A comparative study of similarity-based and GNN-based link prediction approaches

Md Kamrul Islam, Sabeur Aridhi, and Malika Smail-Tabbone

Universite de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France
{kamrul.islam, sabeur.aridhi, malika.smail}@loria.fr

Abstract. The task of inferring the missing links in a graph based on its current structure is referred to as link prediction. Link prediction methods that are based on pairwise node similarity are well-established approaches in the literature. They show good prediction performance in many real-world graphs though they are heuristics and lack of universal applicability. On the other hand, the success of neural networks for classification tasks in various domains leads researchers to study them in graphs. When a neural network can operate directly on the graph, then it is termed as the graph neural network (GNN). GNN is able to learn hidden features from graphs which can be used for link prediction task in graphs. Link predictions based on GNNs have gained much attention of researchers due to their convincing high performance in many real-world graphs. This appraisal paper studies some similarity and GNN-based link prediction approaches in the domain of homogeneous graphs that consists of a single type of (attributed)nodes and single type of pairwise links. We evaluate the studied approaches against several benchmark graphs with different properties from various domains.

Keywords: Neural network · Homogeneous graph · Graph labelling · Node embedding.

1 Introduction

One of the most interesting and long-standing problems in the field of graph mining is link prediction that predicts the probability of a link between two unconnected nodes based on available information in the current graph such as node attributes or graph structure [1]. The prediction of missing or potential links helps us toward the deep understanding of structure, evolution and functions of real-world complex graphs [2]. Some applications of link prediction include friend recommendation in social networks ([3]), product recommendation in e-commerce [4], knowledge graph completion [5], and finding interactions between proteins [6].

A large category of link prediction methods is based on some heuristics that measure the proximity between nodes to predict whether they are likely to have a link. Though these heuristics can predict links with high accuracy in many graphs, they lack universal applicability to different kinds of graphs. For example, the common neighbor heuristic assumes that two nodes are more likely to

connect if they have many common neighbors. This assumption may be correct in social networks, but is shown to fail in protein-protein interaction (PPI) networks where two proteins sharing many common neighbors are actually less likely to interact [7]. In case of using these heuristics, it is required to manually choose different heuristics for different graphs based on prior beliefs or expensive trial and error process. On the other hand, learning-based link prediction approaches are able to learn suitable heuristics from the graph itself. The success of the neural network is well-known for machine learning task in many real-world applications like image classification [8], speech recognition [9], video processing [10], natural language processing [11]. The applications can represent the data in Euclidean space and neural network is able to extract the hidden features from the data space. However, the neural network can not be applied directly into the graph domain due to two important challenges [12]. Firstly, a graph contains unordered nodes and a variable number of neighbours for each node. Secondly, the assumption of independence of data is no longer true for graphs as each node is linked to some other nodes. The first attempt to study the neural network in the graph domain was done in [14]. Then, Graph Neural Networks (GNNs) has become a powerful tool for learning hidden features in graphs. In the last decades, researchers have developed many GNN-based methods which are used for several tasks completion such as graph classification [15], node classification [16], and link prediction [17].

In this paper, we first introduce the link prediction problem and highlight similarity-based and GNN-based methods. Then, we choose a few approaches from both link prediction categories to evaluate their performances on different types of graphs, namely simple or homogeneous graphs and node-attributed graphs. We compare their performance with respect to the prediction accuracy and computational time.

2 Link Prediction: Problem and Approaches

Consider an undirected graph at a particular time t where nodes represent entities and links represent the relationships between pair entities (or nodes). The link prediction problem is defined as discovering or inferring a set of missing links (existing but not observed) in the graph at time $t + \Delta t$. The problem can be illustrated with a simple undirected graph in Fig. 1, where circles represent nodes and lines represent links between pair of nodes. Black solid lines represent observed links and red dashed lines represent missing links in the current graph. Fig. 1a shows the snapshot of the graph at time t, where two missing links exist between node pairs (x, y) and (g, i). The link prediction problem aiming to predict the appearance of these two missing links as observed links in the graph in near future $t + \Delta t$, as illustrated in Fig. 1b.

2.1 Similarity-based Link Prediction

The similarity-based approach is the most commonly used approach for link prediction which is developed based on the assumption that two nodes in a

(a) Graph at time t (b) Graph at time $t + \Delta t$

Fig. 1: Illustration of link prediction problem

graph interact if they are similar. The definition of similarity is a crucial and non-trivial task that varies from domain to domain even from the graph to graph in the same domain [18]. As a result, numerous similarity-based approaches have been included in the literature to predict links in small to large graphs. Some similarity-based approaches use the local neighbourhood information to compute similarity score are known as local similarity-based approach. Another category of similarity-based approach is global approaches those use the global topological information of graph. The computational complexity of global approaches makes them unfeasible to be applied on large graphs as they use the global structural information such as adjacency matrix [18]. For this reason, we are considering only the local similarity-based approaches in the current study. We have studied 13 popular similarity-based approaches for link prediction. Table 1 summarizes the approaches with the basic principle and similarity function.

These approaches except CCLP use node degree, common neighborhood or links among common neighborhood information to compute similarity scores. CCLP uses the clustering coefficient (CC) of each common neighbour to compute the role of its to the similarity score. The clustering coefficient is defined as the ratio of the number of triangles and the expected number of triangles passing through a node. If t_z is the number of triangles passing through node z and Γz is the neighbourhood of z then the clustering coefficient (CC_z) of node z is defined as

$$CC_z = \frac{2 \times t_z}{|\Gamma z| \left(|\Gamma z| - 1\right)} \tag{1}$$

Overall, these local similarity-based approaches except PA work well when the graphs have a high number of common neighbours between a pair of nodes. However, the SA, HDI and LLHN suffer from outlier when one of the two nodes has no neighbour. In addition, some of the approaches like JA, SO, HPI suffer from the outlier when both of the nodes have no neighbour.

2.2 Graph Neural Network(GNN)-based Link Prediction

Graph neural network (GNN) is an extension of the neural network to be applied to graph data. A GNN computes the node representation based on the available node information. It aggregates the information from its neighbours to find its final representation and the representation is fed into a multi-layer neural network for several downstream tasks like node classification, link prediction and graph

Table 1: Summary of studied similarity-based approaches. The similarity function is defined to predict a link between two nodes x and y. Γx and Γx denote the neighbour sets of nodes x and y respectively. $r_{x,y}$ denotes the link between two nodes x, y.

