes is given a higher similarity score than the randomly selected pair of unconnected links. The algorithmic calculation of AUC is given by:

$$AUC = \frac{n' + 0.5n''}{n} \quad (1)$$

where n is the number of total independent comparisons and n' is the number of comparisons in which missing link has higher score than unconnected link while n'' is the number of comparisons when they show equal scores.

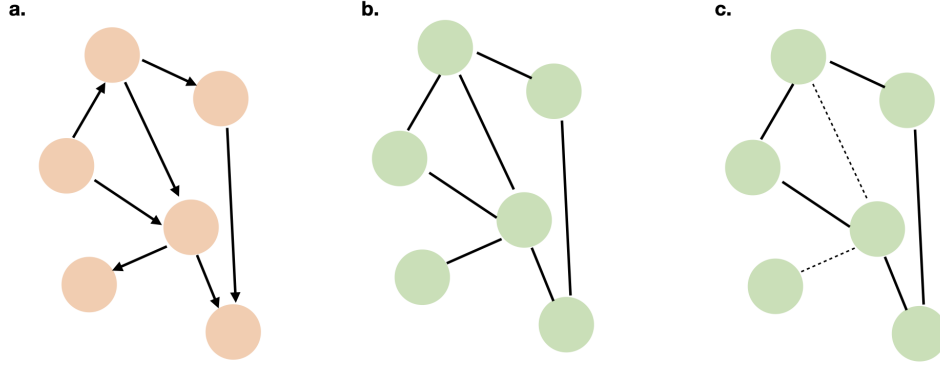


Figure 1: Imaginary representation of a. directed whole graph b. undirected whole graph c. undirected training graph

To provide a few visualization examples for complex networks, Figure 2 demonstrates the network structure of the two different hashtag co-occurrence graphs (*#askmeanything* and *#lovemylife*) of the Instagram posts from 04/01/2020 to 04/08/2020. These two different figures clearly demonstrate the variability of the network structure even in the same fields, i.e. Figure 2.a. shows different sub-communities with its more sparse structure, while Figure 2.b. represents a densely connected network example.

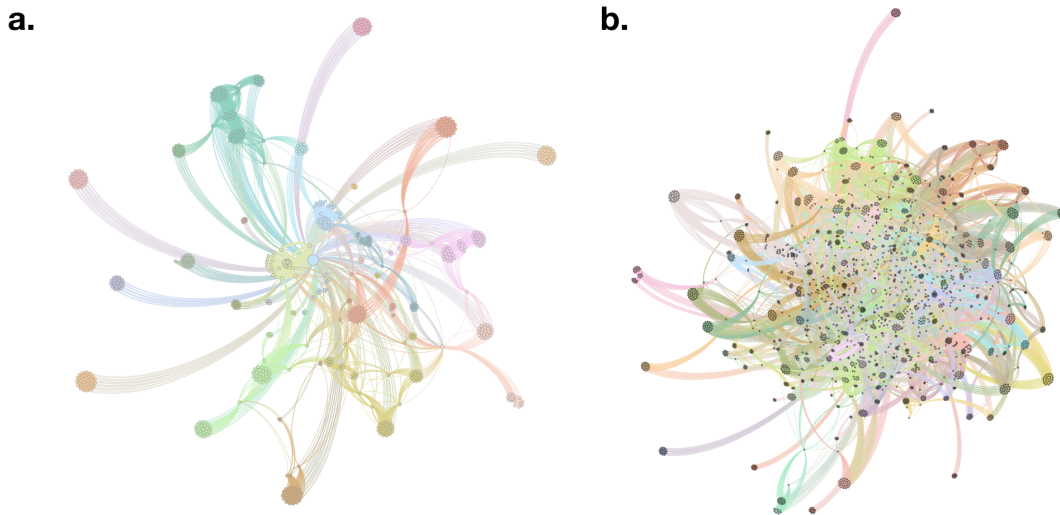


Figure 2: Hashtag co-occurrence graph (via Gephi) of a. *#askmeanything* (757 nodes and 16046 edges), b. *#lovemylife* (2748 nodes and 63413 edges). Each network is colored by the modularity ranking.

3 Similarity Based Methods

Similarity-based methods, which mainly focus on the topological structure of the graph, are the most straightforward and oldest link prediction metrics. These methods try to figure missing links out by assigning similarity score, $s_{(v_x, v_y)}$, between node pairs (v_x and v_y) using the structural property of the graphs. These methods can be investigated under three main categories: local, quasi-local, and global approaches.

3.1 Local Similarity-Based Approaches

Local similarity-based approaches are based on the assumption that if node pairs have common neighbor structures, they will probably form a link in the future. Because they use only local topological information based on neighborhood-related structures rather than considering the whole network topology, they are faster than the global similarity-based approaches. Many studies also showed their superior performance, especially on the dynamic networks [50]. However, they are restricted to compute the similarity of all possible combinations of the node pairs since they only rank similarity between close nodes with a distance of less than two.

3.1.1 Common Neighbors (CN)

CN is one of the most extensive information retrieval metrics for link prediction tasks due to its high efficiency despite its simplicity. The idea behind CN is very intuitive; that the probability of being linked for two nodes in the future is affected by the number of their common neighboring nodes, i.e., two nodes will highly probably establish a link if they have more shared nodes. The score of this metric can be defined as follows:

$$s_{(v_x, v_y)}^{CN} = |\Gamma(v_x) \cap \Gamma(v_y)| \quad (2)$$

where $\Gamma(\cdot)$ represents the set of adjacent nodes.

It should be noted that, resulting score using CN is not normalized, and only shows the relative similarity of different node-pairs by considering shared nodes between them. Newman used CN to show that the probability of collaboration between two scientists in the future can be estimated by their previous common collaborators [51].

3.1.2 Jaccard Index (JC)

The metric not only takes the number of common nodes into account as in CN, but also normalizes it by considering the total set of number of shared and non-shared neighbors. The equation of this score proposed by Jaccard [52] is:

$$s_{(v_x, v_y)}^{JC} = \frac{|\Gamma(v_x) \cap \Gamma(v_y)|}{|\Gamma(v_x) \cup \Gamma(v_y)|} \quad (3)$$

3.1.3 Salton Index (SL)

SL is the metric that is also known as cosine similarity. It calculates the cosine angle between the two columns of the adjacency matrix and is identified as the ratio of the number of shared neighbors of v_x and v_y to the square root of inner-product of their degrees [53] as follows:

$$s_{(v_x, v_y)}^{SL} = \frac{|\Gamma(v_x) \cap \Gamma(v_y)|}{\sqrt{|\Gamma(v_x)| \cdot |\Gamma(v_y)|}} \quad (4)$$

Wagner & Leydesdorff [54] showed that SI is an efficient metric, especially when the aim is to visualize the constructional pattern of relations in a graph.

3.1.4 Sørensen Index (SI)

The index, which is very similar to JC, is generated to make a comparison between different ecological samples [55] such that:

$$s_{(v_x, v_y)}^{SI} = \frac{|\Gamma(v_x) \cap \Gamma(v_y)|}{|\Gamma(v_x)| + |\Gamma(v_y)|} \quad (5)$$

The difference in using the summation of the degrees instead of degrees of their union makes SI less outlier sensitive compared to JC [56].

3.1.5 Preferential Attachment Index (PA)

Motivated by the study by Barabasi & Albert [57], new nodes joining the network are more likely to connect with the nodes with higher connections (hub) than the nodes with lower degrees, PA can be formulated as:

$$s_{(v_x, v_y)}^{PA} = |\Gamma(v_x)| \cdot |\Gamma(v_y)| \quad (6)$$

3.1.6 Adamic-Adar Index (AA)

The metric is employed for the necessity of the comparison of two web-pages by Lada Adamic and Eytan Adar [58]. It simply uses the idea of giving more weight to the relatively fewer common neighbors such that:

$$s_{(v_x, v_y)}^{AA} = \sum_{v_z \in |\Gamma(v_x) \cap \Gamma(v_y)|} \frac{1}{\log|\Gamma(v_z)|} \quad (7)$$

where v_z refers to a common neighbor for nodes v_x and v_y (connected/linked to both).

Although this metric has similarities to CN, the vital difference is that the logarithm term penalizes the shared neighbors of the two corresponding nodes. It should be noted that while the other metrics include only two nodes (v_x and v_y) and/or their degrees in their equations so far, AA also relates familiar neighbors (v_z) to these two nodes (v_x and v_y).

3.1.7 Resource Allocation Index (RA)

Motivated by physical process of resource allocation, a very similar metric to AA is developed by Zhou et al. [59] which can be formulated as:

$$s_{(v_x, v_y)}^{RA} = \sum_{v_z \in |\Gamma(v_x) \cap \Gamma(v_y)|} \frac{1}{|\Gamma(v_z)|} \quad (8)$$

The difference in the denominator ($|\Gamma(v_z)|$) of RA rather than its logarithm ($\log|\Gamma(v_z)|$) as in AA penalizes the contribution of common neighbors more. Many studies show that this discrepancy is insignificant, and the resulting performances of these two metrics are very similar when the average degree of the network is low; however, RA is superior when the average degree is high [60].

3.1.8 Hub Promoted Index (HP)

The index is proposed for assessing the similarity of the substrates in the metabolic networks [61], and can be defined as follows:

$$s_{(v_x, v_y)}^{HP} = \frac{|\Gamma(v_x) \cap \Gamma(v_y)|}{\min(|\Gamma(v_x)|, |\Gamma(v_y)|)} \quad (9)$$

HP is determined by the ratio of common neighbors of both v_x and v_y to the minimum of degrees of v_x and v_y . Here, link formation between lower degree nodes and the hubs is more promoted while the formation of the connection between hub nodes are demoted [30].

3.1.9 Hub Depressed Index (HD)

The totally opposite analogy of HP is also considered by Lü and Zhou [47], and it is determined by the ratio of common neighbors of both v_x and v_y to the maximum of degrees of v_x and v_y . Here, link formation between lower degree nodes and link formation between hubs is promoted. However, the connection between hub nodes and lower degree nodes are demoted such that:

$$s_{(v_x, v_y)}^{HD} = \frac{|\Gamma(v_x) \cap \Gamma(v_y)|}{\max(|\Gamma(v_x)|, |\Gamma(v_y)|)} \quad (10)$$

3.1.10 Leicht-Holme-Newman Index (LHN)

The index, very similar to SI, is defined as the ratio of the number of shared neighbors of v_x and v_y to the product of their degrees (the expected value of the number of paths of length between them) [62]. It can be represented by:

$$s_{(v_x, v_y)}^{LHN} = \frac{|\Gamma(v_x) \cap \Gamma(v_y)|}{|\Gamma(v_x)| \cdot |\Gamma(v_y)|} \quad (11)$$

The only difference in the denominator compared to SI shows that SI always assigns higher score than LHN, i.e. $|\Gamma(v_x)| \cdot |\Gamma(v_y)| \geq |\Gamma(v_x) \cap \Gamma(v_y)|$.

3.1.11 Parameter Dependent Index (PD)

Zhou et al. [63] proposed a new metric to improve the prediction accuracy for popular links and unpopular links. PD can be defined as:

$$s_{(v_x, v_y)}^{PD} = \frac{|\Gamma(v_x) \cap \Gamma(v_y)|}{|\Gamma(v_x)| \cdot |\Gamma(v_y)|^\beta}, \quad (12)$$

where β is a free parameter and can be tuned to the topology of the graph. One can easily recognizes that PD is degraded to CN, SL and LHN when $\beta = 0$, $\beta = 0.5$, and $\beta = 1$, respectively.

3.1.12 Local Affinity Structure Index (LAS)

LAS shows the affinity relationship between a pair of nodes and their common neighbors. The hypothesis is that a higher affinity of two nodes and their common neighbors increases the probability of getting connected [64], such as:

$$s_{(v_x, v_y)}^{LAS} = \frac{|\Gamma(v_x) \cap \Gamma(v_y)|}{|\Gamma(v_x)|} + \frac{|\Gamma(v_x) \cap \Gamma(v_y)|}{|\Gamma(v_y)|} \quad (13)$$

3.1.13 CAR Based Index (CAR)

When a node interacts with another neighbor node, it is called a first-level neighborhood; whereas, the interaction between the first-level neighbor node and its neighbor node is called the second-level neighborhood for the seed node. According to Cannistraci [65], researchers mostly consider the first-level neighborhood because the second-level neighborhood is noisy. However, the second-level neighborhood carries essential information regarding the topology of the network. Therefore, CB filters these noises and considers nodes interlinked with neighbors mostly. The similarity metric can be calculated as follows:

$$s_{(v_x, v_y)}^{CAR} = \sum_{v_z \in \Gamma(v_x) \cap \Gamma(v_y)} 1 + \frac{|\Gamma(v_x) \cap \Gamma(v_y) \cap \Gamma(v_z)|}{2}. \quad (14)$$

3.1.14 The Individual Attraction Index (IA)

Dong et al. [66] proposed an index that relates not only to the common neighbors of the nodes individually but also the effect of the sub-network created by those. The IA score can be formulated as:

$$s_{(v_x, v_y)}^{IA} = \sum_{v_z \in \Gamma(v_x) \cap \Gamma(v_y)} \frac{e_{v_z}}{|\Gamma(v_z)|}, \quad (15)$$

where e_{v_z} is the number of links among node v_z with nodes v_x and v_y , and their common neighbors. Since IA considers the existence of links between all common neighbors, the algorithm is very time-consuming.

