An Experimental Evaluation of Similarity-Based and Embedding-Based Link Prediction Methods on Graphs

Md Kamrul Islam, Sabeur Aridhi, Malika Smaïl-Tabbone

An experimental evaluation of similarity-based and embedding-based link prediction methods on graphs

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 missing links or predicting future ones 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 and show good prediction performance in many real-world graphs though they are heuristic. On the other hand, graph embedding approaches learn low-dimensional representation of nodes in graph and are capable of capturing inherent graph features, and thus support the subsequent link prediction task in graph. This paper studies a selection of methods from both categories on several benchmark (homogeneous) graphs with different properties from various domains. Beyond the intra and inter category comparison of the performances of the methods, our aim is also to uncover interesting connections between Graph Neural Network(GNN)-based methods and heuristic ones as a means to alleviate the black-box well-known limitation.

KEYWORDS

Link Prediction, Graph Neural Network, Homogeneous Graph & 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], and knowledge graph completion [5].

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 any kind 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

DOI:10.5121/ijdkp.2021.11501

(PPI) networks [6]. In case of using these heuristics, it is required to manually choose different heuristics for different graphs based on prior beliefs or rich expertise.

On the other hand, machine learning methods have shown their impressive performance in many real-world applications like image classification, natural language processing etc. The built models assume that the input data is represented as independent vectors in a vector space. This assumption is no longer applicable for graph data as graph is a non-Euclidean structure and the nodes in a graph are linked to some other nodes [7]. To overcome this limitation, a lot of efforts have been devoted to develop novel graph embeddings where the nodes, edges, graphs are represented in a low-dimensional vector space. In last decade, graph embedding has been established as a popular supporting tool for solving several analytical problems in graphs like node classification, node clustering, link prediction. The embedding approaches represent a part of a graph (or the whole graph) in a low dimensional vector space while preserving the graph information [8].

There are some review studies in the literature which focus either on similarity-based approaches [9], [10] or embedding-based approaches [8], [11] for link prediction task in graphs. Thus, to the best of our knowledge, a study including methods from both categories is missing in the literature. In this paper, we try to fill this gap. This is an extended version of our conference paper [12]. We first introduce the link prediction problem and briefly describe selected similarity-based and embedding-based methods. Then, we evaluate their performances on different types of graphs, namely homogeneous graphs. We compare their performances on diverse graph groups (sharing characteristics). We also propose a few interesting connections between similarity-based and embedding-based methods. Finally, we list some future research works.

2 LINK PREDICTION 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 $t + \Delta t$ based on the snapshot of the graph at time 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.

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

Figure 1: Illustration of link prediction problem

Several link prediction approaches have been proposed in the literature. We focus on the two popular categories: (1) similarity-based approaches and (2) embedding-based approaches.

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 graph interact if they are similar. Generally, the links with high similarity scores are predicted as truly missing links. 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 [9]. As a result, numerous similarity-based approaches have been proposed 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 approaches is global approaches that 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 [9]. For this reason, we are considering only the local similarity-based approaches in the current study. We have studied six popular similarity-based approaches for link prediction. Considering the citations for a duration from publishing to the running year(i.e. 2020), we define popularity of each approach as the average citation per year.

Table 1 summarizes the approaches with the basic principle and similarity function. These approaches in Table 1 except CCLP(Clustering Coefficient-based Link Predic-

Table 1: Summary of studied similarity-based approaches. The similarity function is defined to predict a link between two nodes *x* and *y*. Γ*x* and Γ*y* denote the neighbour sets of nodes *x* and *y* respectively.