Approach	**Principle**	**Similarity-function**
Adamic-Adar (AA) [3]	Variation of CN where each common neighbour is logarithmically penalized by its degree	$S^{AA}(x,y) = \sum_{z \in \Gamma x \cap \Gamma y} \frac{1}{log\lvert\Gamma z\rvert}$
Common Neighbours (CN) [19]	Two nodes are more likely to be linked share more neighbours	$S^{CN}(x,y) = \lvert\Gamma x \cap \Gamma y\rvert$
Resource Allocation (RA) [20]	Based on the resource allocation process to further penalize the high degree common neighbours by more amount	$S^{RA}(x,y) = \sum_{z \in \Gamma x \cap \Gamma y} \frac{1}{\lvert\Gamma z\rvert}$
Preferential Attachment (PA) [21]	Based on the rich-get-richer concept where the link probability between two high degree nodes is higher than two low degree nodes	$S^{PA}(x,y) = \lvert\Gamma x\rvert \times \lvert\Gamma y\rvert$
Jaccard Index(JA) [22]	Normalization of CN where the score is penalized for each non-common neighbour	$S^{JA}(x,y) = \frac{\lvert\Gamma x \cap \Gamma y\rvert}{\lvert\Gamma x \cup \Gamma y\rvert}$
Salton Index(SA) [23]	Motivated by cosine similarity that defines link probability based on cosine angle between adjacency vectors for nodes pair	$S^{SA}(x,y) = \frac{\lvert\Gamma x \cap \Gamma y\rvert}{\sqrt{\lvert\Gamma x\rvert \times \lvert\Gamma y\rvert}}$
Sørensen Index(SO) [24]	Describing the overall proportion of common neighbours from a local perspective.	$S^{SO}(x,y) = \frac{2 \times \lvert\Gamma x \cap \Gamma y\rvert}{\lvert\Gamma x\rvert + \lvert\Gamma y\rvert}$
Hub Promoted Index (HPI) [25]	Promoting link formation between high-degree nodes and hubs	$S^{HPI}(x,y) = \frac{\lvert\Gamma x \cap \Gamma y\rvert}{max(\lvert\Gamma x\rvert, \lvert\Gamma y\rvert)}$
Hub Depressed Index (HDI) [25]	Promoting link formation between low-degree nodes and hubs.	$S^{HDI}(x,y) = \frac{\lvert\Gamma x \cap \Gamma y\rvert}{min(\lvert\Gamma x\rvert, \lvert\Gamma y\rvert)}$
Local Leicht-Holme-Newman (LLHN) [26]	Utilizing both of real and expected amount of common neighbours between a pair of nodes to define their similarity.	$S^{LLHN}(x,y) = \frac{\lvert\Gamma x \cap \Gamma y\rvert}{\lvert\Gamma x\rvert \times \lvert\Gamma y\rvert}$
Individual Attraction (IA) [27]	Maximizing the likelihood of link formation for highly interlinked nodes pair.	$S^{IA}(x,y) = \sum_{z \in \Gamma x \cap \Gamma y} \frac{\lvert r_{z,\Gamma x \cap \Gamma y}\rvert + 2}{\lvert\Gamma z\rvert}$
Cannistrai–Alanis–Ravai (CAR) [28]	Utilization of level-2 links along with common neighbourhood information in computing the pairwise similarity score	$S^{CAR}(x,y) = \sum_{z \in \Gamma x \cap \Gamma y} 1 + \frac{\lvert\Gamma x \cap \Gamma y \cap \Gamma z\rvert}{2}$
Clustering Coefficient-based Link Prediction (CCLP) [29]	Quantification of the contribution of each common neighbour by utilizing the local clustering coefficient of the node.	$S^{CCLP}(x,y) = \sum_{z \in \Gamma x \cap \Gamma y} CC_z$

classification. Based on the architecture, GNNs are broadly categorized into five categories: recurrent graph neural network (RecGNN), convolution graph neural network (ConvGNN), graph auto-encoder (GAE), and spatial-temporal graph neural network (STGNN) [12]. RecGNNs are the pioneers of GNNs those work based on the assumption that the nodes constantly exchange the information with the neighbours until a stable state is reached. Motivated by the convolution operation of the neural network in the image domain, ConvGNNs compute the embedding of a node by aggregating its own information and neighbours information. GAEs are the unsupervised version of GNN those encode the nodes into a latent vector space and reconstruct the graph to learn the embedding. STGNNs are used to learn the hidden features in a spatio-temporal graph based on the spatial and temporal dependency with time. Recently, researchers have studied the attention mechanism in RecGNN and ConvGNN to improve the prediction performance by allowing them to focus on the most relevant parts of the graph [13]. ConvGNNs has become popular in recent years due to its efficient graph convolution operation [12, 30]. In this paper, we focus on the link prediction approaches based on ConvGNN. A ConvGNN starts with defining the neighbourhood Γv_i of each node v_i in the graph $G(V, E)$ which is a crucial task as it can affect the accuracy and computational time. Some popular neighbourhood definitions include immediate neighbours, multi-hop neighbours [17, 31], sampling-based neighbours [32, 33]. The feature vector, x_i of each node, v_i is then computed based on its attribute and structural information. The feature vectors of nodes are fed into a stack of layers to learn the hidden features of the graph. A simple ConvGNN update the node representation in each layer in the following three basic steps [30, 34]

1. **Computation of neural messages:** The neural messages of each link for next layer is computed based on the current representations of both end nodes of the link. If h_i^l and h_j^l are current representations of a nodes pair (v_i, v_j), the message of the link is defined as

$$m_{ij}^{l+1} = MSG(h_i^l, h_j^l, r_{ij}) \tag{2}$$

   Here, l represents the current layer, $r_{ij} \in E$ is the relation between the nodes pair and MSG is the message computation function for the links. Many GNN models use the link types [35] or link weight [36] for encoding r_{ij}. The initial representations of nodes v_i and v_j are x_i and x_j respectively (i.e. $h_i^1 = x_i$ and $h_j^1 = x_j$).
2. **Aggregating the neighbour information:** The next operation of the layer is to aggregate the neighbour messages m_{ij}^l for node v_i. An aggregation function is defined as

$$M_i^{l+1} = AGGR(m_{ij}^{l+1} | v_j \in \Gamma v_i) \tag{3}$$

   Here, Γv_i is the set of neighbours of node v_i and $AGGR$ is the aggregation function. Some popular aggregation functions exist in the literature such as mean/max pooling [37], sort pooling [38], permutation invariant [39].

3. **Updating the node representation:** In this step, the representation or embedding of node v_i in the next layer is updated based on the current embedding, h_i^l and the aggregated message, M_i^{l+1}

$$h_i^{l+1} = UPDATE(h_i^l, M_i^{l+1}) \tag{4}$$

Here, the $UPDATE$ function is a non-linear function like sigmoid, recified linear unit(ReLU), hyperbolic tangent(TanH). The output embedding h_i^{l+1} is the input for next layer.

Each layer in the model follows these three steps and generates nodes embedding. The embedding from the last layer is fed into a standard classifier such as multilayer perception (MLP) with a softmax layer for downstream tasks. The parameters of the classifier are optimized using optimizer like Adam, stochastic gradient descent(SGD) along with loss functions such as cross-entropy, mean absolute error(MAE), mean squared error(MSE) and backpropagation.

There exist many link prediction approaches based on ConvGNN in the literature. Most of them are applicable to homogeneous graphs and few of them are applicable to heterogeneous graphs which consist of multiple types of nodes and links, node and link attributes and multiple links between pairs of nodes. We study two recent GNN-based link prediction approaches which are applicable to homogeneous graphs only as our study is confined to those graphs. The first one is WLNM (Weisfeiler-Lehman Neural Machine) that utilizes only the structural information of nodes for the link prediction task. SEAL (Sub-graphs, Embeddings and Attributes) is the second one that uses the structural, latent and attribute information of node for the same task. The approaches are briefly described below.