3.1.15 The Mutual Information Index (MI)

This method examines the link prediction problem using information theory, and measures the likelihood of conditional self-information when their common neighbors are known [67] and formulated as:

$$s_{(v_x, v_y)}^{MI} = -I(e_{v_x, v_y} | v_z), \quad (16)$$

where $v_z \in |\Gamma(v_x) \cap \Gamma(v_y)|$ and $I(\cdot)$ is the self-information function for a node and can be calculated by (17). The smaller value means the higher likelihood to be linked. If all the link between common neighbors be independent of each other, the self-information of that node pair can be calculated as [30]:

$$I(e_{v_x, v_y} | v_z) = \log 2 \frac{|\{e_{v_x, v_y} : v_x, v_y \in \Gamma(v_z), e_{v_x, v_y} \in E\}|}{\frac{1}{2} |\Gamma(v_z)| (|\Gamma(v_z)| - 1)}. \quad (17)$$

3.1.16 Functional Similarity Weight (FSW)

This index is first used by Chou et al. in order to understand the similarity of physical or biochemical characteristics of proteins [68]. Their motivation is based on the Czekanowski-Dice distance used in [69] to estimate the functional similarity of proteins. This score can be defined as:

$$s_{(v_x, v_y)}^{FSW} = \left(\frac{2|\Gamma(v_x) \cap \Gamma(v_y)|}{|\Gamma(v_x) - \Gamma(v_y)| + 2|\Gamma(v_x) \cap \Gamma(v_y)| + \beta} \right)^2. \quad (18)$$

Here, β is used to penalize the nodes with very few common neighbors, and defined as:

$$\beta = \max(0, \Gamma_{avg} - (|\Gamma(v_x) - \Gamma(v_y)|) + (|\Gamma(v_x) \cap \Gamma(v_y)|)), \quad (19)$$

where Γ_{avg} is the average number of neighbours in the network.

3.1.17 Local Neighbors Link Index (LNL)

Motivated by the cohesion between common neighbors and predicted nodes, both attribute, and topological features are examined in [70] as:

$$s_{(v_x, v_y)}^{LNL} = \sum_{v_z \in |\Gamma(v_x) \cap \Gamma(v_y)|} w(v_z), \quad (20)$$

where $w(v_z)$ is the weight function that can be measured by:

$$w(v_z) = \frac{\sum_{v_u \in \Gamma(v_x) \cup v_x} \delta(v_z, v_u) + \sum_{v_v \in \Gamma(v_y) \cup v_y} \delta(v_z, v_v)}{|\Gamma(v_z)|}. \quad (21)$$

Here, $\delta(a, b)$ is a boolean variable which is equal to 1 if there exist a link between a and b; otherwise equals to 0.

3.2 Global Similarity-Based Approaches

Global similarity-based approaches, on the contrary of local ones, use the whole topology of the network to rank similarity between node pairs; therefore, they are not limited to measure the similarity between nodes that are locating far away from each other. Although considering the whole topology of the network gives more flexibility in link prediction analysis, it also increases the algorithm's time complexity. Since ensemble of all paths between node pairs is used, they can also be called path-based methods.

3.2.1 Katz Index (KI)

The metric, which is defined by Katz [71], sums over the sets of paths and exponentially damped by length to be counted more intensively with shorter paths. This index can be formulated as:

$$s_{(v_x, v_y)}^{KI} = \sum_{i=1}^{\infty} \beta^i \cdot |A_{v_x v_y}^{(i)}|. \quad (22)$$

Here, β is a free parameter ($\beta > 0$) and also called the ‘‘damping factor’’. One can realize that KI yields very similar score when β is low enough, because the paths which have higher lengths contribute less, and similarity index is simply determined by the shorter paths [47].

In the case of $\beta < \frac{1}{\lambda_1^A}$, where λ_1^A is the largest eigenvalue of the adjacency matrix, the similarity matrix can be written as follows:

$$S^{KI} = (I - \beta A)^{-1} - I, \quad (23)$$

where I is the identity matrix.

3.2.2 Global Leicht-Holme-Newman Index (GLHN)

The idea behind GLHN is very similar to that of KI, since it also considers a high similarity for the nodes if the number of paths between these corresponding nodes are high [62] such that:

$$S^{GLHN} = \beta_1 (I - \beta_2 A)^{-1}, \quad (24)$$

where β_1 and β_2 are free parameters, and smaller value of β_2 considers a higher importance for the shorter paths, and vice versa.

3.2.3 SimRank (SR)

This index computes the similarity starting from the hypothesis “two objects are similar if they are related to similar objects.”, and is recursively defined [72]. SR is equal to 1 when node $v_x = v_y$, otherwise:

$$s_{(v_x, v_y)}^{SR} = \gamma \cdot \frac{\sum_{v_{z_1} \in \Gamma(v_x)} \sum_{v_{z_2} \in \Gamma(v_y)} s_{(v_{z_1}, v_{z_2})}^{SR}}{|\Gamma(v_x)| \cdot |\Gamma(v_y)|}, \quad (25)$$

where $\gamma \in [0, 1]$ is called decay factor and controls how fast the effect of neighbor node pairs (v_{z_1} and v_{z_2}) reduces as they move away from the original node pairs (v_x, v_y). SR can be explained in terms of a random walk process, that is, $s_{(v_x, v_y)}^{SR}$ measures how long the two random walkers are expected to meet on a particular node, starting with the v_x and v_y nodes. Its applicability is constrained on large networks due to its computational complexity [60, 73].

3.2.4 Pseudo-inverse of the Laplacian Matrix (PLM)

Using *Laplacian matrix* $L = D - A$ rather than *Adjacency matrix* A gives an alternative representation of a graph, where D is the unit diagonal matrix [74] ($D_{i,j} = 0$ and $D_{i,i} = \sum_j A_{i,j}$). The Moore-Penrose pseudo-inverse of the Laplacian matrix, represented by L^+ can be used in the calculation of proximity measures [75]. Since PLM is calculated as inner product cosine similarity, it is also called “cosine similarity time” in the literature [60] and can be calculated as:

$$s_{(v_x, v_y)}^{PLM} = \frac{L_{(v_x, v_y)}^+}{\sqrt{L_{(v_x, v_x)}^+ L_{(v_y, v_y)}^+}}. \quad (26)$$

3.2.5 Hitting Time (HT) and Average Commute Time (ACT)

Motivated by random walk, introduced by mathematician Karl Pearson [76], HT is defined as the average number of steps to be taken by a random walker starting from v_x to reach node v_y . Because HT is not a symmetric metric, one may consider to ACT, which is defined as the average number of steps to be taken by the random walker starting from v_x to reach the node v_y , and that from v_y to reach node v_x . Therefore, HT can be computed by:

$$s_{(v_x, v_y)}^{HT} = 1 + \sum_{v_z \in \Gamma(v_x)} P_{v_x, v_z} s_{(v_z, v_y)}^{HT}. \quad (27)$$

Here, $P_{i,j} = D^{-1}A$, where A and D are the adjacency and unit diagonal matrix [60]. Accordingly, ACT can be formulated as:

$$s_{(v_x, v_y)}^{ACT} = s_{(v_x, v_y)}^{HT} + s_{(v_y, v_x)}^{HT}. \quad (28)$$

For the sake of computational simplicity, ACT can be computed in closed form by using the pseudo-inverse of the Laplacian matrix of the graph as follows [75]:

$$s_{(v_x, v_y)}^{ACT} = m(L_{(v_x, v_x)}^+ + L_{(v_y, v_y)}^+ - 2L_{(v_x, v_y)}^+). \quad (29)$$

One challenge of HT and ACT is that it gives very small proximity measures when terminal node has high stationary probability π_{v_y} , regardless of the identity of the starting node. This problem can be solved by normalizing the scores as $-s_{(v_x, v_y)}^{HT} \cdot \pi_{v_y}$ and $-(s_{(v_x, v_y)}^{HT} \cdot \pi_{v_y} + s_{(v_y, v_x)}^{HT} \cdot \pi_{v_x})$, respectively [50].

3.2.6 Rooted PageRank (RPR)

PageRank (PR) is the metric used by Google Search to determine the relative importance of the webpages by treating links as a vote. The recursively defined PR on $G(V, E)$ can be obtained for a single node as follows:

$$s_{(v_x)}^{PR} = \frac{1 - \beta}{|V|} + \beta \sum_{v_z \in \Gamma^{-1}(v_x)} \frac{s_{(v_z)}^{PR}}{|\Gamma(v_x)|}, \quad (30)$$

where β is the damping factor. Personalized PR (PRP) can be obtained by inner product of the two PR values of the nodes as:

$$s_{(v_x, v_y)}^{PRP} = s_{(v_x)}^{PR} \cdot s_{(v_y)}^{PR}. \quad (31)$$

RPR, on the other hand, defines that the rank of a node is proportional to the likelihood that it can be reached through a random walk [60] such that:

$$s_{(v_x, v_y)}^{RPR} = (1 - \beta)(1 - \beta P_{v_x, v_y})^{-1}. \quad (32)$$

Here, $P_{i,j} = D^{-1}A$, where A is the adjacency matrix and D is the unit diagonal matrix. It should be noted that, one can calculate PR by averaging the columns of RPR [31].

3.2.7 Escape Probability (EP)

The metric, which can be derived from RPR, measures the likelihood that the random walk starting from node v_x visits node v_y before coming back to the node v_x again [77]. Let $Q(v_x, v_y)$ to be equal to $(1 - \beta D^{-1}A)^{-1} = s_{(v_x, v_y)}^{RPR} / (1 - \beta)$, the equation of EP can be written as follows [31]:

$$s_{(v_x, v_y)}^{EP} = \frac{Q(v_x, v_y)}{Q(v_x, v_x) \cdot Q(v_y, v_y) - Q(v_x, v_y) \cdot Q(v_y, v_x)}. \quad (33)$$

3.2.8 Random Walk with Restart (RWR)

In a random walk (RW) algorithm, the probability vector of reaching a node starting from the node v_x can be defined as:

$$p_{v_x}^{\rightarrow}(t) = M^T p_{v_x}^{\rightarrow}(t-1), \quad (34)$$

where M is called transition probability matrix, and can be calculated by $A_{i,j} / \sum_k A_{i,k}$, where A is the adjacency matrix [78]. Since RW does not yield symmetric matrix, the metric of RWR, very similar to RPR, looks for the probability that a random walker starting from node v_x visits node v_y and come back to the initial state node v_x at the steady state such that:

$$s_{(v_x, v_y)}^{RWR} = p_{v_x}^{\rightarrow v_y} + p_{v_y}^{\rightarrow v_x}. \quad (35)$$

3.2.9 Maximal Entropy Random Walk (MERW)

The basic MERW algorithm, based on the maximum uncertainty principle, was proposed as a result of the necessity to define uniform path distribution in Monte Carlo simulations [79]. However, its application on the stochastic models are very recent [80]. Li et al. [81] proposed MERW to maximize the entropy of a random walk as follows:

$$\lim_{t \rightarrow \infty} \frac{-\sum_{A_{v_x v_y}^t \in A^t} p(A_{v_x v_y}^t) \ln p(A_{v_x v_y}^t)}{t}. \quad (36)$$

Here, $p(A_{v_x v_y}^t)$ is the multiplication of the iterative transition matrices ($M_{v_x v_z} \cdot M_{v_z v_q} \dots M_{v_q v_y}$), where M_{ij} can be calculated as follows:

$$M_{v_i v_j} = \frac{A_{v_i v_j}}{\lambda} \frac{\psi_{v_j}}{\psi_{v_i}}, \quad (37)$$

where A is the adjacency matrix, and ψ is the normalized eigenvector with normalization constant λ [30].

3.2.10 The Blondel Index (BI)

The index is proposed by Blondel et al. [82] to measure the similarity for the automatic extraction of synonyms in a monolingual dictionary. Although BI is used to quantify the similarity between two different graphs, Martinez et al. show that investigating the similarity of two vertices in a single graph can also be evaluated in an iterative manner as:

$$S(t) = \frac{AS(t-1)A^T + A^T S(t-1)A}{\|AS(t-1)A^T + A^T S(t-1)A\|_F}, \quad (38)$$

where $S(t)$ refers to the similarity matrix in iteration t and $S(0) = I$. $\|M\|_F$ is the Frobenius matrix norm and can be calculated as follows:

$$\|M_{m \times n}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n M_{i,j}^2}. \quad (39)$$

The similarity metric is obtained when $S(t)$ is converged such that $s_{(v_x, v_y)}^{BI} = S_{v_x, v_y}(t = c)$, where $t = c$ denotes the steady state level.