Approach	Principle	Similarity-function
Adamic-Adar (AA) [3]	Variation of CN where each common neighbour is logarithmically penalized by its degree	$S^{AA}(x,y) = \sum_{z \in \Gamma x \cap \Gamma y} \frac{1}{log\lvert \Gamma z \rvert}$
Resource Allocation (RA) [13]	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) [14]	Based on the rich-get-richer concept where the link probability between two high degree nodes is higher than two low degree nodes	$S^{PA}(x,y) = \lvert \Gamma x \rvert \times \lvert \Gamma y \rvert$
Hub Promoted Index (HPI) [15]	Promoting link formation between high-degree nodes and hubs	$S^{HPI}(x,y) = \frac{\lvert \Gamma x \cap \Gamma y \rvert}{max(\lvert \Gamma x \rvert, \lvert \Gamma y \rvert)}$
Local Leicht-Holme-Newman (LLHN) [16]	Utilizing both of real and expected amount of common neighbours between a pair of nodes to define their similarity	$S^{LLHN}(x,y) = \frac{\lvert \Gamma x \cap \Gamma y \rvert}{\lvert \Gamma x \rvert \times \lvert \Gamma y \rvert} \sum_{z \in \Gamma x \cap \Gamma y} CC_z$
Clustering Coefficient-based Link Prediction (CCLP) [17]	Quantification of the contribution of each common neighbour by utilizing the local clustering coefficient of nodes	$S^{CCLP}(x,y) = \sum_{z \in \Gamma x \cap \Gamma y} CC_z$

tion) [17] use node degree, common neighborhood or links among common neighborhood

information to compute similarity scores. AA, RA and CCLP handcraft the computation of weight of each common neighbours based on their neighbourhood size or clustering co-efficient(CC). The clustering coefficient is defined as the ratio of the number of triangles and the expected number of triangles passing through a node [17]. 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 in Equation 1.

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

On the other hand, HPI, PA and LLHN assigns equal weights to neighbours. 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, LLHN suffers from outlier (infinite similarity score) when one of the end nodes has no neighbour. HPI also suffers from the outlier (infinite similarity score) when both of end nodes have no neighbour.

Graph embedding-based link prediction

A graph embedding approach embeds the nodes of a graph into low-dimensional vector space where connected nodes are closer to each other. The embedding vector of a link is then computed based on the embedding of end nodes and a classifier is used to classify it as existent or non-existent link. Random walk-based and neural network-based embedding are three popular methods of embedding [8]. The first one samples the nodes based on the random walk process in graph and adopts skip-gram model to represents them in a low-dimensional vector. The second category is designed based on neural network(NN). The success of NN in image, speech, text processing where data can be represented in Euclidean form, motivates researchers to study GNNs as a kind of NN that operates directly on graphs. GNNs provide an end-to-end graph embedding [8]. In our study, we are interested in two specific GNN architectures called convolution GNN (ConvGNN) and graph auto-encoder [7]. Inspired by the convolution operation of NN, ConvGNNs compute the embedding of a node by aggregating its own and neighbours information.The graph auto-encoders are auto-encoders for learning representations of graphs. The general idea of auto-encoders is simple which consist of a NN-based encoder and a decoder to compute embedding of a graph using an optimization procedure [18]. In the following, we present six embedding-based link prediction approaches including one random-walk based (Node2Vec), three CNN-based (WLNM, SEAL, GAT), and two auto-encoder based (VGAE, LVGAE). We choose Node2Vec to represent simple non-deep learning methods, WLNM to represent the methods which learn only structural features, SEAL to represent the methods which maximize the use of available information (structural, node attributes, latent features), GAT to represent the methods which define different roles of different neighbours, VGAE to represent auto-encoder with deep neural networks, and LVGAE to represent simple linear auto-encoder.

Node2Vec:

Motivated by the classical skip-gram model in natural language processing, Grover& Leskovec [19] developed Node2Vec representation learning approach that optimizes a neighbourhood preserving objective function using Stochastic Gradient Descent (SGD). Node2Vec starts with a fixed size neighbourhood sampling using guided random walk. Unlike the classical random walk, Node2Vec defines a 2^{nd} order random walk that inter-polate between BFS(Breadth First Search) and DFS(Depth First Search)-based sampling

strategy where two parameters p and q are used to compute the transition probability during the walk. These parameters control how fast the walk explores and leaves the neighborhood of the starting node. The node embedding is then generated based on the popular skip-gram model where the co-occurrence probability among the neighbours those appear within a window.