Weisfeiler-Lehman Neural Machine (WLNM) Based on the well-known Weisfeiler-Lehman canonical labelling algorithm [40], Zhang &Chen developed a link prediction approach for graph called Weisfeiler-Lehman Neural Machine (WLNM) [32]. WLNM learns the structural features from the graph and uses in prediction task. WLNM is a three steps link prediction approach that starts with extracting sub-graphs, labelling and encoding the nodes and ends with training and evaluating the neural network. Fig. 2 illustrates the training process of WLNM with one existent link (A,B) and one non-existent link(c,d). The three steps of WLNM link prediction approach are described as following.

1. **Sub-graph extraction:** WLNM starts with extracting the k-vertex neighbouring sub-graph of a link called enclosing sub-graph. k is the user-defined parameter that defines the size of sub-graph. For a given link, 1-hop neighbours are added in the sub-graph, then 2-hop neighbours and so on until the number of neighbours is greater or equal to k. If there are k' nodes in sub-graph such that $k' > k$ then $k' - k$ nodes with higher hop number are removed sub-graph.
2. **Node labelling and encoding:** Weisfeiler-Lehman (WL) is a popular graph labelling algorithm that uses the concept of signature string for each

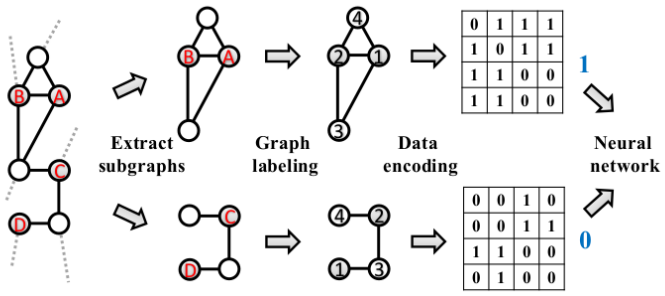

Fig. 2: Illustration of WLNM approach [32]

node to compute node labels [40]. Instead of using classical WL algorithm, WLNM develops a hashing based color refinement process for faster node labelling. If two nodes have still the same label, WLNM uses the naughty node labelling algorithm to break the tie [41]. The nodes are sorted according to the node label in increasing order and an upper triangular adjacency matrix is computed.

3. **Neural network training and evaluation:** WLNM uses a fully connected multi-layer perception (MLP) neural network to learn structural features from the sub-graph. The output layer of the MLP is a softmax layer that classifies the link into two classes (existent and non-existent). The upper triangular adjacency matrix of the sub-graph is vertically fed into the MLP to train and evaluate WLNM approach. The neural network is trained for both of existent and non-existent links.

WLNM is a simple GNN-based link prediction approach which is able to learn the link prediction heuristics from a graph. In contrast to similarity-based heuristics, WLNM has universal applicability properties. However, WLNM truncates some neighbours to limit the number of nodes in the sub-graph to a user-defined size. The truncated neighbours may be informative for the prediction task.

Learning from Sub-graphs, Embeddings and Attributes (SEAL) Zhang et al [17] developed a ConvGNN-based link prediction approach namely SEAL to learn from latent and explicit features of nodes along with the structural information of graph. Unlike WLNM, SEAL is able to handle neighbours of variable size. SEAL replaces the fully-connected neural network in WLNM with a graph neural network to learn the graph features efficiently. The overall architecture of the approach is shown in Fig. 3. Like WLNM, SEAL also consists of three major steps which are described as follows:

1. **Sub-graph extraction and node labelling:** Likewise WLNM, SEAL approach uses the concept of local sub-graph instead of the whole graph for a link in prediction task. SEAL defines the sub-graph as the h-hop neighbours of a link which is built by the union operation on the h-hop neighbours

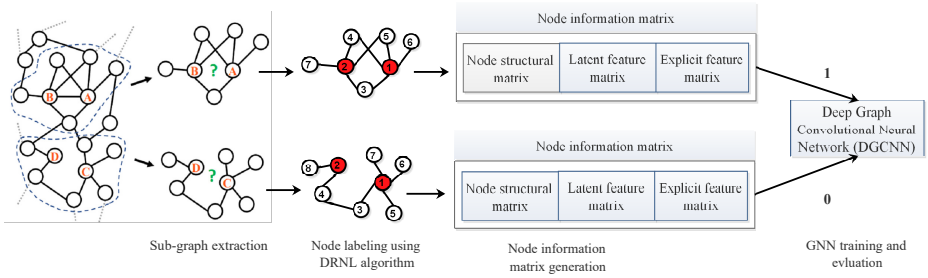

Fig. 3: Architecture of SEAL approach

of nodes of the link. For example, a 1-hop enclosing sub-graph contains all first-order or immediate neighbours, a 2-hop enclosing sub-graph contains all first-order and second-order neighbours. In Fig. 3, the sub-graph for link (A,B) consists of 7 nodes(5 neighbours and 2 end nodes) and the sub-graph for link (C,D) consists of 8 nodes(6 neighbours and 2 end nodes). The approach shows that setting a small h can still provide good prediction performance. Then a unique label is assigned to each node of the sub-graph to indicate its importance in the prediction task. SEAL designs a new node labelling algorithm namely DRNL (double-radius node labelling) based on the topological distances of the node from both ends of the link in the sub-graph.

2. **Node information matrix construction:** The information matrix of a node in SEAL is defined based on its structural label(structural feature), embedding(latent feature) and attribute(explicit feature). One hot encoding technique is applied to the labelled sub-graph to compute the structural vector of nodes. The structural feature vector of the node is then concatenated with the latent feature vector of the node. The latent feature is the low-dimensional latent representation/embedding of a node which is obtained by factorizing the adjacency matrix from the graph. SEAL uses the Node2Vec [42] algorithms to learn the latent feature vector for each node in sub-graph. The last part of the information vector of the node is an explicit feature vector which is computed based on the continuous or discrete attributes of the node. One hot coding technique is used to find the explicit feature vector of each node.
3. **Neural network training and evaluation:** The learned node information matrix of the sub-graph is feed into a GNN called DGCNN (Deep Graph Convolutional Neural Network) [38] to perform the link prediction task. DGCNN consists of propagation-based convolution layer and aggregation layer to aggregate the neighbour's information vector. DGCNN uses a sort-pooling layer to unify the size of the representation of the sub-graph.

SEAL utilizes the available information in the graph to improve the prediction performance. However, SEAL is limited to be applied on homogeneous graphs though many real work graphs are heterogeneous graphs. Moreover, the use of latent feature affects the computational time of SEAL.