3.3 Quasi-Local Similarity-Based Approaches

The trade-off between the efficiency of the information regarding the whole network topological structure for the global approaches and the less time complex algorithms for the local-based methods have resulted in the emergence of quasi-local similarity-based methods for link prediction. Similarly, these approaches are limited in the calculation of the similarity between arbitrary node pairs. However, quasi-local similarity methods provide an opportunity to compute the similarity between a node and its neighbors of neighbors. Although some of the quasi-local similarity-based methods consider the whole topology of the network, their time complexity is less than that of global similarity-based approaches.

3.3.1 The Local Path Index (LPI)

The index, very similar to the well known approaches KI and CN, considers the local path with a wider perspective by not only employing the information of the nearest neighbors, but also the next 2 and 3 nearest neighbors [83, 59] such that:

$$S^{LP} = A^2 + \beta A^3, \quad (40)$$

where β is a free parameter to adjust the relative importance of the neighbors within the length $l = 2$ distances and length $l = 3$ distances. The metric can be also extended for the higher orders as:

$$S^{LP^{(L)}} = \sum_{l=2}^L \beta^{l-2} A^l. \quad (41)$$

Due to an increasing complexity in the higher orders of LP, the neighbors within the length 3 distances are preferable. One can easily realize that this similarity matrix simplifies to CN when $l = 2$, and may produce a very similar result to KI given low β values without the inverse transform process. The similarity between two nodes can be evaluated via $s_{(v_x, v_y)}^{LP} = S_{v_x, v_y}^{LP}$.

3.3.2 Local (LRW) and Superposed Random Walks (SRW)

Although the random walk-based algorithms perform well, the sparsity and the computational complexity regarding massive networks are challenging for these algorithms. Thus, Liu and Lü proposed the LRW metric [84], in which the initial resources for the random walker are assigned based on their importance in the graph. LRW considers the node degree as an important feature and does not concentrate on the stationary state. Instead, the number of iterations is fixed to perform a few-step random walk. LRW can be formulated as:

$$s_{(v_x, v_y)}^{LRW}(t_c) = \frac{|\Gamma(v_x)|}{2|E|} p_{v_y}^{\vec{v}_x}(t_c) + \frac{|\Gamma(v_y)|}{2|E|} p_{v_x}^{\vec{v}_y}(t_c). \quad (42)$$

Since superposing all the random walkers starting from the same nodes may help prevent the sensitive dependency of LRW to the farther neighboring nodes, SRW is proposed as:

$$s_{(v_x, v_y)}^{SRW}(t_c) = \sum_{t=1}^{t_c} s_{(v_x, v_y)}^{LRW}(t). \quad (43)$$

3.3.3 Third-Order Resource Allocation Based on Common Neighbor Interactions (RACN)

Motivated by the RA index, Zhang et al. [85] proposed RACN in which resource of nodes are allocated to the neighbors as:

$$s_{(v_x, v_y)}^{RACN} = \sum_{v_z \in \Gamma(v_x) \cap \Gamma(v_y)} \frac{1}{|\Gamma(v_z)|} + \sum_{e_{v_i, v_j} \in E, |\Gamma(v_j)| < |\Gamma(v_i)|} \left(\frac{1}{|\Gamma(v_i)|} - \frac{1}{|\Gamma(v_j)|} \right), \quad (44)$$

where $v_i \in \Gamma(v_x)$ and $v_j \in \Gamma(v_y)$. The superiority of the RACN over the original RA has been shown in [40] using varying datasets.

3.3.4 FriendLink Index (FL)

The similarity of two nodes is determined according to the normalized counts of the existing paths among the corresponding nodes with varying length L . The formulation for the FL index is as follows:

$$s_{(v_x, v_y)}^{FL} = \sum_{l=1}^L \frac{1}{l-1} \frac{|A_{v_x, v_y}^l|}{\prod_{j=2}^l (|V| - j)}, \quad (45)$$

where $|V|$ is the number of vertices in graph. The metric is favorable due to its high performance and speed [86].

3.3.5 PropFlow Predictor Index (PFP)

PFP is a metric which is inspired by Rooted PageRank, and simply equals to the probability that the success of random walk started from node v_x and terminates at node v_y in not more than l steps [87]. This restricted random walk selects the links based on weights, denoted as ω [60] such that:

$$s_{(v_x, v_y)}^{PFP} = s_{(v_x, v_x)}^{PFP} \frac{\omega_{v_x v_y}}{\sum_{v_z \in \Gamma(v_x)} \omega_{v_x v_z}}. \quad (46)$$

The most important superiority of PFP is its widespread use in directed, undirected, weighted, unweighted, sparse or dense networks.

4 Probabilistic Methods

Probabilistic models are supervised models that use Bayes rules. The most important drawback of some of these models is their being slow and costly for large networks [45]. In the following, we introduce five most important probabilistic methods of link prediction.

4.1 Hierarchical Structure Model

This model was developed based on the observation that many real networks present a hierarchical topology [88]. This maximum likelihood based method, searches for a set of hierarchical representations of the network and then sorts the probable node pairs by averaging over all the hierarchical representations explored. The model was first proposed in the work of [89] in which it develops a hierarchical network model which can be represented by a dendrogram, with $|N|$ leaves and $|N| - 1$ internal nodes. Each leaf is a node from the original network and each internal node represents the relationship of the descendent nodes in the dendrogram. A value of p_r is also attributed to each internal node r which represents the probability with which a link exists between the branches descending from it. If D is a dendrogram that represents the network, the likelihood of dendrogram with a set of internal node probabilities (p_r) is:

$$\mathcal{L}(D, \{p_r\}) = \prod_{r \in D} p_r^{E_r} (1 - p_r)^{L_r R_r - E_r}. \quad (47)$$

In the above equation, E_r is the number of links that connect nodes that have a node r as their lowest common ancestor in D . L_r and R_r represent the number of leaves in the left and the right subtrees that are rooted in r , respectively. Setting $p_r^* = \frac{E_r}{L_r R_r}$ maximizes the likelihood function 47. Replacing p_r with p_r^* in the likelihood function, likelihood of a dendrogram at its maximum can be calculated by:

$$\mathcal{L}(D) = \prod_{r \in D} \left[(1 - p_r^*)^{1 - p_r^*} p_r^* p_r^* \right]^{L_r R_r}. \quad (48)$$

These equations are utilized then to perform link prediction. After a Markov Chain Monte Carlo method is used to sample a large number of dendrograms with probabilities proportional to their likelihood, the connection probability between two nodes v_i and v_j is estimated by averaging over all sampled dendrograms. This task is performed for all sampled dendrograms and subsequently, the node pairs are sorted based on the corresponding average probabilities. The higher the ranking, the more likely that the link between the node pair exists. A major drawback of hierarchical structural model is its computational cost and being very slow for a network consisting of a large set of nodes.

4.2 Stochastic Blockmodel

Stochastic block models are based on the idea that nodes that are heavily interconnected should form a block or community [90]. In a stochastic block model, nodes are separated into groups and the probability that two nodes are connected to each other is merely dependant on the group to which they belong [91]. Stochastic block models have successfully been applied to model the structure of complex networks [92, 93]. They have also been utilized to predict the behavior in drug interactions [94]. The work of [95] uses a block model to predict conflict between team members. [96] also utilizes stochastic block model to develop a probabilistic recommender system.

As noted above, The probability that two nodes i and j are connected is depends on the blocks that they belong to. A block model $M = (P, Q)$ is completely determined by the partition P of nodes into groups and the matrix Q of probabilities of linkage between groups. While numerous partitions (models) can be considered for a network, the likelihood of a model A^O can be calculated by the following [97, 91]:

$$\mathcal{L}(A^O | P, Q) = \prod_{\alpha \leq \beta} Q_{\alpha\beta}^{l_{\alpha\beta}^O} (1 - Q_{\alpha\beta})^{r_{\alpha\beta} - l_{\alpha\beta}^O}. \quad (49)$$

In equation 49, $l_{\alpha\beta}^O$ is the number of links in A^O between nodes in groups α and β of P , and $r_{\alpha\beta}$ is the maximum number of links possible, which is $|\alpha||\beta|$ when $\alpha \neq \beta$ and $\binom{|\alpha|}{2}$ when $\alpha = \beta$. Setting $Q_{\alpha\beta}^* = \frac{l_{\alpha\beta}^O}{r_{\alpha\beta}}$ maximizes the likelihood function 49. By applying Bayes theorem the probability (*reliability*) of a link with maximum likelihood can be computed.

Similar to hierarchical structure model discussed in section 4.1, a significant shortcoming of this method is it being very time-consuming. While Metropolis algorithm ([98]) can be utilized to sample partitions, this approach is still impractical for a large network.

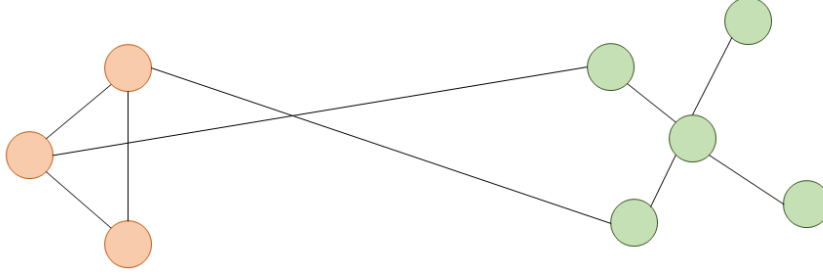


Figure 3: **An example of Blockmodel likelihood calculation** Here a probable partitioning is presented. The block on left is α and the block on right is β . $Q_{\alpha\alpha}^* = 1$, $Q_{\alpha\beta}^* = \frac{2}{12}$, and $Q_{\beta\beta}^* = \frac{5}{6}$. Hence, likelihood is calculated as follows:
 $1^3 \times 1 \times \frac{2}{15}^2 \times \frac{13}{15}^{13} \times \frac{1}{2}^5 \times \frac{1}{2}^5 \approx 2.701 \times 10^{-6}$

4.3 Network Evolution Model

[99] proposed a network topology based model for link prediction. In this model, probabilistic flips of the existence of edges are modeled by a "copy-and-paste" process between the edges [99]. The problem of link prediction is defined as follows: the data domain is represented as a graph $G = (V, s)$, where V is the set of nodes of the network and $s : V \times V \rightarrow [0, 1]$ is an edge label function. $s(v_i, v_j)$ indicates the probability that an edge exists between i and j . $s^{(t)}$ shows the edge label function at time t , and its Markovian nature, i.e. $s^{(t+1)}$ only depends on $s^{(t)}$. The fundamental idea behind the proposed network edge label *copy-and-paste* mechanism is that if a node is having a strong influence on another node, then the second nodes association will be highly affected by the second node. The probability of an edge existing between nodes i and j at time $t + 1$ is as follows:

$$s^{t+1}(v_i, v_j) = \frac{1}{|V| - 1} \left(\sum_{k \neq i, j} w_{v_k v_j} s^{(t)}(v_k, v_i) + w_{v_k v_i} s^{(t)}(v_k, v_j) \right) + \left(1 - \frac{1}{|V| - 1} \sum_{k \neq i, j} w_{v_k v_j} + w_{v_k v_i} \right) s^{(t)}(v_i, v_j) \quad (50)$$

Where $w_{v_k v_j}$ is the probability that an edge label is copied from node v_k to node v_j . In equation 50 the first term represents the probability that the edge label for (v_i, v_j) is changed by copy and pasting. The second term represents when the same edge label is unchanged. The linkages are obtained by iteratively updating equation 50 until convergence. The objective function according to which the parameters are set is solved by an expectation maximization type transductive learning.

4.4 Local Probabilistic Model

The work of [100] proposed a local probabilistic model for link prediction, in which the focus of the original paper is particularly in the context of evolving co-authorship networks. Given the candidate link, e.g. nodes v_i and v_j , first the *central neighborhood set* of v_i and v_j are determined, which is the set of nodes that are most relevant to estimating the co-occurrence probability. The central neighborhood set are chosen from the nodes that lie along paths of shorter length between v_i and v_j . [100] proposes an algorithm to determine central neighborhood set, which is as follows: first collecting all nodes that lie on length-2 simple paths, then those on length-3 simple paths, and so on. The paths are then ordered based on the frequency scores and the ones with highest score are chosen [100]. A path length threshold is also considered for the sake of decreasing computational cost ([100] proposes a threshold of 4 for their specific problem). Next, they form a transaction dataset which is formed by a chronological set of events (co-authoring articles). A non-derivable itemset mining is performed on the this dataset which results in all non-redundant itemsets along with their frequencies. In the end a Markov Random Field (MRF) graph model is trained using the derived dataset. The resulting final model gives the probability of existence of each link v_i and v_j .