Weisfeiler-Lehman Neural Machine (WLNM):

Based on the well-known Weisfeiler-Lehman(WL) canonical labelling algorithm [20], Zhang & Chen [21] developed the Weisfeiler-Lehman Neural Machine (WLNM) to learn the structural features from the graph and use it in the link prediction task.

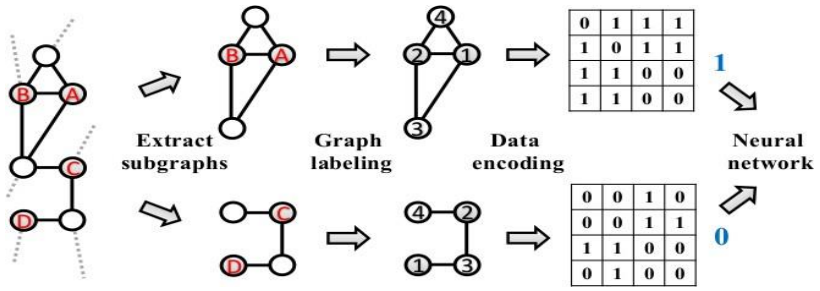

Figure 2: Illustration of WLNM [21] with existent(A,B) and non-existent link(C,D)

As illustrated in Fig. 2, WLNM is a three steps link prediction approach that starts with extracting sub-graphs those contain a predefined number of neighbour nodes, labelling and encoding the nodes in the sub-graph using WL algorithm and ends with training and evaluating the neural network. WLNM is a simple GNN-based link prediction approach which is able to learn the link prediction heuristics from a graph. The downside of WLNM is that it truncates some neighbours to limit the sub-graph size to a user-defined size which are may be informative for the prediction task.

Learning from Sub-graphs, Embeddings and Attributes (SEAL):

Zhang & Chen [22] developed a ConvGNN-based link prediction approach called 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. The overall architecture of the approach is shown in Fig. 3. Like WLNM, SEAL also consists of three

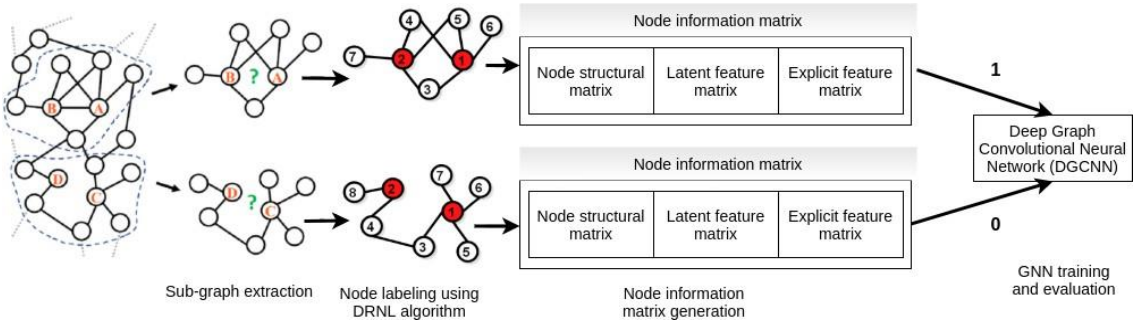

Figure 3: Architecture of SEAL approach [22]

major steps: (1) sub-graph extraction and node labelling, (2) node information matrix construction, and (3) neural network training and evaluation. 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.