3 Experimental Design

3.1 Datasets Characteristics

We perform the comparative study of the above discussed similarity and GNN based link prediction approaches in graphs from different domains. To evaluate and describe the performance of the link prediction approaches, we choose ten benchmark graphs from different areas: Ecoli [43], FB15K [44], NS [45], PB [46],Power [47], Router [48], USAir [49], WN18 [50], YAGO3-10 [51], and Yeast [52]. Ecoli and Yeast are two biological graphs those represent the biological relations between operons in Escherichia Coli bacteria and protein-protein interaction in yeast. PB (Political Blog) graph represents the network among political blog pages in US where the blog pages are identified as nodes and hyperlinks between the blog pages are identified as links of the graph. We consider the original directed links as undirected links. Net Science (NS) graph represents a collaboration network of researchers who publish papers on network science. Power is an electrical grid network of western US representing the network describing high voltage transmission among generators, transformers and substations. The Router graph represents the router-level internet where each router has an identifier and undirected links with other routers. The USAir graph represents the network of the US air transportation system that consists of attributed nodes (airports) and links between two airports. FB1K, WN18 and YAGO3-10 are simplified knowledge graphs. The original FB15K is a Freebase Knowledge Graph which was extracted from Wikidata and DBPedia. This knowledge graph contains 540188 triples where each triple consists of identifiers of freebase entity with a relationship name between them. WN18 is another knowledge graph that is a large lexical graph of English. The last knowledge graph is YAGO3-10 that was prepared at the Max Planck Institute for Computer Science in Saarbrucken in 2015. These knowledge graphs consist of subject-relationship type-object triples. However, as most studied approaches are applicable to homogeneous graphs only, we simplify these knowledge graphs by overlooking the types of relationships and reducing multiple links to single links between nodes/entities. All of the graphs are considered as undirected graphs. In this study, we are considering them as large graphs instead of knowledge graphs.

We use the Gephi tool [53] to extract the topological statistics of the graphs. The characteristics of the graph datasets are summarized in Table 2. Based on the number of nodes, these graphs are categorized into small/medium graphs with less or equal 10000 nodes and large graphs with more than 10000 nodes.

3.2 Construction of Train and Test sets

We follow a random sampling validation protocol to evaluate the performance of the studied approaches [32, 54]. The train and test datasets are prepared from a graph G(V, E), where V is the set of vertices and E is the set of existent links. Two types of both training and test datasets are prepared from the graph. The first training dataset is positive training dataset that contains randomly

Table 2: Topological statistics of graph datasets: number of nodes(#Node), links(#Link), average node degree (NDeg), total triangle(#Triangle), clustering coefficient (C.Coef), average path length (APL), network diameter (Diam) and type of graph.

Graphs	#Node	#Link	NDeg	#Triangle	C.Coef	APL	Diam	Graph type
Ecoli	1805	42325	46.898	459809	0.350	2.714	10	Homogeneous
FB15K	14949	260183	44.222	565104	0.218	2.716	8	Homogeneous
NS	1461	2742	3.754	3764	0.878	5.823	17	Homogeneous
PB	1222	14407	23.579	49549	0.239	2.787	8	Homogeneous
Power	4941	6594	2.669	651	0.107	18.989	46	Homogeneous
Router	5022	6258	2.492	803	0.033	6.449	15	Homogeneous
USAir	332	2126	12.807	12181	0.749	2.738	6	Homogeneous, node-attributed
WN18	40943	75769	3.709	5107	0.077	7.426	18	Homogeneous
YAGO3-10	113273	758225	18.046	225094	0.114	22.999	14	Homogeneous
Yeast	2375	11693	9.847	60689	0.388	5.096	15	Homogeneous

selected 90% observed links and an equal number of non-existent links form the negative training dataset. The remaining 10% existent links form the positive test dataset and an equal number of non-existent links form the negative test dataset. At the same time, the graph connectivity of the training set and the test set is guaranteed. We prepare five train and five test datasets for evaluating the performance of the approaches.

For evaluating the performance of similarity-based approaches, the graph is built from the positive training dataset whereas, for graph neural network-based approaches, the graph is built from the original graph that contains both of positive train and test datasets. However, a link is temporarily removed from the graph to train it to the GNN-based approaches or to predict its existence. The approaches are evaluated on positive and negative test datasets. For WLNM, we choose the neighbour size to 10 and for SEAL we choose the hop to 1 for all graphs. The similarity scores of similarity-based approaches for test links are computed based on the training graphs which contain only train links. The performance of link prediction approach is quantified by defining two standard evaluation metrics, precision and AUC (Area Under the Curve). All of the approaches are run on a Dell Latitude 5400 machine with 32GB primary memory and core i7 (CPU 1.90GHz) processor.

3.3 Computation procedures for Precision and AUC

Precision describes the fraction of missing links that are accurately predicted as existent link [55–57]. To compute the precision, all of the predicted links from a test set are ranked in decreasing order of their scores. If L_r is the number of existing links (in the positive test set) among the L-top ranked predicted links

then the precision is defined as

$$Precision = \frac{L_r}{L} \tag{5}$$

The precision is a measure of result relevance. The higher the precision indicates the higher accuracy of the prediction approach. An ideal prediction approach has a precision of 1.0 that means all the missing links are accurately predicted. We set L to the number of existent links in the test set.

On the other hand, the metric AUC is measured to demonstrate the ability of an approach in distinguishing between an existent and a non-existent link. It is defined as the probability that a randomly chosen missing link has a higher similarity score than a randomly chosen non-existent link [56]. Suppose, n existent and n non-existent links are chosen from positive and negative test sets. If n_1 is the number of existent links having a higher score than non-existent links and n_2 is the number of existent links having equal score as non-existent links then AUC is defined as

$$AUC = \frac{n_1 + 0.5n_2}{n} \tag{6}$$

An AUC of more than 0.5 indicates that the prediction index has a better effect than choosing links randomly and vice versa. Generally, the degree to which AUC exceeds 0.5 indicates how much good the prediction approach. We consider half of the total links in the positive test set and negative test set to compute AUC.

4 Analysis of the Results

4.1 Comparison of Prediction Accuracy with Precision and AUC

The prediction approaches are evaluated in each of the five sets (train and test set) of each graph and performance metrics (precision, AUC) are recorded. The maximum and minimum similarity scores are computed from the top-L for each test set of each graph. Table 3 shows the mean maximum(Max Score) and minimum similarity (Min Score) scores for each similarity-based in each graph. We measure the precision in two different ways based on the top-L test links. Firstly, we use Equation 5 as it is where L_r is the number of positive links in top-L test links. However, the minimum similarity scores for many similarity-based approaches are very low (close to 0) that creates difficulty to make a separation between some positive and negative test links. To overcome this problem, we define a threshold when defining L_r. However, defining threshold to similarity-based approaches is again a non-trivial task as the maximum and minimum scores vary for different graphs and even for different test sets. To overcome this problem, we define a threshold as the average of the maximum and minimum score in top-L links. We compute the number of positive test links in top-L links (as L_r) as those having similarity scores above the threshold. We compute the threshold-based precision only for similarity-based approaches as GNN-based approaches do learn the threshold. The corrected precision is shown in parentheses in Table 3. Each value of the table is the mean over the five test sets. The

evaluation metrics precision and AUC for the studied approaches in the seven small to medium-size graphs are tabulated in Table 3.