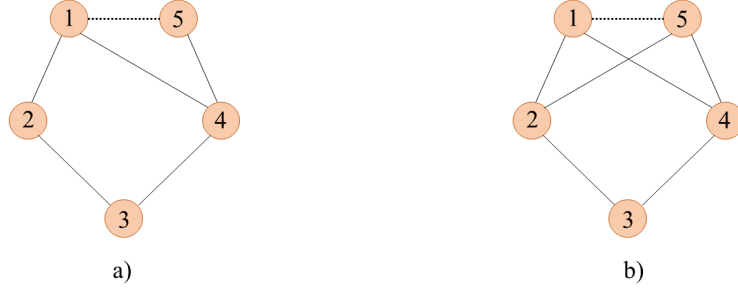


Figure 4: a) There is only one path of length two from node 1 to node 5 in the network and $\Pr_2((1, 5) \in E) = c_2$. b) There are two paths of length 2 from the node 1 to node 5, therefore $\Pr_2((i, j) \in E) = \frac{c_2^2}{c_2^2 + (1 - c_2)^2}$

4.5 Probabilistic Model of Generalized Clustering Coefficient

This method which is proposed by [101], focuses on analyzing the predictive power of clustering coefficient [101]. The generalized clustering coefficient $C(k)$ of degree k is defined as [101]:

$$C(k) = \frac{\text{number of cycles of length } k \text{ in the graph}}{\text{number of paths of length } k} \quad (51)$$

As explained in [101], generalized clustering coefficients describe the correlation between cycles and paths in a network. Therefore, the probability of formation of a particular link is determined by the number of cycles (of different lengths) that will be constructed by adding that link [101]. The concept of *cycle formation model* is explained as follows: a cycle formation model of degree k ($k \geq 1$) is governed by k link generation mechanisms, $g(1), g(2), \dots, g(k)$, which are each described by c_1, c_2, \dots, c_k . If $P_{v_i v_j k}$ shows a path from v_i to v_j with length k , then $c_k = P((v_i, v_j) \in E | |P_{v_i v_j k}| = 1)$ (the probability that there is a link between i and j , given that there is one path of length k between them). We know that if there are more than one path with length k from v_i to v_j , then the probability that there is a link between them increases (See Figure 4 for instance). Therefore:

$$\begin{aligned} \text{if } P((v_i, v_j) \in E | |P_{v_i v_j k}| = 1) = c_k \ \& \ |P_{v_i v_j k}| = m \rightarrow \\ P((v_i, v_j) \in E | |P_{v_i v_j k}| = m) &= \frac{c_k^m}{c_k^m + (1 - c_k)^m} \end{aligned} \quad (52)$$

And due to the fact that total link occurrence probability between v_i and v_j is a result of the effect of multiple mechanisms of cycle formation model of degree k ($CF(k)$) is calculated by:

$$\begin{aligned} P_{m_2, \dots, m_k} = P((v_i, v_j) \in E | |P_{v_i v_j 2}| = m_2, \dots, |P_{v_i v_j k}| = m_k) = \\ \frac{c_1 c_2^{m_2} \dots c_k^{m_k}}{(c_1 c_2^{m_2} \dots c_k^{m_k}) + (1 - c_1)(1 - c_2)^{m_2} \dots (1 - c_k)^{m_k}} \end{aligned} \quad (53)$$

5 Relational Models

One drawback of previously mentioned methods is that they don't incorporate vertex and edge attributes to model the joint probability distribution of entities and links that associate them [45]. Probabilistic Relational Models (PRM) [42] are an attempt to use the rich logical structure of the underlying data that is crucial for complicated problems. One major limitations of Bayesian networks is the lack the concept of an "object" [43]. Bayesian PRMs [41, 42] include the concept of object in the context of Bayesian networks, in which each object can have their attributes and relations exist between objects and their attributes. Figure 5 is an example of a schema for a simple domain. A relational model consists of a set of *classes*, $\Upsilon = \{Y_1, Y_2, \dots, Y_n\}$. In Figure 5, $\Upsilon = \{\text{Journalist}, \text{Newspaper}, \text{Reader}\}$. Each class also contains some descriptive attributes, the set of which is shown with $A(Y)$. For example, Journalist has attributes *Popularity*, *Experience*, and *Writing skills*. In order for objects to be able to refer to other objects, each class is also associated with a set of *reference slots*, which is shown by $Y.\rho$. *Slot chains* also exist which are references between multiple objects (similar to $f(g(x))$). $Pa(Y.A)$ shows the set of *parents* of $Y.A$. For instance, in Figure 5, a journalist's *Popularity* depends on her *Experience* and *Writing skills*. Dependency can also be a result of a slot chain, meaning

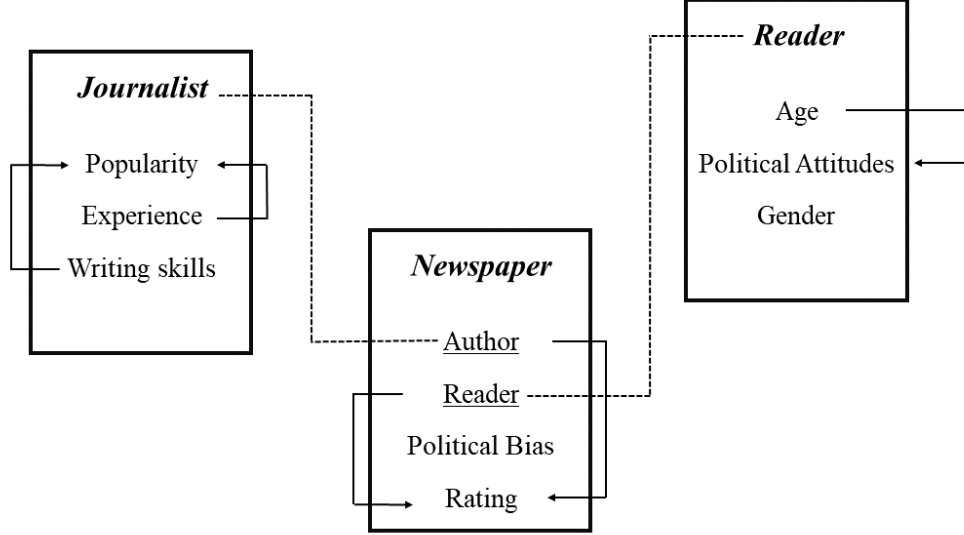


Figure 5: An example of a relational schema for a simple domain. The underlined attributes are references slots of the class and the arrows show the types of objects to which they are referring.

some attributes of a class depend on some attributes of another class. The joint probability distribution in a *PRM* can be calculated as follows [43]:

$$P(I|\sigma_r, S, \theta_S) = \prod_{Y_i} \prod_{A \in A(Y_i)} \prod_{y \in \sigma_r(Y_i)} P(I_{y.A} | I_{Pa(y.A)}) \quad (54)$$

In Equation 54, I shows an instance of a schema S , which specifies for each class Y , the set of objects in the class, a value for each attribute $y.A$, and a value for each reference slot $y.\rho$. Also, σ_r is a *relational skeleton* which denotes a partial specification of an instance of a schema, and specifies the set of objects for each class and the relations that hold between the objects [43].

The task of link prediction can then be performed by considering the probability of existence of a link between two objects in the relational model [44]. The work of [102] show that deriving the distribution of missing descriptive attributes will benefit estimation of link existence likelihood. Besides a Relational Bayesian Network in which the model graph is a directed acyclic graph, Relational Markov Network is also proposed [103, 104] in which the graph model is an undirected graph and can be utilized for the task of link prediction. Relational Markov Networks address two shortcomings of directed models: They don't constraint the graph to be acyclic, which allows for various possible graph representations. Also, they are well suited for discriminative training [105].

There exist other relational models for the task of link prediction. DAPER model is a directed acyclic type of probabilistic entity-relationship model [106]. The advantage of DAPER model is its being more expressive than aforementioned models [107]. Other Bayesian relational models in the literature include stochastic relational model [20] which models the stochastic structure of entity relationships by a tensor of multiple Gaussian processes [47], relational dependency network [108, 109], and parametric hierarchical Bayesian relational model [110].

6 Learning-based Methods

The feature extraction-based methods that are discussed earlier in this paper, provide a starting point for the systematic prediction of missing and/or future associations available through learning the effective attributes. Among the effective features for link prediction, employing the topological attributes that can be extracted from the graph structure is the foundation of all learning-based link prediction algorithms, from which the pair-wise shortest distance attribute is the most common topological feature. Besides the topological attributes, some machine learning models benefit from the node and domain specific attributes, referred to as the aggregated and proximity features, respectively [111].

Introduction of supervised learning algorithms to the problem of link prediction led to the state-of-the-art models that achieve high prediction performances [112]. These models view the problem of link prediction as a classification task.

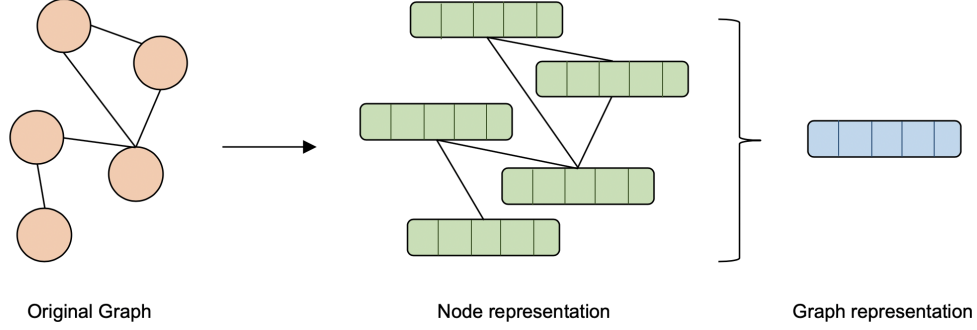


Figure 6: **An example of node and graph representation.** Here the node representation vectors are aggregated to generate a single graph representation.

To approach the link prediction problem, supervised models are supposed to tackle a few challenges, including the unbalanced data classes resulting from the sparsity property of real networks, and the extraction of the topological, proximity, and aggregated attributes as independent informative features [113]. There is an extensive literature on classification models for link prediction, including the application of traditional machine learning methods into this field of research. Support Vector Machines, K-nearest Neighbors, Logistic Regression, Ensemble Learning and Random Forrest, Multilayer Perceptron, Radial Basis Function network, and Naive Bayes are just a few supervised learning methods that are extensively used in link prediction. A comparison between a few of these supervised methods have been presented in [111], where surprisingly, SVM with RBF kernel is reported very successful in the accuracy and low squared error of the model.

Although the traditional machine learning models for link prediction rely on user-defined feature encoding, the evolution of these models has led to the generation of automatic feature encoders, which prevent hand-engineered attributes [113]. These models aim to learn graph encoding, node, and/or domain-related features into low-dimensional space. These machine learning algorithms for graphs are referred to as representation learning or graph embedding-based models for link prediction, which can be trained using neural networks or dimensionality reduction algorithms [114]. Representation learning algorithms try to preserve the structure of the embedded graph in the low dimensional vector space by preserving the node roles in the graph and/or the graph structure, for example via keeping the neighboring nodes closer to each other [115].

Mapping the graph to a vector space is also known as encoding. On the contrary, the reconstruction of the node neighborhood from the embedded graph is referred to as decoding. Graph representation can be learned via supervised or unsupervised methods using an appropriate optimization algorithm to learn the embeddings [113]. This mapping can be defined for graph $G = \langle V, E \rangle$ as $f : v_x \rightarrow v_{x'} \in \mathbb{R}^d, \forall x \in [n]$ such that $d \ll |V|$, where n denotes the total number of vertices, v_x is a sample node that has been embedded to d -dimensional vector space, and the embedded node is represented by $v_{x'}$.

Optimizing the graph mappings consists of joint optimization of the encoder and the decoder, which leads to learning graph feature preservative transformation [113]. The most commonly used optimization algorithm for this problem is stochastic Gradient Descent algorithm. The decoding function receives a set of node embeddings as input to decode the graph statistics or class information by reconstructing the node neighborhood. A pairwise decoder that maps a pair of embedded nodes to a real value measurement of the proximities based on the original graph neighborhoods can be defined as follows [113]:

$$DEC(DEC(v_x), DEC(v_y)) = DEC(v_{x'}, v_{y'}) \approx s_L(v_x, v_y), \quad (55)$$

where s_L refers to the class information or graph statistics for the two nodes v_x and v_y in the original graph G , and $v_{x'}$ and $v_{y'}$ are the corresponding embeddings for these nodes, respectively. The mapping by the decoder can be denoted as $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$ [113].