Graph Attention Networks (GAT):

In Graph Convolutional Networks (GCN) [23], the convolution operation is defined based on close neighbors where all neighbors contribute equally which affects the prediction performance. To overcome this shortcoming, Velickovic et al. [24] presents GAT by leveraging attention mechanism for learning different weights (or coefficients) to different nodes in a neighborhood. The attention learning mechanism starts with defining a graph attention layer where the input is the set of node features, $\mathrm{h}=\{\vec{h}_1, \vec{h}_2, .., \vec{h}_N\}$ for N nodes. The layer produces a transformed set of node feature vectors $\mathrm{h}'=\{\vec{h}'_1, \vec{h}'_2 .., \vec{h}'_N\}$, where h_i and h'_i are input and output embeddings of the node e_i. The attention layer is defined as Equation 2.

$$c_{ij} = f_a(W\vec{h}_i, W\vec{h}_j) \tag{2}$$

where c_{ij} is the attention coefficient of the edge (e_i, e_j), $\vec{h}_i, \vec{h}_j$ are embeddings of nodes e_i, e_j, W is a parametrized linear transformation matrix mapping the input features to a higher dimensional output feature space, and f_a is a shared attention mechanism. GAT uses the LeakyReLU nonlinearity as the activation function of the attention layer. The coefficient indicates the importance of node e_j to node e_i. GAT uses the following softmax function (Equation 3) over the first order neighbours of a node including itself to compute the normalized attention coefficient, α_{ij} of the edge (e_i, e_j).

$$\alpha_{ij} = softmax(c_{ij}) = \frac{exp(c_{ij})}{\sum_{k \in N_i} exp(c_{ik})} \tag{3}$$

where N_i is the set of neighbours for node e_i. The output embedding of the node e_i is generated using the attention coefficients as in Equation 4.

$$\vec{h}'_i = \sum_{j \in N_i} \alpha_{ij} W\vec{h}_j \tag{4}$$

GAT extends the single head concept to multi-head mechanism to learn more stable attentions by averaging the coefficients over multi-head attentions. For link prediction, the embedding of end nodes are feed into a fully connected NN.

Variational Graph Auto-encoder (VGAE):

The above discussed approaches are supervised link prediction approaches. In contrast, Variational graph auto-encoder (VGAE) [25] is an unsupervised approach which uses the well-known variational auto-encoder (VAE) framework [26] to learn graph-structured data. The main idea of VAE is that it represents an input to a multivariate Gaussian distribution rather than a point in low dimensional vector space and then the decoder randomly sample a low-dimensional embedding from the distribution to reconstruct the input from this embedding [26]. Likewise traditional auto encoder frameworks, VGAE consists of two components, an encoder and a decoder. The encoder of VGAE is a 2-layer GCN which

takes the adjacency matrix (A), degree matrix(D), and a feature matrix(F) of graph as inputs. The first layer generates a low-dimensional feature matrix as in Equation 5.

$$X = GCN(F, A) = ReLU(A^{J}F^{J}W_0) \quad with\ A^{J} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \tag{5}$$

where W_0 is the weight matrix of first layer. The second layer generates the mean(μ) and standard deviation(σ) of the Gaussian distribution as in Equation 6.

$$\mu = GCN_{\mu}(X, A) = A^{J}XW_{\mu} \qquad log\sigma^2 = GCN_{\sigma}(X, A) = A^{J}XW_{\sigma} \tag{6}$$

where W_{μ}, W_{σ} are weight matrices of second layer. The embeddings(Z) of nodes are computed using parameterization trick as in Equation eq:zcal

$$Z = \mu + \sigma * \epsilon \quad with\ \epsilon = \mathrm{N}(0, 1) \tag{7}$$

The decoder is a generative model which is defined as the inner product between latent variables to reconstructed adjacency matrix $\hat{A}$, which is defined as

$$\hat{A} = \sigma(Z^{T}Z) \tag{8}$$

where $\sigma(.)$ is the logistic sigmoid function. The weight matrices are learned by minimizing the reconstruction loss capturing the similarity of A and $\hat{A}$.