Table 3 shows that, overall, the similarity-based approaches give high precision (without defining threshold) and AUC values in well-connected (high clustering coefficient, high node degree) graphs while GNN-based approaches show good precision and AUC in all graphs. In the Ecoli graph, CCLP shows the highest precision (0.96) while the lowest precision(0.78) is recorded for the PA approach. The precisions of other similarity-based approaches are close to the highest precision score. The highest clustering coefficient contributes to the success of CCLP in terms of precision in Ecoli. However, the precision of similarity-based approaches drops drastically when computing precision based on the threshold as many positive links with very low similarity scores (even 0) comparing to the threshold. The precision of WLNM and SEAL approaches are lower than the similarity-based approaches and they are 0.867 and 0.807 respectively. The highest and lowest AUC values in Ecoli are found for SEAL and PA approaches respectively. The AUC value of another GNN-based approach WLNM is also very high and close to the highest AUC value. The high values of these two GNN-based approaches state that they are highly efficient in distinguishing between existent and non-existent links in Ecoli graph. Similar performance is found for other well-connected graphs (NS, PB, USAir and Yeast). In NS graph, SEAL performs with the best precision (0.96) and AUC (0.99) score and PA is the worst approach which shows the lowest precision and AUC values of 0.69 and 0.66 respectively. The precision scores of other approaches lie between 0.8 to 0.9 while the AUC values are between 0.9 to 0.95. A remarkable precision(highest) is found for HPI in NS graph while the precision scores of some similarity-based approaches like AA, CN, PA, RA are still very low when applying the threshold method. Overall, the AUC values of GNN-based approaches are higher than the similarity-based approaches in NS graph. In PB graph, the highest precision score is recorded in similarity-based approaches RA, CAR and CCLP whereas the highest AUC value is found for the GNN-based approach SEAL. LLHN performs worst in PB concerning both metrics. The precision of other approaches near or above 0.8. The high average node degree plays a role in most of the similarity-based approaches in performing better than the GNN-based approaches in terms of precision scores in PB graph. However, the precision of similarity-based approaches drops to below 0.2 when applying the threshold in computing precision. Similarity-based approaches shows very low precisions and low AUCs in two sparse graphs, Power and Router whereas the GNN-based approaches are still able to provide high precisions and AUCs in both of the graphs. In both of USAir and Yeast graphs, SEAL shows the best results with precision of 0.94 and 0.89 and AUC of 0.96 and 0.98 respectively while the lowest precision and AUC values are recorded for WLNM and LLHN respectively. The use of node attributes for SEAL in USAir during prediction task influences in the improvement of the performance metrics. Overall, SEAL shows the highest AUC values in all graphs. The use of latent feature along with structural feature is the vital reason behind this success. Table 3 shows that GNN-based approaches pro-

Table 3: AUC and Precision values with Max Scores and Min Scores in small/medium graphs. Precision in () is computed based on threshold in top-L links. Graph-wise highest/lowest metrics are indicated in bold fonts while approach-wise highest/lowest metrics are shown in italic.

App.	Metrics	Ecoli	NS	PB	Power	Router	USAir	Yeast
AA	Precision	0.90(0.06)	0.87(0.15)	**0.86**(0.01)	0.17(0.02)	***0.07***(0.01)	*0.92*(0.16)	0.83(0.06)
	Max scor	32.84	5.83	33.41	3.04	5.60	16.69	23.71
	Min scor	2.86	1.14	0.58	0.00	0.00	2.70	0.00
	AUC	0.93	*0.94*	0.92	0.58	*0.54*	*0.94*	0.91
CN	Precision	0.91(0.07)	0.87(0.22)	**0.86**(0.02)	0.17(0.04)	***0.07***(.004)	*0.92*(0.23)	0.83(0.06)
	Max scor	153	11.0	119	29.0	15.0	51.0	90.33
	Min scor	12.0	1.40	3.00	0.00	0.00	9.67	0.00
	AUC	0.93	0.93	0.91	0.58	*0.54*	*0.95*	0.91
PA	Precision	**0.78**(0.05)	**0.69**(0.02)	0.83(0.01)	0.49(0.02)	*0.41*(0.01)	***0.85***(0.13)	**0.79**(0.06)
	Max scor	65679	362.0	61052	53.0	2397	8298.7	10642
	Min scor	3532	12.0	855.7	4.0	1.0	739.3	95.0
	AUC	**0.80**	**0.66**	*0.90*	**0.46**	***0.43***	*0.90*	**0.86**
RA	Precision	0.91(0.03)	0.87(0.15)	**0.86**(0.01)	0.17(0.03)	***0.07***(0.01)	*0.92*(0.10)	0.83(0.07)
	Max scor	1.70	1.80	4.19	0.84	1.32	2.83	2.37
	Min scor	0.19	0.40	0.03	0.00	0.00	0.32	0.00
	AUC	*0.94*	*0.94*	0.92	0.58	0.54	*0.94*	0.91
JA	Precision	*0.90*(0.11)	0.87(0.42)	0.79(0.07)	0.17(0.07)	***0.07***(0.01)	0.88(0.18)	0.83(0.34)
	Max scor	0.49	0.60	0.37	0.60	0.39	0.45	0.50
	Min scor	0.10	0.09	0.04	0.00	0.00	0.17	0.00
	AUC	*0.94*	0.92	0.87	0.58	*0.53*	0.92	0.91
SA	Precision	*0.91*(0.10)	0.87(0.67)	0.80(0.06)	0.17(0.07)	***0.07***(0.01)	0.90(0.15)	0.83(0.40)
	Max scor	0.98	1.00	0.75	0.94	0.90	0.91	1.00
	Min scor	0.22	0.51	0.11	0.11	0.00	0.41	0.00
	AUC	*0.94*	*0.94*	0.87	0.57	*0.54*	0.91	0.90
SO	Precision	*0.90*(0.11)	0.87(0.64)	0.79(0.07)	0.17(0.06)	***0.07***(0.01)	0.88(0.18)	0.83(0.34)
	Max scor	0.98	1.00	0.74	0.93	0.90	0.91	1.00
	Min scor	0.19	0.46	0.07	0.00	0.00	0.34	0.00
	AUC	*0.94*	*0.94*	0.87	0.57	*0.54*	0.90	0.91
HPI	Precision	0.90(0.20)	0.87(**0.96**)	0.80(0.15)	0.17(0.13)	***0.07***(0.02)	*0.91*(0.45)	0.83(0.70)
	Max scor	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Min scor	0.33	0.83	0.21	0.00	0.00	0.77	0.00
	AUC	*0.94*	*0.94*	0.85	0.58	*0.54*	0.91	0.90
HDI	Precision	*0.90*(0.08)	0.87(0.64)	0.79(0.05)	0.17(0.03)	***0.07***(0.01)	0.88(0.18)	0.83(0.24)
	Max scor	0.97	1.00	0.68	0.89	0.89	0.85	1.00
	Min scor	0.14	0.33	0.05	0.00	0.00	0.24	0.00
	AUC	*0.94*	*0.94*	0.86	0.58	*0.53*	0.90	0.91
LLHN	Precision	*0.89*(.001)	0.87(0.13)	**0.74**(.001)	0.17(0.03)	***0.07***(.003)	0.87(0.03)	0.83(0.01)
	Max scor	0.32	1.00	0.42	2.06	0.83	0.58	0.67
	Min scor	0.00	0.10	0.00	0.00	0.00	0.01	0.00
	AUC	0.91	*0.93*	**0.76**	0.58	*0.53*	**0.77**	0.90
IA	Precision	0.90(0.07)	0.87(0.27)	0.85(0.03)	0.17(0.12)	***0.07***(0.01)	*0.92*(0.28)	0.83(0.07)
	Max scor	149.4	10.7	91.2	4.3	8.1	46.6	80.6
	Min scor	12.5	2.6	3.5	0.0	0.0	11.2	0.0
	AUC	*0.93*	*0.93*	0.91	0.58	*0.54*	*0.93*	0.91
CAR	Precision	0.91(0.04)	0.87(0.18)	**0.86**(0.02)	0.17(0.03)	***0.07***(0.01)	*0.92*(0.24)	0.83(0.06)
	Max scor	4833	46.0	1515.2	2.3	25.2	555	1831
	Min scor	50.2	1.4	3.0	0.0	0.0	46.0	0.0
	AUC	*0.93*	*0.93*	0.91	0.59	*0.54*	0.91	0.91
CCLP	Precision	***0.96***(0.06)	0.73(0.21)	**0.86**(0.01)	**0.08**(0.01)	***0.07***(0.01)	0.91(0.18)	0.82(0.06)
	Max scor	30.6	8.0	27.0	1.2	1.1	21.1	39.2
	Min scor	1.8	0.3	0.3	0.0	0.0	2.9	0.0
	AUC	***0.95***	0.87	0.91	0.54	*0.53*	0.94	0.90
WLNM	Precision	0.87	0.84	*0.78*	**0.84**	***0.89***	0.85	0.87
	AUC	0.93	*0.95*	0.93	*0.76*	0.92	0.86	0.86
SEAL	Precision	0.81	***0.96***	0.80	*0.66*	0.80	**0.94**	**0.89**
	AUC	**0.95**	***0.99***	**0.94**	***0.77***	**0.94**	**0.96**	**0.98**