Diverse graph embedding techniques have been proposed in recent years, and thus, the representation learning algorithms for link prediction can be categorized from different perspectives. Possible divisions of algorithms can be according to the their applications, objective function, the properties of the embedded graph, or the features that the models consider. The first possible division can be (i) distance preserving algorithms, and (ii) structure preserving methods [116]. The former category includes models that construct an objective functions to achieve similar rankings for the distances between the original graph and the embedded version, according to higher order distances. Examples of algorithms

belonging to this category are Laplacian Eigenmaps [117], Graph Factorization [118], GraRep [119], and HOPE [120]. However, the latter category includes algorithms that aim to preserve the roles of nodes within the graph and its embedding, by either using random walks, deep auto-encoders, or Graph Convolutional Networks (GCN). Examples of models for this category are node2vec [114], DNGR [121], SDNE [122], VGAE [123], and graph semi-supervised algorithms and convolutional neural networks [124, 125, 126].

Although many natural graphs and application domains, such as online social networks, involve dynamic characteristics and temporal features, effective algorithms for the modeling of dynamic graphs is still lacking. Resulted from the increasing interest in modeling dynamic graphs, recent algorithms such as dyngraph2vec [116], DynGEM [127], DynGraphGAN [128], and [129] have emerged which focus on the learning of temporal features related to node associations in order to capture the underlying network dynamics of evolution. Thus, the second possible division for representation algorithms for link prediction can be (i) static graph embedding, and (ii) dynamic graph embedding. Common approaches in dynamic graph embedding methods are to initialize the embedding for the current time step according to the previous time step, or to use incremental Singular Value Decomposition (SVD) to update the embeddings [116].

Representation learning algorithms for machine learning can be divided into categories based on their decoder function, similarity measure for graphs, and the loss function in the models [113]. Therefore, the third possible categorization for these algorithms includes (i) Matrix Factorization Based Models, (ii) Random Walk Based Models, and (iii) Deep Neural Network Models. Link prediction studies based on these representation learning methods can be viewed as either generative or discriminative models or a combination of the two. The generative models for graph embedding learn the representation of the graph according to the underlying features and the distribution of these associations. A few generative models consider both node and edge formations by directly learning over the adjacency matrix of the graph. An example is GraphRNN [130]. In discriminative graph embedding methods, the models learn to predict the probability of edge existence between nodes [131]. For the rest of this paper, we focus on the third method of categorization to provide a detailed review on learning-based link prediction models.

6.1 Matrix Factorization-Based Methods

In these models, the vector representation of the topology-related features produces an N -dimensional space, where $N = |V|$ is the number of vertices in the network. The main purpose for matrix factorization-based methods is to reduce the dimensionality while preserving the nonlinearity and locality of the graph, via employing deterministic measures of node similarity in the graph. However, the global structure of the graph topology may be generally lost [132]. SVD is one of the commonly used methods as a result of its feasibility in low-rank approximations [133, 120]. Here, the link function $L(\cdot)$ is defined as $G \approx L(U \cap U^T)$, where $U \in \mathbb{R}^{|V| \times k} \cap \mathbb{R}^{k \times k}$, where k denotes the number of latent variables in SVD. The similarity between the node pairs $s(v_x, v_y)$ is defined by $L(u_{v_x}^T \cap u_{v_y}^T)$. Since, the methods based on graph embedding techniques using inner product decoders aim to follow and improve of one of the earliest dimensionality reduction techniques, Laplacian Eigenmaps, we look into the details regarding these methods.

6.1.1 Laplacian Eigenmap Methods

The original algorithm that was proposed by Belkin and Niyoki in [117] aims to expand the manifold for the data. This algorithm, first, constructs the graph via connecting nodes to their K nearest neighbors. Then, the weight W_{ij} is calculated for the connection between the pair of nodes based on the heat kernel. The algorithms is followed by minimizing a loss function for the graph embedding using the weight of node pairs. The decoder in the encoder-decoder framework of the Laplacian Eigenmaps can be defined as:

$$DEC(v_{x'}, v_{y'}) = \|v_{x'} - v_{y'}\|_2^2, \quad (56)$$

where the weights for pairs of nodes are calculated by the loss function based on the node pair similarity in the graph:

$$\mathcal{L} = \sum_{(v_x, v_y) \in D} DEC(v_{x'}, v_{y'}) \cdot s_G(v_x, v_y). \quad (57)$$

6.1.2 Inner-product Methods

The inner-product-based embedding models for link prediction embed the graph based on a pairwise inner-product decoder such that the node relationship probability is proportional to dot product of node embeddings:

$$DEC(v_{x'}, v_{y'}) = v_{x'}^T v_{y'}, \quad (58)$$

$$\mathcal{L} = \sum_{(v_x, v_y) \in D} \|DEC(v_{x'}, v_{y'}) - s_G(v_x, v_y)\|_2^2. \quad (59)$$

Examples of the inner-product-based methods for link prediction are Graph Factorization (GF) [118], GraRep [119], and HOPE [120] algorithms. graph factorization model partitions the graph by minimizing the number of neighboring nodes rather than applying edge cuts, as the storage and exchange of parameters for the latent variable models and their inference algorithms are related to nodes. The loss function in GF is the same as in inner-product-based methods, where the stochastic gradient descent algorithm is used for optimization. Therefore, this algorithm is also very efficient to be used for large networks. However, the difference between GF and inner-product-based methods is in the extraction of node similarity directly from the adjacency matrix.

GraRep [119] uses graph global structural information to capture the long distance relationships between nodes, and to discover distinct edges in terms of the transition steps required to traverse a graph between two nodes in k steps. This model reduces the dimension of the vector space besides learning the global topological structure. Using the degree matrix D and the adjacency matrix A , the probability transition matrix can be defined as $M = D^{-1}A$, and The k -step probability transition matrix can be calculated by $M^k = \underbrace{M \dots M}_k$.

Another model developed from the inner-product-based algorithms is HOPE [120], which focuses on the representation and modeling of directed graphs, as any types of graph can be represented with directed associations. This model preserves the asymmetric transitivity for directed graph embeddings. Asymmetric transitivity property captures the structure of the graph by keeping the correlation between the directed edges, such that the probability of the existence of a directed edge from v_x to v_y is high if a directed edge exists for the opposite direction. The asymmetric transitivity property has also application in decoding the embedded graph features. The HOPE model preserves asymmetric transitivity via high-order proximity approximation. A high-order proximity matrix S can be defined as $S = M_g^{-1}M_l$, where M_g and M_l are polynomial matrices. To approximate the high order proximities in this model, a Singular Value Decomposition is applied on the proximity matrix and the optimized vector representations are constructed using the singular values. HOPE supports classical similarity measures as proximity measurements in the algorithm, including the Katz Index (KI), Rooted PageRank (RPR), Common Neighbors (CN), and Adamic-Adar (AA).

6.2 Random Walk-Based Methods

The developed models for link prediction that are designed based on random walks statistics prevent the need to any deterministic similarity measures. In this algorithms, similar embeddings are being produced for nodes that co-occur on graph short random walks. These algorithms investigate the node features, including node centrality and similarity via graphs exploration and sampling with random walks or search algorithms such as Breadth First Search (BFS) and Depth First Search (DFS) [115]. The importance of exploring graphs with search algorithms is more obvious for huge graphs including graphs of social networks to decrease the complexity by limiting node and edge options. The random walk-based models for graphs can be divided into many different categories, according to varying perspectives. One possible deviation for these models include categorization based on their embedding output, for instance, local structure preserving methods, global structure preserving methods, and the combination of the two [134]. Representations with BFS provide information about similarity of nodes in case of their roles in the network, for instance representing a hub in the graph [114]. On the contrary, random walks with DFS can provide information about the communities that nodes belong to. These algorithms have been recently applied along with generative models to introduce edges and nodes directly to the graph [131]. In this paper, we categorize random walk-based methods for graph representation learning based on applying direct search on the graph or operating the search after graph preprocessing. Examples of the random walk-based methods for link prediction are DeepWalk [135], Node2vec [114], and HARP [136].

6.2.1 Random and Biased Search Methods

DeepWalk [135] approaches the graph representation learning by introducing advancements in natural language modeling and unsupervised learning to graphs, through treating walks as sentences. To achieve a superior performance, DeepWalk algorithm considers relational classification, also referred to as collective classification. However, instead of traditional inference-based approaches to relational classification, such as undirected Markov networks with Gibbs Sampling, DeepWalk separates the structural representation from labels. By applying uniform random walks to a set of random vertices using a stream of short walks with a specific length, the nodes' neighborhood and community information are being produced. Generally, the idea in DeepWalk algorithm is to minimize an objective function for cross-entropy loss as:

$$\mathcal{L} = \sum_{(v_x, v_y) \in D} -\log(\text{DEC}(v_x, v_y)), \quad (60)$$

where D is produced from sampled random walks. The embeddings in this algorithm are being roughly learned by:

$$DEC(v_{x'}, v_{y'}) \triangleq \frac{e^{v_{x'}^T v_{y'}}}{\sum_{v_k \in V} e^{v_{x'}^T v_k}} \approx p_{G,K}(v_y | v_x). \quad (61)$$

Starting a length- K random walk from node v_x , the probability of visiting node v_y is $p_{G,K}(v_y | v_x)$, and $K \in 2, \dots, 10$. However, DeepWalk employs the SkipGram optimization model for the objective function to train and learn the graph representations in order to reduce computational complexity, while skipGram is mostly designed for language processing.

Node2vec [114] extends DeepWalk via biased random walks and flexible exploration of node neighborhoods, which lead to learning richer representations. This model approaches graph embedding by a combination of BFS and DFS search algorithms to learn both the community structure and local structure. Node2vec learns the representation of the edges via embedding pairs of nodes by second order random walk and bagging the embedded features of the individual nodes. The random walks in Node2vec are biased as a result of sampling the next node according to the static edge weights. For this reason, two additional hyperparameters have been introduced, p and q , which affect the likelihood and speed of walk exploration and exiting the one-hop neighborhoods.

The GraphGAN model, proposed in [131] contains an online edge generator that adds new edges to the structure of the graph based on random walk starting from a single node v_x . Additionally, the framework for GraphGAN consists of a discriminator, which deals with the problem of link prediction by learning the features of the authentic edges. Thus, GraphGAN bridges the discriminative and generative models for network evolution. The distribution of underlying true connectivity for a vertex v_x is defined as a conditional probability $p_{true}(v | v_x)$ that calculates the distribution of preferred connections for the vertex, over all vertices $v \in V$. The generator aims to generate vertices similar to the real immediate neighbors of a vertex v_x to mislead the discriminator.

6.2.2 Graph Preprocessing-based Methods

HARP [136] is a multi-level graph representation learning algorithm that prevents embedding configuration issues by compressing the graph before learning its representation. Creating a series of compressed graphs with different compression levels allows HARP to embed the hierarchy of graphs descending by size. The smaller graphs, also referred to as coarser graphs, in the series of compressed versions G_0, G_1, \dots, G_L , preserve graph's global structure, where $G_0 = G$ and G_L is the coarsest graph. The coarsening algorithm results in the compression of graphs to "super nodes" to be used as inputs to the random walk-based algorithms such as DeepWalk and LINE [132]. Accordingly, the compressed graph produced by the HARP algorithm approximates the global structure of the original graph. The algorithm includes graph coarsening, graph embedding on the coarsest graph G_L , graph representation prolongation and refinement, and graph embedding of the original graph G .

6.3 Neural Network-Based Methods

The graph representation learning methods that embed nodes independently, result in many deficiencies, including inefficient algorithms with high computational complexity, failing to consider the attributes for nodes, and failing to generalize to new graphs or to model evolving graphs [137]. Therefore, the application of neural networks for graphs have gained extensive interest. This evolution has led to the generation of a family of complex encoder-decoder-based representation learning models, such as Graph Neural Networks (GNNs) [138] and Graph Convolutional Neural Networks (GCNs) [124, 123, 139, 140]. Although the general concept of graph neural networks was first presented in [138], many of neural network-based algorithms for representation learning have been proposed, in which many are able to learn the general structure and attributes of graphs, including the node attributes. Furthermore, efficient strategies to capture multi-modality for graphs, for instance node heterogeneity, have been originated from neural network-based models. Another extension for graph embedding methods that have become achievable by neural networks, is the embedding of subgraphs ($S \subset V$). The attribute aggregation procedure in different neural network architectures may vary according to their connection types, and the usage of filters or gates in the propagation step of the models [134].