Linear Variational Graph Auto-encoder(LGVAE):

To reduce the computational complexity of VGAE, LGVAE [27] replaces the GCN in VGAE with a straightforward linear model w.r.t. the normalized adjacency matrix. In constast to GCN encoders in VGAE, LGVAE aggregates information from immediate neighbors of a node to learn its embeddings.

$$\mu = A^{J}FW_{\mu}, \qquad log\sigma = A^{J}FW_{\sigma} \tag{9}$$

The node embeddings(Z) are computed using the same way (Equation 10) as in VGAE.

$$Z = \mu + \sigma * \epsilon \quad with\ \epsilon = \mathrm{N}(0, 1) \tag{10}$$

The decoder remains same as the decoder in VGAE (8). The weight matrices W_{μ} and W_{σ} are learned by minimizing the reconstruction loss as in VGAE.

3 EXPERIMENTAL DESIGN

Experimental data

We perform the comparative study of the above discussed similarity and embedding based link prediction approaches in simple and undirected graphs from different domains. To evaluate and describe the performance of the link prediction approaches, we choose ten benchmark graphs from different areas: Ecoli [28], FB15K [29], NS [30], PB [31],Power [32], Router [33], USAir [34], WN18 [35], YAGO3-10 [36], and Yeast [37]. FB1K, WN18 and YAGO3-10 are popular knowledge graphs. These knowledge graphs consist of subject-relationship type-object triples. However, as the studied approaches are applicable to homogeneous graphs only. We simplify these knowledge graphs by overlooking the relation names and considering links as undirected links.

The topological statistics 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 10,000 nodes and large graphs with more than 10,000 nodes.

Table 2: Topological statistics of graph datasets: number of nodes(#Nodes), links(#Links), average node degree (NDeg), clustering coefficient (CC), network diameter (Diam) and description. Large graphs are shaded with gray color.

Graphs	#Nodes	#Links	NDeg	CC	Diam	Description
Ecoli	1805	42325	46.898	0.350	10	Nodes: Operons in E.Coli bacteria Edges: Biological relations between operons
FB15K	14949	260183	44.222	0.218	8	Nodes: Identifiers of Freebase KB entity Edges: Link between Freebase entities
NS	1461	2742	3.754	0.878	17	Nodes: Researchers who publish papers on network science Edges: Co-authorship of at least one paper
PB	1222	14407	23.579	0.239	8	Nodes: US political blog page Edges: Hyperlinks between blog pages
Power	4941	6594	2.669	0.107	46	Nodes: Electrical power stations (e.g. generators, transformers) of western US Edges: Power transmission between stations
Router	5022	6258	2.492	0.033	15	Nodes: Network router Edges: Router-router interconnection for providing router-level internet
USAir	332	2126	12.807	0.749	6	Nodes: US airports Edges: Link between two airports if there is at least one direct flight between them
WN18	40943	75769	3.709	0.077	18	Nodes: Entities (or synsets) corresponds to English word senses Edges: Lexical relations between sysnets
YAGO3-10	113273	758225	18.046	0.114	14	Nodes: Entities (such as movies, people, cities, etc.) in YAGO KB Edges: Relations between entities
Yeast	2375	11693	9.847	0.388	15	Nodes: Proteins in yeast Edges: Protein-protein interaction in yeast network

Construction of train and test sets

We follow a random sampling protocol to evaluate the performance of the studied approaches [21]. We prepare train and test set from the experimental graphs. For training dataset, we randomly select 90% existing links (termed as positive train set) and an equal number of non-existing links (termed as negative train set). The remaining 10% existing links (termed as positive test set) and an equal number of non-existing links (termed as negative test set) form the test set. At the same time, the graph connectivity of the training set and the test set is guaranteed. We prepare five train and five test sets 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 embedding-based approaches, the graph is built from 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 embedding-based approaches or to predict its existence. The performance 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 memory and core i7 (CPU 1.90GHz) processor.