vide high-performance metrics in all graphs while similarity-based approaches perform well in some graphs.

The approaches are further evaluated in three large graphs FB15K, WN18 and YAGO3-10 and the results are presented in Table 4. We can see that some similarity-based approaches (AA, CN, PA, RA, IA, CAR) show higher metric values while others (JA, SA, SO, HDI, LLHN) show lower metric values than the GNN-based approaches in FB15K graph. The highest precision score is found for CN, IA, CAR approaches and the highest AUC value is found for SEAL. LLHN is the worst performing approach concerning both of the metrics among all approaches in FB15K graph. However, the precision drops to below 0.1 for all similarity-based approaches when applying the threshold to similarity scores with FB15K graph.

As shown in Table 2, WN18 is a sparse graph with low average node degree (3.709) and clustering coefficient (0.077). This sparsity affects the performance of similarity-based approaches as these approaches except PA highly depend on the common neighbourhood information. The precision scores of all similarity-based approaches are below 0.2 except PA that shows a comparatively good precision score of 0.63. The precision further drops when applying the threshold to similarity scores in top-L links. Compared to the similarity-based approaches, GNN-based approaches show higher precision and AUC values in WN18 graph. The highest precision and AUC values are recorded for WLNM and SEAL approaches respectively. In YAGO3-10 graph, PA performs surprisingly well with precision and AUC values of 0.83 and 0.88 respectively. However, the highest precision and AUC values are found for the SEAL approach. Overall, GNN-based approaches are more suitable across graphs from several domains with respect to precision and AUC values.

From Tables 3 and 4, the node-degree based approach PA shows higher performance comparing to other neighborhood based similarity approaches. The highest precision of PA is found in USAir (0.92) and the lowest one in Router(0.41). Similarity-based approaches based on the common neighborhood show impressive performance in the graphs with high average node degree and clustering coefficient. These approaches show very high precision of above or nearly 0.9 in two well connected Ecoli and USAir graphs. These approaches show very low precision of less than 0.2 in two large graphs, WN18 and YAGO3-10. On the other hand, the GNN-based approaches show very high precision and AUC across all of the experimental graphs including small to large graphs.

4.2 Comparison of Computational Time

The performance is further described in terms of computational time. Every approach is executed for each test set of each graph and their computational times are recorded. The computational time for similarity-based heuristic is the average time required per test link to compute the nodes similarity score. On the other hand, the computational times for GNN-based prediction approaches are the accumulated time for training the GNN and predicting the classes of links (existence or non-existence) in test sets. Table 5 shows the mean computational

Table 4: AUC and Precision values with Max Scores and Min Scores in large graphs. Similar to Table 3

App.	Metrics	FB15K	WN18	YAGO3-10
AA	Precision	*0.77*(0.0002)	***0.13***(0.0002)	0.15(0.0018)
	Max Score	418.60	57.32	24.44
	Min Score	0.12	0.00	0.00
	AUC	*0.82*	0.56	*0.48*
CN	Precision	***0.81***(0.0003)	***0.13***(0.0004)	0.15(0.0012)
	Max Score	1231.3	60.00	98.00
	Min Score	1.00	0.00	0.00
	AUC	*0.80*	0.57	*0.48*
PA	Precision	*0.79*(0.0003)	*0.63*(0.0006)	0.83(0.0006)
	Max Score	9881842.3	10636.7	2426939
	Min Score	942.67	6.33	109.00
	AUC	*0.88*	*0.64*	0.88
RA	Precision	*0.77*(0.0003)	***0.13***(0.0002)	0.15(0.0011)
	Max Score	72.06	20.67	5.16
	Min Score	0.00	0.00	0.00
	AUC	*0.84*	0.57	0.57
JA	Precision	***0.64***(0.0225)	***0.13***(0.0161)	**0.15**(0.0059)
	Max Score	0.50	0.50	0.50
	Min Score	0.01	0.00	0.00
	AUC	*0.68*	**0.56**	*0.46*
SA	Precision	*0.65*(0.0236)	***0.13***(0.0218)	0.15(0.0068)
	Max Score	1.00	1.00	1.00
	Min Score	0.02	0.00	0.00
	AUC	*0.70*	0.57	*0.47*
SO	Precision	***0.64***(0.0225)	***0.13***(0.0180)	0.15(0.0059)
	Max Score	1.00	1.00	1.00
	Min Score	0.01	0.00	0.00
	AUC	*0.69*	0.57	*0.46*
HPI	Precision	*0.69*(0.0959)	***0.13***(0.0796)	0.15(0.0476)
	Max Score	1.00	1.00	1.00
	Min Score	0.05	0.00	0.00
	AUC	*0.75*	0.56	*0.47*
HDI	Precision	***0.64***(0.0137)	***0.13***(0.0121)	0.15(0.0035)
	Max Score	1.00	1.00	1.00
	Min Score	0.01	0.00	0.00
	AUC	*0.68*	0.57	*0.46*
LLHN	Precision	***0.64***(0.0008)	***0.13***(0.0046)	0.15(0.0003)
	Max Score	0.28	1.00	1.00
	Min Score	0.00	0.00	0.00
	AUC	**0.57**	0.57	***0.45***
IA	Precision	***0.81***(0.0003)	***0.13***(0.0505)	0.15(0.0014)
	Max Score	757.1	4.58	95.23
	Min Score	2.00	0.00	0.00
	AUC	*0.80*	0.57	*0.47*
CAR	Precision	***0.81***(0.0003)	*0.13*(0.0004)	0.15(0.0008)
	Max Score	6906	60.00	1430
	Min Score	1.00	0.00	0.00
	AUC	*0.80*	0.57	*0.48*
CCLP	Precision	*0.78*(0.0015)	***0.08***(0.0006)	0.14(0.0013)
	Max Score	51.74	1.67	20.77
	Min Score	0.01	0.00	0.00
	AUC	*0.84*	*0.54*	0.57
WLNM	Precision	*0.67*	***0.84***	0.68
	AUC	*0.68*	*0.79*	0.72
SEAL	Precision	0.77	*0.61*	***0.86***
	AUC	**0.96**	***0.87***	***0.97***

time in milliseconds. From Table 5, it is seen that PA has the lowest mean

Table 5: Computational time (milliseconds). The graph-wise highest and lowest mean computational time are indicated in bold fonts and approach-wise highest and lowest mean computational time are indicated in italic.