In order to learn the information on the neighboring nodes, GNNs [138] aim to learn a state embedding $h_{v_x} \in \mathbb{R}^s$ iteratively, where s is the dimension for the vector representation of node v_x . By stacking the states for all nodes, the constructed vectors H , and the output labels O can be represented as:

$$H = F_g(H, X), \quad (62)$$

$$O = O_g(H, X_N), \quad (63)$$

where F_g is the global transition function, O_g is the global output function, X refers to the feature vector, and X_N stands for the feature vector for all nodes. The updates per iteration can be defined as:

$$H^{t+1} = F(H^t, X), \quad (64)$$

where t denotes the t -th iteration. In this algorithm, learning of the representations can be via a supervised optimization method, such as the gradient-descent method.

Multi-layer Perceptrons (MLPs) are neural network-based representation learning algorithms that approach graph embedding via message passing, in which information flows from the neighboring nodes with arbitrary depth. Message Passing Neural Networks (MPNNs) [141] further extend GNNs and GCNs with proposing a single framework for variants of general approaches, such as incorporating the edge features in addition to the node features.

Another category of graph neural networks are Graph Auto-Encoders (GAE) and Variational Graph Auto-Encoders (VGAE) [123] that aim to learn the node representations in an unsupervised manner. Majority of models based on GAE and its derivations employ Graph Convolutional Networks (GCNs) for the node encoding procedure. Next, these algorithms employ a decoder to reconstruct the graph’s adjacency matrix A . This procedure can be formally represented as:

$$Z = GCN(X, A), \quad (65)$$

where Z is the convolved attribute matrix. GAEs can learn the graph structures using deep neural network architectures, and reduce the graph dimensionality in accordance with the number of channels of the auto-encoder hidden layers [121]. Additionally, GAE-based models are able to embed the nodes into sequences with diverse lengths. This benefits the auto-encoders to not only achieve high performances for testing over the unseen node embeddings, but also aggregate the node attributes to improve their prediction accuracy [113]. Examples of representation models with auto-encoder architectures are GC-MC [140] and Adversarially Regularized Graph Auto-Encoders (ARGA) [142]. Auto-encoders are also being used without neural network architectures, for instance, LINE [132], DNGR [121], and SDNE [122]. The algorithm in LINE consists of a combination of two encoder-decoder structures to study and optimize first and second node proximities in the vector space. Both of the DNGR [121] and SDNE [122] algorithms embed the node local neighborhood information using a random surfing method and approach single embeddings through auto-encoders than pairwise transformations.

Although the graph representation learning models based on GNNs consider both graph structure and node features to embed the graph, they suffer from computational complexity and inefficiency in iterative updating of the hidden states. Furthermore, GNNs use the same parameters for all layers, which limits their flexibility. These architectures are always designed as shallow networks with no more than three layers, and including a higher number of layers is still being considered as a challenge for CNNs [134].

6.3.1 Graph Convolutional Networks

The introduction of neural networks, specially convolutional neural networks to graph structures have led to extract features from complex graphs flexibly. Graph Convolutional Networks (GCNs) [124] tackle the problem of high computational complexity and shallow architectures via defining a convolution operator for the graph. Furthermore, a rich class of convolutional filter functions can be achieved through stacking many convolution layers. The iterative aggregation of a node’s local neighborhood is being used in GCNs to obtain graph embeddings, where this aggregation method leads to higher scalability besides learning graph global neighborhoods. The features for these models include the information from the topology of the network aggregated by the node attributes, when the node features are available from the data domain [134]. Additionally, GCNs can be utilized for node embeddings, as well as subgraph embeddings [113]. Varying convolutional models have been derived from GCNs that employ different convolutional filters in their architecture. These filters can be designed either as spatial filters or spectral filters. The former type of convolutional filters can be operated directly on the original graph and its adjacency matrix, however, the latter type is being utilized on the spectrum of the graph Laplacian [115].

In [143] the problem of link prediction is studied using a combination of two convolutional neural networks for the graph network of molecules. The molecules are represented having a hierarchical structure for their internal and external interactions. The graph structure transformation to a low dimensional vector space is obtained from an internal convolutional layer which is randomly initialized for each node representation and trained by backpropagation. The external convolutional layer receives the embedded nodes as input to learn over the external graph representations. Finally, the link prediction algorithm consists of a multilayer neural network which was accepting the final representations to predict the molecule-molecule interactions by a softmax function.

The algorithms that belong to the family of neighborhood aggregation methods, are also being referred to as convolutional models. An example is GraphSAGE [137], which aggregates the information from local neighborhoods recursively, or iteratively. This iterative characteristic leads the model to be generalizable to unseen nodes. The node attributes for this

model might include simple node statistics such as node degrees, or even textual data for profile information on online social networks. IN GraphSAGE, the evolution of the network is considered for both links and nodes in the graph. Starting from a node, GraphSAGE samples a uniform number of immediate neighboring nodes to collect their local features and map them to feature vectors. GraphSAGE concatenates the node's current representation for a different search depth, for example depth equal to 1, and for a different number of hops. This concatenation continues for the same feature vectors until a defined depth K is met. Then, the results can be fed into a supervised model, such as a fully connected neural network to learn the aggregator's weights. The aggregator architecture is either Mean aggregator, LSTM, or Pooling[137]. The difference between GCN and GraphSAGE is in the node aggregation function; GCN employs element-wise mean followed by weighted sum for this aggregation.

6.3.2 Graph Gated Neural Networks

An extension of GNN models is the introduction of gate mechanisms, such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM), in the propagation step of neural network architectures. This extension has led to achieve a better flow of information. For instance, Gated Graph Neural Network (GGNN) [144] employs GRUs besides optimization techniques to generate sequence output. This extension has resulted in flexible models to avoid recurrence for a fixed number of iterations. Additionally, the introduction of gates allows using the output of the intermediate embeddings of subgraphs. The update functions in this model updates the hidden state h for each node v via incorporating information from the previous time steps and from the other neighboring nodes. The propagation step of models with GRUs can be formulated as:

$$\begin{aligned} a_v^t &= A_v^T [h_1^{t-1} \dots h_N^{t-1}]^T + b, \\ z_v^t &= \sigma(W^z a_v^t + U^z h_v^{t-1}), \\ r_v^t &= \sigma(W^r a_v^t + U^r h_v^{t-1}), \\ \tilde{h}_v^t &= \tanh(W a_v^t + U(r_v^t \odot h_v^{t-1})), \\ h_v^t &= (1 - z_v^t) \odot h_v^{t-1} + z_v^t \odot \tilde{h}_v^t, \end{aligned} \tag{66}$$

where A_v refers to the associations for node v with its neighboring nodes and is a submatrix of the adjacency matrix A . The information aggregation for neighbors of node v is being operated by a , which contains the activations from the edges for both directions. \odot is element-wise matrix multiplication. The update and reset gates are represented by z and r , respectively. Finally, σ is the logistic sigmoid function which can be calculated by $\sigma(x) = 1/(1 + e^{-x})$.

Other examples of gated neural network architectures for graph are Child-Sum Tree-LSTM and N-ary Tree-LSTM proposed in [145], and graph-structured LSTM proposed in [146].

6.3.3 Hierarchical Graph Embedding methods

Although the application of neural networks in graph representation learning resulted in covering many complex features of graphs, many algorithms have only considered the flattened graphs instead of learning the hierarchical structure of the graphs. In other words, the non-local dependencies in high dimensional nature of the graphs cannot be captured and modeled by most of the representation learning methods. The pooling layers between the convolutional layers, used widely in the field of computer vision, are very effective in capturing more general attributes [134]. Thus, many proposed algorithms have aimed to introduce pooling layers to graph analysis. Examples of representation learning models that have addressed hierarchies of subgraphs, with or without pooling layers, are Edge-Conditional Convolution (ECC) [147], DIFFPOOL [148], and GraphRNN [130].

ECC [147] generalizes regular convolution layers for graphs and extends GCNs to deep neural network architectures with filter weights that are conditioned on the edges within the local neighborhood of a vertex. For a neural network with $l \in \{0, \dots, l_{max}\}$ layers, node attribute matrix X , edge attribute matrix L , and edge-specific weight matrix Θ , the convolution operation, referred to as the Edge-Conditioned Convolution, can be formulated as:

$$X^l(v_x) = \frac{1}{|\Gamma(v_x)|} \sum_{v_y \in \Gamma(v_x)} \Theta_{v_y, v_x}^l X^{l-1}(v_y) + b^l \tag{67}$$

where $\Theta_{yx}^l = F^l(L(v_y, v_x); \omega^l)$, $\Gamma(v_x)$ denote the neighboring nodes for vertex v_x , b^l is the learnable bias, and F^l is parameterized by the learnable network weights ω^l . In this model, the information from the local neighborhoods is being combined for successive layers. The pooling module in ECC has designed such that it downsamples the graph recursively, where the sign of the largest Laplacian eigenvector is being employed for graph splitting in the downsampling procedure.

DIFFPOOL [148] This model facilitates learning graph representations of complex structures and their deep features via introducing differentiable pooling with learnable assignments to the field of graph analysis. To achieve this, DIFFPOOL extends GNNs and ECC by including a hierarchical clustering module into the pooling procedure to learn the hierarchies of subgraphs. In this algorithm, the cluster of embedded nodes is being used as the coarsened input to the next GNN layer, leading to learn complex graph representations instead of learning flattened graphs. The graph coarsening procedure is being operated by the DIFFPOOL layer as:

$$(A^{l+1}, X^{l+1}) = DIFFPOOL(A^l, X^l), \quad (68)$$

which generates the new adjacency matrix A^{l+1} and the new node feature matrix X^{l+1} . These outputs are then being used as inputs to the next layer, and the procedure is operated for each of the nodes or clusters of the coarsened graph. The assignment matrix that is being learned in each layer l can be represented as:

$$S^l = softmax(GNN_{l,pool}(A^l, X^l)), \quad (69)$$

where the softmax is being used for row-wise operation. Finally, the standard GNN operation for layer l can be formulate by:

$$Z^l = GNN_{l,embed}(A^l, X^l). \quad (70)$$

GraphRNN [130] is a deep model for graph representation learning with LSTM model, that learns to generate large scale graphs as a joint sequence of node and edge formation, either in an autoregressive manner or recursively. The hierarchy structure achived by the GraphRNN algorithm is due to the graph-level Recurrent Neural Network (RNN) architecture that generates new nodes, besides the edge-level RNN architecture that introduces edges to the sets of nodes. Therefore, GraphRNN can also be classified as a graph gated neural network model with GRUs. The model consists BFS algorithm to limit the number of edge prediction and accordingly, alleviate the computational complexity as a result of node ranking in the edge formation procedure. GraphRNN applies a mapping function to the graph's adjacency matrix to gain the sequences of nodes. The adjacency matrix including node ordering π for node mapping can be represented by $A_{i,j}^\pi = 1[(\pi(v_i), \pi(v_j)) \in E]$. The mapping from graph G to sequences Sec with n nodes, and with node ordering π can be formulated as:

$$Sec^\pi = f_{Sec}(G, \pi) = (Sec_1^\pi, \dots, Sec_n^\pi), \quad (71)$$

where each element of Sec^π is a vector of the adjacency matrix A for node $\pi(v_i)$ in connection with the previous nodes $\pi(v_j)$, $j \in \{1, \dots, i-1\}$ as:

$$Sec_i^\pi = (A_{1,i}^\pi, \dots, A_{i-1,i}^\pi)^T, \forall i \in \{2, \dots, n\}. \quad (72)$$

Using a generative model, GraphRNN learns the probability of graph embedding as a joint distribution $p(G, Sec^\pi)$, where $p(Sec^\pi)$ can itself be decomposed into product of conditional distributions for its elements. The RNN model consists of a state transition function and an output function as:

$$h_i = f_{trans}(h_{i-1}, Sec_{i-1}^\pi), \quad (73)$$

$$\theta_i = f_{out}(h_i), \quad (74)$$

where h_i represents the state of the generated graph, and θ_i specifies the distribution of the adjacency vector for the next node. The generated sequences can be fed into a neural network architecture for the transition fuction f_{trans} , for instance GraphRNN with multivariate Bernoulli using a single layer multilayer perceptron (MLP), or a dependent Bernoulli sequence model with another RNN to generate the edges.