Precision and AUC computation

Precision describes the fraction of missing links which are accurately predicted as existent links [38]. To compute the precision, 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 Equation 11.

$$Precision = \frac{L_r}{L} \tag{11}$$

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.However, there are some challenges with this optimistic way of computing the precision. What if the similarity score is (close to) 0.0 of the lowest ranked link? This issue creates the difficulty to make a separation between some positive and negative test links. Choosing a threshold when defining L_r could be a potential solution to overcome this problem. The distribution of unnormalized similarity scores are different for graphs from different domains and even for two different datasets from the same domain. Moreover, it is nearly impossible to know the distribution of unnormalized similarity score in advance for graph dataset. These two facts make it infeasible task for the user to define the threshold. 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.

On the other hand, the metric AUC is defined as the probability that a randomly chosen existing link has a higher similarity score than a randomly chosen non-existing link [38]. 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 Equation 12.

$$AUC = \frac{n_1 + 0.5n_2}{n} \tag{12}$$

We consider half of the total links in the positive test set and negative test set to compute AUC.

The precision scores talk about how many actually positive links exist in the set of predicted positive links. However, it is also important to assess the false positive or false negative results of a link prediction model as the higher the false-positive result, the lower the efficiency of the model [39], [40]. We use the above computed threshold to classify a link into positive or negative classes. A test link is predicted as positive if the similarity score of the link is greater or equal to the threshold and predicted as negative otherwise. For embedding-based methods, the threshold is learned and test links are already predicted as positive or negative. Based on the true and predicted link existence, four metrics are computed and they are true positives (tp), false negatives (fn), true negatives (tn), false positives (fp). True positives (tp) are the cases where existent test links are predicted as positive and false negatives (fn) are the cases where the existent links are predicted as negatives. Similarly, non-existent test links are predicted as negatives in true negative (tn) cases and the links are predicted as positive in false positive (fp) cases. We compute two standard metrics (false positive rate, false negative rate) to assess the false results and one metric (accuracy) to assess over accuracy of a model. We use the standard definition

of the metrics as Equation 13

$$\begin{aligned} False\ positive\ rate(FPR) &= \frac{fp}{fp+tn} \\ False\ negative\ rate(FNR) &= \frac{fn}{tp+fn} \\ Accuracy(ACC) &= \frac{tp+tn}{tp+fn+tn+fp} \end{aligned} \tag{13}$$

4 EXPERIMENTAL RESULTS

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. We measure the precision in two different ways based on the top-L test links as described in Section 3.3. We compute the threshold-based precision only for similarity-based approaches as embedding-based approaches do learn the threshold. The auto-encoder based VGAE and LVGAE approaches require the whole adjacency matrix of a graph on memory. Due to the memory constaint, experiments with auto-encoder based VGAE and LVGAE approaches for large scale graphs (WN18, YAGO3-10) failed. The maximum and minimum similarity scores are computed from the top-L for each test set of each graph. Table 3 shows the results in each of the seven small/medium and three large-size graphs. Each value of the table is the mean over the five test sets. The standard deviation values of both metrics for all approaches in all graphs are very small and they are not included in the table.

It can be clearly seen from Table 3 that the ranges of unnormalized similarity scores are different for different similarity-based approaches and also different in different datasets for the same similarity-based approach. Moreover, the minimum similarity scores are very low (close to 0) in some datasets. These observations prove that in real-world applications, it is difficult to choose a threshold and to assess good precision for similarity-based approaches.

From Table 1, the similarity-based approaches are mostly defined based on the common neighbourhood. As expected, they show low precision (without defining threshold) and AUC values in sparse graphs (low CC, low node degree) like Power, router and high precision for other well-connected graphs in Table 3. Exceptionally, PA shows better prediction performance in sparse graphs as it considers individual node degree instead of common neighbourhood for computing similarity score. The precision scores using the threshold-based method drops drastically in most of the cases as many falsely predicted positive links are identified (i.e. predicted links with very low scores). Surprisingly, HPI shows competitive threshold-based precision value in NS dataset. No single similarity-based approach wins in all small/medium size graphs.