Approach	Ecoli	FB15K	NS	PB	Power	Router	USAir	WN18	YAGO 3-10	Yeast
AA	221	495	71	106	73	74	*15*	288	*910*	107
JA	28	121	26	25	63	60	*13*	256	*647*	27
PA	**28**	**120**	**21**	**23**	61	58	***12***	251	*642*	26
RA	330	494	70	110	65	*63*	104	274	*915*	95
CN	30	120	24	24	59	56	14	249	***629***	**25**
SA	58	226	40	48	105	102	*21*	480	*1310*	48
SO	60	228	44	47	104	98	*22*	476	*1298*	49
HPI	87	236	63	70	149	142	*33*	493	*1400*	70
HDI	60	227	40	49	102	96	*19*	466	*1367*	47
LLHN	63	147	39	48	99	95	*20*	465	*1370*	47
IA	412	420	57	218	57	*55*	82	262	*938*	103
CAR	280	303	54	134	**54**	***53***	63	**157**	*643*	117
CCLP	409	654	108	280	157	*152*	257	492	*1696*	255
WLNM	612	837	170	453	257	245	*153*	541	*1440*	363
SEAL	**886**	**1221**	**398**	**940**	***340***	**419**	**524**	**868**	***2713***	**403**

computational time among the similarity-based approaches in half of the graph sets as it requires a simple multiplication operation of degrees of two end nodes in a link. The computational time for simple CN approaches are close to the PA approaches in all approaches. Similarity approaches those quantify the role of each neighbour or level-2 links such as JA, RA, IA require higher processing time. The highest computational times are found for CCLP similarity-based approach in all graphs as CCLP explores level-3 links for computing the similarity score. However, CAR requires the minimum computational time to predict links in sparse graphs (Power, Router, WN18) as these graphs have very lower clustering coefficient comparing to other graphs. The computational times of these approaches are affected by the graph properties such as average node degree, number of nodes and links, average clustering coefficient. For example, the computational time of all similarity-based approaches in NS graph is more than in USAir as NS is larger than USAir in terms of the number of nodes and link. The computational time in PB graph is more than in NS approaches as PB has more average node degree than NS though the number of nodes is higher in NS.

Compared to similarity-based approaches, the computational times of GNN-based ones are higher as they learn the heuristics from the graph during the training operation. Table 5 shows that the computational times for SEAL are greater than WLNM in all graphs as SEAL utilizes the structural, latent and explicit features of graph comparing whereas WLNM utilizes only the structural features of the graph. One noticeable point is that the computational time of WLNM is more in PB, NS graphs than USAir as USAir is the smallest graph whereas SEAL reverses the case as it uses the node attributes in USAir. The highest computational time is recorded for SEAL among all the studied approaches.

We also see that the computational times for GNN-based approaches grow by more amount than the similarity-based approaches. For example, the minimum computational time for PA in USAir grows by an amount of 629 milliseconds in YAGO3-10 graph whereas for SEAL it grows by an amount of 2189 milliseconds. Overall, similarity-based approaches are more efficient than GNN-based approaches concerning the computational time. Except for SEAL, the approach-wise comparison in terms of computational time shows that all approaches show the highest and lowest computational time in the largest experimental graph (YAGO3-10) and smallest graph (USAir) respectively, as expected. SEAL shows higher computational time in USAir than two sparse graphs (Router and Power) as it uses the attribute features of USAir and also the latter graphs have low average node degree.

5 Conclusion

In this paper, we study several link prediction approaches for homogeneous graphs from similarity-based and GNN-based learning categories with their working principles and limitations. The approaches were evaluated against ten benchmark graphs with different properties from various domains. The precision of similarity-based approaches was computed in two different ways to overcome the difficulty of tuning the threshold for deciding the link existence based on the similarity score.

The experimental results show the superiority of GNN-based approaches over similarity-based ones with respect to the prediction performance across various graphs. In contrast, compared to similarity-based approaches, these GNN-based approaches are less suitable when the graphs need fast processing. The computational time of GNN-based approaches is further affected when applied to large graphs. In addition, the 'black box' problem of conventional neural networks remains unsolved with GNNs where it is very difficult to retrace the internal process of GNN. This work could help a new user to study similarity and GNN-based link prediction approaches and also the corresponding evaluation protocols.

One perspective of this work is to achieve a good trade-off between prediction accuracy and computational time by developing a GNN-based link approach in a distributed and parallel environment. In addition, the approach is expected to be applicable to the heterogeneous graphs such as knowledge graphs.

References

1. Xu, Z., Pu, C., Yang, J.: Link prediction based on path entropy. Physica A: Statistical Mechanics and its Applications, **456**, pp. 294–301, (2016).
2. Shen, Z., Wang, W. X., Fan, Y., Di, Z., Lai, Y. C.: Reconstructing propagation networks with natural diversity and identifying hidden sources. Nature communications, **5**(1), pp. 1–10, (2014).
3. Adamic, L. A., Adar, E.: Friends and neighbors on the web. Social networks, **25**(3), pp. 211–230 (2003).

4. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer, **42**(8), pp. 30–37,(2009).
5. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. In Proceedings of the IEEE, **104**(1), pp. 11–33, (2015).
6. Airoldi, E. M., Blei, D. M., Fienberg, S. E., Xing, E. P.: Mixed membership stochastic blockmodels. Journal of Machine Learning Research, **9**(Sep), 1981–2014 (2008).
7. Kovács, I. A., Luck, K., Spirohn, K., Wang, Y., Pollis, C., Schlabach, S., ..., Calderwood, M. A.: Network-based prediction of protein interactions. Nature Communications, **10**(1), pp. 1–8,(2019).
8. Paoletti, M. E., Haut, J. M., Plaza, J., Plaza, A.: A new deep convolutional neural network for fast hyperspectral image classification. ISPRS Journal of Photogrammetry and Remote Sensing, **145**, pp. 120–147, Elsevier, (2018).
9. Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., ..., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, **29**(6), pp. 82–97, IEEE, (2012).
10. Redmon, J., Divvala, S., Girshick, R., Farhadi, A: You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788, (2016).
11. Luong, M. T., Pham, H., Manning, C. D.: Effective approaches to attention-based neural machine translation. In Proceedings of the Empirical Methods in Natural Language Processing, pp. 1412–1421, (2015).
12. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S. Y.: A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, pp. 1–21, (2020).
13. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In Proceedings of the International Conference on Learning Representations, pp. 1–12, (2018).
14. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks, **20**(1), pp. 61–80, (2008).
15. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In Advances in Neural Information Processing Systems, pp. 4800–4810, (2018).
16. Kipf, T. N., Welling, M.: Semi-supervised classification with graph convolutional networks. In Proceedings of the International Conference on Learning Representations, pp. 4700–4708, (2016).
17. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. In Advances in Neural Information Processing Systems, pp. 5165–5175, (2018).
18. Martínez, V., Berzal, F., Cubero, J. C.: A survey of link prediction in complex networks. ACM Computing Surveys (CSUR), **49**(4), pp. 1–33, (2016).
19. Lorrain, F., White, H. C.: Structural equivalence of individuals in social networks. The Journal of Mathematical Sociology, **1**(1), pp. 49–80, Taylor & Francis, (1971).
20. Zhou, T., Lü, L., Zhang, Y. C.: Predicting missing links via local information. The European Physical Journal B, **71**(4), pp. 623–630, Springer, (2009).
21. Barabási, A. L., Albert, R.: Emergence of scaling in random networks. Science, **286**(5439), pp. 509–512, American Association for the Advancement of Science, (1999).
22. Jaccard, P.: Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bull Soc Vaudoise Sci Nat, **37**, pp. 547–579, (1901)

23. Salton, G., McGill, M.: Introduction to modern information retrieval, pp. 448, McGraw-Hill, New York (1983).
24. Sørensen, T., Sørensen, T. A., Sørensen, T. J., Biering-Sørensen, T.: A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons, (1948).
25. Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., Barabási, A. L.: Hierarchical organization of modularity in metabolic networks. Science, **297**(5586), American Association for the Advancement of Science, (2002).
26. Leicht, E. A., Holme, P., Newman, M. E.: Vertex similarity in networks. Physical Review E, **73**(2), pp. 026120, (2006).
27. Dong, Y., Ke, Q., Wang, B., Wu, B.: Link prediction based on local information. In 2011 International Conference on Advances in Social Networks Analysis and Mining, pp. 382–386, IEEE, (2011, July).
28. Cannistraci, C. V., Alanis-Lobato, G., Ravasi, T.: From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. Scientific Reports, **3**(1), pp. 1–14, Nature, (2013).
29. Wu, Z., Lin, Y., Wang, J., Gregory, S.: Link prediction with node clustering coefficient. Physica A: Statistical Mechanics and its Applications, **452**, pp. 1–8, (2016).
30. Zhang, Z., Cui, P., Zhu, W.: Deep learning on graphs: A survey. IEEE Transactions on Knowledge and Data Engineering, (2020).
31. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K. I., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In International Conference on Machine Learning, pp. 5453–5462, (2018).
32. Zhang, M., Chen, Y.: Weisfeiler-lehman neural machine for link prediction. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining pp. 575–583, (2017, August).
33. Huang, W., Zhang, T., Rong, Y., Huang, J.: Adaptive sampling towards fast graph representation learning. In Advances in Neural Information Processing Systems, pp. 4558–4567, (2018).
34. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. In Advances in Neural Information Processing Systems, pp. 9240–9251, (2019).
35. Zitnik, M., Agrawal, M., Leskovec, J.: Modeling polypharmacy side effects with graph convolutional networks. Bioinformatics, **34**(13), pp. 457–466, (2018).
36. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 974–983, (2018, July).
37. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems, pp. 1024–1034, (2017).
38. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In 32nd AAAI Conference on Artificial Intelligence, (2018, April).
39. Xu, K., Hu, W., Leskovec, J., Jegelka, S. (2018). How powerful are graph neural networks?. In International Conference on Learning Representations, (2019).
40. Weisfeiler, B., Lehman, A. A.: A reduction of a graph to a canonical form and an algebra arising during this reduction. Nauchno-Technicheskaya Informatsia, **2**(9), pp. 12–16, (1968).

41. McKay, B. D., Piperno, A.: Practical graph isomorphism, II. Journal of Symbolic Computation, **60**, pp. 94-112, Elsevier, (2014).
42. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864, (2016).
43. Salgado, H., Santos-Zavaleta, A., Gama-Castro, S., Millán-Zárate, D., Díaz-Peredo, E., Sánchez-Solano, F., ... , Collado-Vides, J.: RegulonDB (version 3.2): transcriptional regulation and operon organization in Escherichia coli K-12. Nucleic Acids Research, **29**(1), pp. 72–74, (2001).
44. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In Advances in Neural Information Processing Systems, pp. 2787–2795, (2013).
45. Newman, M. E.: Finding community structure in networks using the eigenvectors of matrices. Physical review E, **74**(3), pp. 036104 (2006).
46. Ackland, R. : Mapping the US political blogosphere: Are conservative bloggers more prominent?. In BlogTalk Downunder 2005 Conference, Sydney (2005).
47. Watts, D. J., Strogatz, S. H.: Collective dynamics of 'small-world'networks. Nature, **393**(6684), pp. 440, (1998).
48. Spring, N., Mahajan, R., Wetherall, D.: Measuring ISP topologies with Rocketfuel. ACM SIGCOMM Computer Communication Review, **32**(4), 133-145, (2002).
49. Handcock, M. S., Hunter, D., Butts, C. T., Goodreau, S. M., Morris, M.: Statnet: An R package for the Statistical Modeling of Social Networks. (2003). Web page http://www. csde. washington. edu/statnet.
50. Bordes, A., Glorot, X., Weston, J., Bengio, Y.: A semantic matching energy function for learning with multi-relational data. Machine Learning, **94**(2), pp. 233–259, Springer, (2014).
51. Mahdisoltani, F., Biega, J., Suchanek, F. M.: Yago3: A knowledge base from multilingual wikipedias, In 7th Biennial Conference on Innovative Data Systems Research, Asilomar, United States, (2013, January).
52. Von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S. G., Fields, S., Bork, P.: Comparative assessment of large-scale datasets of protein–protein interactions. Nature, **417**(6887), pp. 399–403, (2002).
53. Bastian, M., Heymann, S., Jacomy, M.: Gephi: an open source software for exploring and manipulating networks. In 3rd International AAAI Conference on Weblogs and Social Media, pp. 17–20, (2009, March).
54. Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., ... , Huang, Z. (2019). Deep graph library: Towards efficient and scalable deep learning on graphs. In ICLR Workshop on Representation Learning on Graphs and Manifolds, (2019).
55. Yang, J., Zhang, X. D.: Predicting missing links in complex networks based on common neighbors and distance. Scientific Reports, **6**, p. 38208, Nature Publishing Group, (2016).
56. Pan, L., Zhou, T., Lü, L., Hu, C. K.: Predicting missing links and identifying spurious links via likelihood analysis. Scientific Reports, **6**(1), pp. 1–10, Nature Publishing Group, (2016).
57. Wu, Z., Lin, Y., Zhao, Y., Yan, H.: Improving local clustering based top-L link prediction methods via asymmetric link clustering information. Physica A: Statistical Mechanics and its Applications, **492**, pp. 1859–1874, Elsevier, (2018).