6.3.4 Graph Attention Networks

The incorporation of the attention mechanism to the graph neural networks resulted in the emergence of models that are applicable to both inductive and transductive problems. This is achieved via extending graph convolutions to include attention layers, for instance, masked self-attentional layers and stacking these layers with varying wights for nodes in local neighborhoods. Graph Attention Network (GAT) [149] is an efficient convolution-style natural network that assigns varying importances to different nodes within a neighborhood. In this model, a single graph attentional layer which can be stacked for deep architectures is being porposed to compute the coefficients for the attention mechanism of a pair of nodes as:

$$\alpha_{v_x, v_y} = \frac{\exp(LeakyReLU(a^T [Wh_{v_x}] [Wh_{v_y}])))}{\sum_{v_z \in \Gamma(v_x)} \exp(LeakyReLU(a^T [Wh_{v_x}] [Wh_{v_z}])))}, \quad (75)$$

where α_{v_x, v_y} refers to the attention coefficient from node v_y to v_x . Having the set of node features $h = h_1, \dots, h_N$, the output of this layer is the weight matrix W for shared linear transformation for all nodes. Additionally, a new set

of node features $h' = h'_1, \dots, h'_N$ with different cardinality is being generated. Finally, a denotes the weight vector for a single feedforward layer in the neural network architecture, where LeakyReLU has been employed to introduce non-linearity. The new node features can be calculated by:

$$h'_{v_x} = \sigma \left(\sum_{v_y \in \Gamma(v_x)} \alpha_{v_x, v_y} W h_{v_y} \right). \quad (76)$$

Other proposed models for attention mechanisms are Gated Attention Networks (GAAN) [150], Heterogeneous Graph Attention Networks (HGAN) [151], and Knowledge Graph Attention Network for Recommendation (KGAT) [152].

7 Network Data Sets

One of the challenging tasks in network research is the implementation and validation of the proposed methods and models. In the majority of the network research, the popular collections of data sets are used as a common sense: A friendship network of 34 members of a Karate Club and 78 interactions among them [153], the power network of an electrical grid of western US with 4941 nodes and 6594 edges [154], an internet-based router network with 5022 nodes and 6258 edges [155], a protein-protein interaction network that contains 2617 proteins and 11855 interactions [156], a collaboration network of 1589 authors with 2742 interactions [157], an airline network of 332 nodes and 2126 edges that show the connection between airports², a social network of 62 dolphins in New Zealand with 159 interactions [158], a biological network of cerebral cortex of Rhesus macaque with 91 nodes and 1401 edges [159].

Data set collection is a time-consuming and labor-intensive work. While some studies build their own data set, researchers mostly prefer to employ an existing data set. Some popular collections of network data sets that might be used in link prediction studies are as follows:

- SNAP[160]: A collection of more than 90 network data sets by Stanford Network Analysis Platform. With biggest data set consisting of 96 million nodes.
- BioSNAP[161]: More than 30 Bio networks data sets by Stanford Network Analysis Platform
- KONECT[162]: This collection contains more than 250 network data sets of various types, including social networks, authorship networks, interaction networks, etc.
- PAJEK[163]: This collection contains more than 40 data sets of various types.
- Network Repository[164]: A huge collection of more than 5000 network data sets of various types, including social networks.
- Uri ALON[165]: A collection of complex networks data sets by Uri Alon Lab.
- NetWiki[166]: More than 30 network data sets collection of various types.
- WOSN 2009 Data Sets[167]: A collection of Facebook data provided by social computing group.
- Citation Network Data set[168]: A collection of citation network data set extracted from DBLP, ACM, and other sources.
- Grouplens Research[169]: A movie rating network data set.
- ASU social computing data repository[170]: A collection of 19 network data sets of various types: cheminformatics, economic networks, etc.
- Nexus network repository[171]: A repository collection of network data sets by iGraph.
- SocioPatterns[172]: A collection of 10 network data sets collected by SocioPatterns interdisciplinary research collaboration.
- Mark Newman[173]: A collection of Network data sets by Mark Newman.
- Graphviz[164]: An interactive visual graph mining and analysis.

8 Taxonomy

According to the methods explained earlier in this paper, we propose a taxonomy to better categorize the link prediction models. In our propose taxonomy, the link prediction techniques are mainly categorized under two sections: Feature learning and feature extraction techniques (Figure 7).

²<http://vlado.fmf.uni-lj.si/pub/networks/data/>

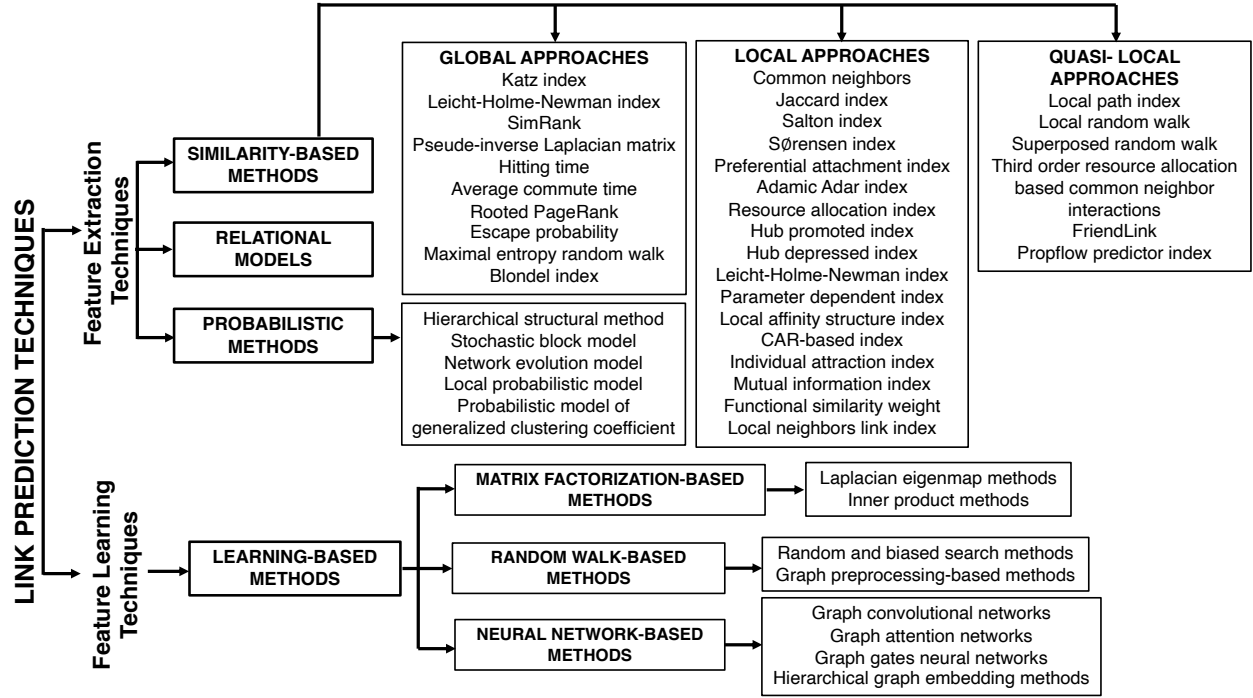


Figure 7: A taxonomy for the feature extraction techniques and feature learning methods in link prediction literature.

9 Discussion

Studying complex networks to predict the emerging links or missed associations is feasible through a variety of approaches discussed in this paper. Link prediction is an ongoing research problem, given its importance in many applications. New techniques with superior performance and high scalability and fast computation are expected to be proposed in the forthcoming future. To the best of our knowledge, this work presented the most comprehensive survey analysis on the link prediction problem in complex networks with a carefully-designed taxonomy presented in Figure 7. Finally, a set of network datasets consisting of the most popular and the most important ones, that can help in studying and implementing link prediction algorithms are introduced.

References

- [1] Frank Harary and Robert Z Norman. *Graph theory as a mathematical model in social science*. Number 2. University of Michigan, Institute for Social Research Ann Arbor, 1953.
- [2] Oliver Mason and Mark Verwoerd. Graph theory and networks in biology. *IET systems biology*, 1(2):89–119, 2007.
- [3] Ramë Likaj, Ahmet Shala, Merita Mehmetaj, Pajazit Hyseni, and Xhevahir Bajrami. Application of graph theory to find optimal paths for the transportation problem. *IFAC Proceedings Volumes*, 46(8):235–240, 2013.
- [4] TB Boffey. *Graph theory in operations research*. Macmillan International Higher Education, 1982.
- [5] P. Erdős and A. Rényi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.
- [6] Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.
- [7] Béla Bollobás. Random graphs. In *Modern graph theory*, pages 215–252. Springer, 1998.
- [8] Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.
- [9] William Aiello, Fan Chung, and Linyuan Lu. A random graph model for massive graphs. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 171–180. Acm, 2000.
- [10] Michał Karoński. A review of random graphs. *Journal of Graph Theory*, 6(4):349–389, 1982.
- [11] Agostino Di Ciaccio and Giovanni Maria Giorgi. Statistical analysis of social networks. *Rivista Italiana di Economia Demografia e Statistica*, 67(3/4):103–110, 2013.

- [12] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.
- [13] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.
- [14] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.
- [15] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *Link mining: models, algorithms, and applications*, pages 337–357. Springer, 2010.
- [16] Joshua S Weitz, Philip N Benfey, and Ned S Wingreen. Evolution, interactions, and biological networks. *PLoS biology*, 5(1):e11, 2007.
- [17] Sergey N Dorogovtsev and Jose FF Mendes. Evolution of networks. *Advances in physics*, 51(4):1079–1187, 2002.
- [18] Sergei N Dorogovtsev and José FF Mendes. *Evolution of networks: From biological nets to the Internet and WWW*. OUP Oxford, 2013.
- [19] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st international conference on World Wide Web*, pages 519–528. ACM, 2012.
- [20] Kai Yu, Wei Chu, Shipeng Yu, Volker Tresp, and Zhao Xu. Stochastic relational models for discriminative link prediction. In *Advances in neural information processing systems*, pages 1553–1560, 2007.
- [21] Stefan Stieglitz and Linh Dang-Xuan. Emotions and information diffusion in social media—sentiment of microblogs and sharing behavior. *Journal of management information systems*, 29(4):217–248, 2013.
- [22] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1019–1028. ACM, 2010.
- [23] Jaideep Srivastava. Data mining for social network analysis. In *Intelligence and Security Informatics, 2008. ISI 2008. IEEE International Conference on*, pages xxxiii–xxxiv. IEEE, 2008.
- [24] Pedro Domingos. Mining social networks for viral marketing. *IEEE Intelligent Systems*, 20(1):80–82, 2005.
- [25] Mike Thelwall, David Wilkinson, and Sukhvinder Uppal. Data mining emotion in social network communication: Gender differences in myspace. *Journal of the American Society for Information Science and Technology*, 61(1):190–199, 2010.
- [26] Mingrui Xia, Jinhui Wang, and Yong He. Brainnet viewer: a network visualization tool for human brain connectomics. *PloS one*, 8(7):e68910, 2013.
- [27] Bobby-Joe Breitkreutz, Chris Stark, and Mike Tyers. Osprey: a network visualization system. *Genome biology*, 4(3):R22, 2003.
- [28] James Moody, Daniel McFarland, and Skye Bender-deMoll. Dynamic network visualization. *American journal of sociology*, 110(4):1206–1241, 2005.
- [29] Nicholas A Christakis and James H Fowler. Social network visualization in epidemiology. *Norsk epidemiologi Norwegian journal of epidemiology*, 19(1):5, 2009.
- [30] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM Computing Surveys (CSUR)*, 49(4):69, 2017.
- [31] Han Hee Song, Tae Won Cho, Vacha Dave, Yin Zhang, and Lili Qiu. Scalable proximity estimation and link prediction in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 322–335. ACM, 2009.
- [32] Desislava Hristova, Anastasios Noulas, Chloë Brown, Mirco Musolesi, and Cecilia Mascolo. A multilayer approach to multiplexity and link prediction in online geo-social networks. *EPJ Data Science*, 5(1):24, 2016.
- [33] Mahdi Jalili, Yasin Orouskhani, Milad Asgari, Nazanin Alipourfard, and Matjaž Perc. Link prediction in multiplex online social networks. *Royal Society open science*, 4(2):160863, 2017.
- [34] Chengwei Lei and Jianhua Ruan. A novel link prediction algorithm for reconstructing protein–protein interaction networks by topological similarity. *Bioinformatics*, 29(3):355–364, 2013.
- [35] Nantia Iakovidou, Panagiotis Symeonidis, and Yannis Manolopoulos. Multiway spectral clustering link prediction in protein-protein interaction networks. In *Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine*, pages 1–4. IEEE, 2010.