As expected, embedding-based approaches show very good precision and AUC scores across all of the the small/medium size graphs compared to similarity-based approaches. What about their comparative performances? No single approach wins in all datasets. Node2Vec shows highest precision scores in some datasets though it is simpler than other embedding-based approaches. The consideration of more distant neighbours in embedding computation during random walk could be the most possible reason behind this success. The use of latent information along with structural information in SEAL for the datasets during prediction task likely explains the improvement of the metric AUC. The best tuning of parameters could be the most possible reason behind the best balance between the prediction metrics in GAT. The precision and AUC scores of auto-encoder

Table 3: AUC and Precision(Prec) values with Max Scores(Mx scr) and Min Scores (Mn scr). Precision with * mark(Prec*) is computed based on threshold in top-L links. Graph-wise highest metrics are indicated in bold fonts while approach-wise highest metrics are shown in underline. – denotes the failed experiment.

App.	Metric	Ecoli	NS	PB	Power	Router	USAir	Yeast	FB15K	WN18	YAGO 3-10
AA	Prec	0.90	0.87	0.86	0.17	0.07	0.92	0.83	0.77	0.13	0.15
	Prec*	0.06	0.15	0.01	0.02	0.01	0.16	0.06	0.0002	0.0002	0.0018
	Mx scr	32.84	5.83	33.41	3.04	5.60	16.69	23.71	418.60	57.32	24.44
	Mn scr	2.86	1.14	0.58	0.00	0.00	2.70	0.00	0.12	0.00	0.00
	AUC	0.93	0.94	0.92	0.58	0.54	0.94	0.91	0.82	0.56	0.48
PA	Prec	0.78	0.69	0.83	0.49	0.41	0.85	0.79	0.79	0.63	0.83
	Prec*	0.05	0.02	0.01	0.02	0.01	0.13	0.06	0.0003	0.0006	0.0006
	Mx scr	65679	362.0	61052	53.0	2397	8298	10642	9881842	10637	2426939
	Mn scr	3532	12.0	855.7	4.0	1.0	739.3	95.0	942.67	6.33	109.00
	AUC	0.80	0.66	0.90	0.46	0.43	0.90	0.86	*0.88*	*0.64*	0.88
RA	Prec	0.91	0.87	0.86	0.17	0.07	0.92	0.83	0.77	0.13	0.15
	Prec*	0.03	0.15	0.01	0.03	0.01	0.10	0.07	0.0003	0.0002	0.0011
	Mx scr	1.70	1.80	4.19	0.84	1.32	2.83	2.37	72.06	20.67	5.16
	Mn scr	0.19	0.40	0.03	0.00	0.00	0.32	0.00	0.00	0.00	0.00
	AUC	0.94	0.94	0.92	0.58	0.54	0.94	0.91	0.84	0.57	0.57
HPI	Prec	0.90	0.87	0.80	0.17	0.07	0.91	0.83	0.69	0.13	0.15
	Prec*	0.20	0.96	0.15	0.13	0.02	0.45	0.70	0.0959	0.0796	0.0476
	Mx scr	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Mn scr	0.33	0.83	0.21	0.00	0.00	0.77	0.00	0.05	0.00	0.00
	AUC	0.94	0.94	0.85	0.58	0.54	0.91	0.90	0.75	0.56	0.47
LLHN	Prec	0.89	0.87	0.74	0.17	0.07	0.87	0.83	0.64	0.13	0.15
	Prec*	0.001	0.13	0.001	0.03	0.003	0.03	0.01	0.0008	0.0046	0.0003
	Mx scr	0.32	1.00	0.42	2.06	0.83	0.58	0.67	0.28	1.00	1.00
	Mn scr	0.00	0.10	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
	AUC	0.91	0.93	0.76	0.58	0.53	0.77	0.90	0.57	0.57	0.45
CCLP	Prec	**0.96**	0.73	0.86	0.08	0.07	0.91	0.82	0.78	0.08	0.14
	Prec*	0.06	0.21	0.01	0.01	0.01	0.18	0.06	0.0015	0.0006	0.0013
	Mx scr	30.6	8.0	27.0	1.2	1.1	21.1	39.2	51.74	1.67	20.77
	Mn scr	1.8	0.3	0.3	0.0	0.0	2.9	0.0	0.01	0.00	0.00
	AUC	**0.95**	0.87	0.91	0.54	0.53	0.94	0.90	0.84	0.54	0.57
WLNM	Prec	0.87	0.84	0.78	0.84	**0.89**	0.85	0.87	0.67	**0.84**	0.68
	AUC	0.93	0.95	0.93	0.76	0.92	0.86	0.86	0.68	0.79	0.72
SEAL	Prec	0.81	*0.96*	0.80	0.66	0.80	**0.91**	0.89	0.77	0.61	**0.86**
	AUC	**0.95**	**0.99**	**0.94**	0.77	**0.94**	**0.94**	**0.98**	**0.96**	**0.87**	**0.97**
GAT	Prec	0.84	0.93	0.84	0.72	0.81	0.88	**0.91**	**0.85**	0.74	0.84
	AUC	0.85	0.90	0.86	0.70	0.79	0.87	0.89	0.87	0.79	0.83
Node2Vec	Prec	0.91	**0.97**	**0.91**	**0.86**	0.80	0.81	0.85	0.79	0.83	0.82
	AUC	0.90	0.96	0.90	**0.82**	0.75	0.85	0.94	0.88	0.79	0.80
VGAE	Prec	0.75	0.91	0.86	0.82	0.80	0.78	0.75	0.81	–	–
	AUC	0.76	0.93	0.87	0.83	0.84	0.80	0.75	0.78	–	–
LVGAE	Prec	0.76	0.88	0.73	0.81	0.81	0.79	0.86	0.74	–	–
	AUC	0.77	0.86	0.73	0.86	0.83	0.80	0.83	0.68	–	–