- [36] Edoardo M Airolidi, David M Blei, Stephen E Fienberg, Eric P Xing, and Tommi Jaakkola. Mixed membership stochastic block models for relational data with application to protein-protein interactions. In *Proceedings of the international biometrics society annual meeting*, volume 15, 2006.
- [37] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. Beyond link prediction: Predicting hyperlinks in adjacency space. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [38] Yang Ma, Xingxing Liang, Jincai Huang, and Guangquan Cheng. Intercity transportation construction based on link prediction. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1135–1138. IEEE, 2017.
- [39] Bruce A Desmarais and Skyler J Cranmer. Forecasting the locational dynamics of transnational terrorism: A network analytic approach. *Security Informatics*, 2(1):8, 2013.
- [40] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *arXiv preprint arXiv:1802.09691*, 2018.
- [41] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.
- [42] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, volume 99, pages 1300–1309, 1999.
- [43] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. *Introduction to statistical relational learning*, 8, 2007.
- [44] Lisa Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3(Dec):679–707, 2002.
- [45] Mohammad Al Hasan and Mohammed J Zaki. A survey of link prediction in social networks. In *Social network data analytics*, pages 243–275. Springer, 2011.
- [46] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.
- [47] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.
- [48] Xu-Wen Wang, Yize Chen, and Yang-Yu Liu. Link prediction through deep learning. *bioRxiv*, page 247577, 2018.
- [49] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.
- [50] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.
- [51] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001.
- [52] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.
- [53] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.
- [54] Caroline S Wagner and Loet Leydesdorff. Mapping the network of global science: comparing international co-authorships from 1990 to 2000. *International journal of Technology and Globalisation*, 1(2):185–208, 2005.
- [55] Thorvald Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. Skr.*, 5:1–34, 1948.
- [56] Bruce McCune, James B Grace, and Dean L Urban. *Analysis of ecological communities*, volume 28. MjM software design Gleneden Beach, OR, 2002.
- [57] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.
- [58] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.
- [59] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.
- [60] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58(1):1–38, 2015.
- [61] Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *science*, 297(5586):1551–1555, 2002.

- [62] Elizabeth A Leicht, Petter Holme, and Mark EJ Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.
- [63] Yu-Xiao Zhu, Linyuan Lü, Qian-Ming Zhang, and Tao Zhou. Uncovering missing links with cold ends. *Physica A: Statistical Mechanics and its Applications*, 391(22):5769–5778, 2012.
- [64] Qingshuang Sun, Rongjing Hu, Zhao Yang, Yabing Yao, and Fan Yang. An improved link prediction algorithm based on degrees and similarities of nodes. In *Computer and Information Science (ICIS), 2017 IEEE/ACIS 16th International Conference on*, pages 13–18. IEEE, 2017.
- [65] Carlo Vittorio Cannistraci, Gregorio Alanis-Lobato, and Timothy Ravasi. From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific reports*, 3:1613, 2013.
- [66] Yuxiao Dong, Qing Ke, Bai Wang, and Bin Wu. Link prediction based on local information. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pages 382–386. IEEE, 2011.
- [67] Fei Tan, Yongxiang Xia, and Boyao Zhu. Link prediction in complex networks: a mutual information perspective. *PloS one*, 9(9):e107056, 2014.
- [68] Hon Nian Chua, Wing-Kin Sung, and Limsoon Wong. Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions. *Bioinformatics*, 22(13):1623–1630, 2006.
- [69] Christine Brun, François Chevenet, David Martin, Jérôme Wojcik, Alain Guénoche, and Bernard Jacq. Functional classification of proteins for the prediction of cellular function from a protein-protein interaction network. *Genome biology*, 5(1):R6, 2003.
- [70] Juan Yang, Lixin Yang, and Pengye Zhang. A new link prediction algorithm based on local links. In *International Conference on Web-Age Information Management*, pages 16–28. Springer, 2015.
- [71] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.
- [72] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.
- [73] David Liben-Nowell. *An algorithmic approach to social networks*. PhD thesis, Massachusetts Institute of Technology, 2005.
- [74] Daniel A Spielman. Spectral graph theory and its applications. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 29–38. IEEE, 2007.
- [75] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering*, 19(3):355–369, 2007.
- [76] Karl Pearson. The problem of the random walk. *Nature*, 72(1867):342, 1905.
- [77] Hanghang Tong, Christos Faloutsos, Christos Faloutsos, and Yehuda Koren. Fast direction-aware proximity for graph mining. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 747–756. ACM, 2007.
- [78] Linyuan Lü and Tao Zhou. Link prediction in weighted networks: The role of weak ties. *EPL (Europhysics Letters)*, 89(1):18001, 2010.
- [79] JH Hetherington. Observations on the statistical iteration of matrices. *Physical Review A*, 30(5):2713, 1984.
- [80] J Duda. *Extended maximal entropy random walk*. PhD thesis, Ph. D. dissertation, Jagiellonian University, 2012.[Online]. Available: <http://www.fais.uj.edu.pl/documents/41628/d63bc0b7-cb71-4eba-8a5a-d974256fd065>, 2012.
- [81] Rong-Hua Li, Jeffrey Xu Yu, and Jianquan Liu. Link prediction: the power of maximal entropy random walk. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1147–1156. ACM, 2011.
- [82] Vincent D Blondel, Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM review*, 46(4):647–666, 2004.
- [83] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. Similarity index based on local paths for link prediction of complex networks. *Physical Review E*, 80(4):046122, 2009.
- [84] Weiping Liu and Linyuan Lü. Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5):58007, 2010.

- [85] Jianpei Zhang, Yuan Zhang, Hailu Yang, and Jing Yang. A link prediction algorithm based on socialized semi-local information. *Journal of Computational Information Systems*, 10(10):4459–4466, 2014.
- [86] Alexis Papadimitriou, Panagiotis Symeonidis, and Yannis Manolopoulos. Fast and accurate link prediction in social networking systems. *Journal of Systems and Software*, 85(9):2119–2132, 2012.
- [87] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 243–252. ACM, 2010.
- [88] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003.
- [89] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98, 2008.
- [90] Anna Goldenberg, Alice X Zheng, Stephen E Fienberg, and Edoardo M Airolidi. *A survey of statistical network models*. Now Publishers Inc, 2010.
- [91] Roger Guimerà and Marta Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009.
- [92] Tiago P Peixoto. Hierarchical block structures and high-resolution model selection in large networks. *Physical Review X*, 4(1):011047, 2014.
- [93] Toni Vallès-Català, Tiago P Peixoto, Marta Sales-Pardo, and Roger Guimerà. Consistencies and inconsistencies between model selection and link prediction in networks. *Physical Review E*, 97(6):062316, 2018.
- [94] Roger Guimera and Marta Sales-Pardo. A network inference method for large-scale unsupervised identification of novel drug-drug interactions. *PLoS Comput Biol*, 9(12):e1003374, 2013.
- [95] Núria Rovira-Asenjo, Tània Gumí, Marta Sales-Pardo, and Roger Guimera. Predicting future conflict between team-members with parameter-free models of social networks. *Scientific reports*, 3(1):1–6, 2013.
- [96] Antonia Godoy-Lorite, Roger Guimerà, Cristopher Moore, and Marta Sales-Pardo. Accurate and scalable social recommendation using mixed-membership stochastic block models. *Proceedings of the National Academy of Sciences*, 113(50):14207–14212, 2016.
- [97] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.
- [98] Gyan Bhanot. The metropolis algorithm. *Reports on Progress in Physics*, 51(3):429, 1988.
- [99] Hisashi Kashima and Naoki Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *Sixth International Conference on Data Mining (ICDM’06)*, pages 340–349. IEEE, 2006.
- [100] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 322–331. IEEE, 2007.
- [101] Zan Huang. Link prediction based on graph topology: The predictive value of generalized clustering coefficient. *Available at SSRN 1634014*, 2010.
- [102] Mustafa Bilgic, Galileo Mark Namata, and Lise Getoor. Combining collective classification and link prediction. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, pages 381–386. IEEE, 2007.
- [103] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. *arXiv preprint arXiv:1301.0604*, 2012.
- [104] Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *Advances in neural information processing systems*, pages 659–666, 2004.
- [105] Ben Taskar, Pieter Abbeel, Ming-Fai Wong, and Daphne Koller. Relational markov networks. *Introduction to statistical relational learning*, pages 175–200, 2007.
- [106] David Heckerman, Chris Meek, and Daphne Koller. Probabilistic entity-relationship models, prms, and plate models. *Introduction to statistical relational learning*, pages 201–238, 2007.
- [107] David Heckerman, Christopher Meek, and Daphne Koller. Probabilistic models for relational data. Technical report, Technical Report MSR-TR-2004-30, Microsoft Research, 2004.
- [108] Jennifer Neville and David Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8(Mar):653–692, 2007.

- [109] David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. Dependency networks for collaborative filtering and data visualization. *arXiv preprint arXiv:1301.3862*, 2013.
- [110] Zhao Xu, Volker Tresp, Kai Yu, Shipeng Yu, and Hans-Peter Kriegel. Dirichlet enhanced relational learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 1004–1011, 2005.
- [111] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *SDM06: workshop on link analysis, counter-terrorism and security*, 2006.
- [112] Liang Duan, Shuai Ma, Charu Aggarwal, Tiejun Ma, and Jinpeng Huai. An ensemble approach to link prediction. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2402–2416, 2017.
- [113] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.
- [114] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.
- [115] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.
- [116] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, 2020.
- [117] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.
- [118] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48. ACM, 2013.
- [119] Shaosheng Cao, Wei Lu, and Qionghai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900. ACM, 2015.
- [120] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114. ACM, 2016.
- [121] Shaosheng Cao, Wei Lu, and Qionghai Xu. Deep neural networks for learning graph representations. In *AAAI*, pages 1145–1152, 2016.
- [122] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.
- [123] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.
- [124] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [125] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.
- [126] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.
- [127] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*, 2018.
- [128] Yun Xiong, Yao Zhang, Hanjie Fu, Wei Wang, Yangyong Zhu, and S Yu Philip. Dyngraphgan: Dynamic graph embedding via generative adversarial networks. In *International Conference on Database Systems for Advanced Applications*, pages 536–552. Springer, 2019.
- [129] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B Schardl, and Charles E Leiserson. Evolvegc: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, pages 5363–5370, 2020.
- [130] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, pages 5694–5703, 2018.
- [131] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. *arXiv preprint arXiv:1711.08267*, 2017.

- [132] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.
- [133] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *arXiv preprint arXiv:1711.08752*, 2017.
- [134] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.
- [135] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.
- [136] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [137] Will Hamilton, Zhitaoying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.
- [138] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.
- [139] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.
- [140] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.
- [141] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.
- [142] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.
- [143] Shonosuke Harada, Hirotaka Akita, Masashi Tsubaki, Yukino Baba, Ichigaku Takigawa, Yoshihiro Yamanishi, and Hisashi Kashima. Dual convolutional neural network for graph of graphs link prediction. *arXiv preprint arXiv:1810.02080*, 2018.
- [144] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.
- [145] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.
- [146] Victoria Zayats and Mari Ostendorf. Conversation modeling on reddit using a graph-structured lstm. *Transactions of the Association for Computational Linguistics*, 6:121–132, 2018.
- [147] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.
- [148] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804*, 2018.
- [149] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [150] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*, 2018.
- [151] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, pages 2022–2032, 2019.
- [152] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 950–958, 2019.
- [153] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.
- [154] Duncan J Watts and Steven H Strogatz. Collective dynamics of ‘small-world’ networks. *nature*, 393(6684):440–442, 1998.

- [155] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.
- [156] Christian Von Mering, Roland Krause, Berend Snel, Michael Cornell, Stephen G Oliver, Stanley Fields, and Peer Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, 417(6887):399–403, 2002.
- [157] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.
- [158] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.
- [159] Nikola T Markov, MM Ercsey-Ravasz, AR Ribeiro Gomes, Camille Lamy, Loic Magrou, Julien Vezoli, P Misery, A Falchier, R Quilodran, MA Gariel, et al. A weighted and directed interareal connectivity matrix for macaque cerebral cortex. *Cerebral cortex*, 24(1):17–36, 2014.
- [160] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, June 2014.
- [161] Marinka Zitnik, Sagar Maheshwari Rok Sosič, and Jure Leskovec. BioSNAP Datasets: Stanford biomedical network dataset collection. <http://snap.stanford.edu/biodata>, August 2018.
- [162] Jérôme Kunegis. Konect: The koblenz network collection. In *Proceedings of the 22Nd International Conference on World Wide Web, WWW '13 Companion*, pages 1343–1350, New York, NY, USA, 2013. ACM.
- [163] Vladimir Batagelj and Andrej Mrvar. Pajek datasets. <http://vlado.fmf.uni-lj.si/pub/networks/data/>, 2006.
- [164] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [165] Usi Alon lab. Collection of complex networks. <http://www.weizmann.ac.il>, 2006.
- [166] Peter Mucha and Mason Porter. Netwiki shared data. <http://netwiki.amath.unc.edu>, 2013.
- [167] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, August 2009.
- [168] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, pages 990–998, 2008.
- [169] grouplens. Movielens rating dataset. <https://grouplens.org/datasets/movielens/>.
- [170] R. Zafarani and H. Liu. Social computing data repository at ASU, 2009.
- [171] Nexus network repository. <https://igraph.org/r/doc/nexus.html>, 2015.
- [172] Sociopattern research collaboration. <http://www.sociopatterns.org/datasets/>.
- [173] Mark Newman. Mark newman network datasets collection. <http://www-personal.umich.edu/~mejn/netdata>, 2013.