based approaches are behind the other embedding-based approaches. Improper tuning of parameters could be the possible reason behind the performance degradation. Table 3 shows that embedding-based approaches provide high-performance metrics in all graphs while similarity-based approaches perform well in some graphs (in terms of optimistic precision).

Considering the three large graphs (FB15K, WN18 and YAGO3-10), the prediction metrics for similarity-based approaches are much lower than small/medium scale graphs, especially in WN18 and YAGO3-10 graphs. Likewise the results in small/medium size graphs, the precision scores of these approaches further drops drastically to less than 0.1 when applying the threshold. Unsurprisingly, the prediction scores for embedding-based approaches in large graphs are high as in small/medium scale graphs. The notable point in the prediction metrics for large graphs is that Node2Vec is less competitive than other embedding-based approaches in these large graphs.

We further compute the false positive rate, false negative rate and overall accuracy of the approaches which are tabulated in Table 4. We compute the thresholds for similarity-based approaches based on their top-L links scores which are still not optimal. Many of existent and most of the test links have very low similarity scores. As a result, false positive rates of similarity-based approaches are very low and false negative rates is very high for almost all graph datasets in Table 4. The overall accuracy scores of these approaches are around 0.5. Exceptionally, HPI shows good accuracy in some datasets, especially in NS dataset where the accuracy is more than 0.9. On the other hand, embedding-based approaches learn the threshold. The false positive rates of embedding-based approaches are much higher than the similarity-based approaches, but the false negative rates are much much lower than the similarity-based approaches. Among embedding-based approaches, the auto-encoder based VGAE and LVGAE have higher false negative rates than other embedding-based approaches. Overall, the accuracy scores of similarity-based approaches stay around 0.5 with few exceptions. As expected, the accuracy scores of embedding based approaches are much higher than the similarity-based approaches. With respect to accuracy, GAT shows highest/nearly highest scores in nearly half of the total graphs. And no single embedding-based link prediction approach wins in all graphs.

Embedding-based link prediction approaches show better performance because they learn heuristics from graphs. However, it is not clear which heuristic(s) are learned. We want to take benefit from this study to get insight of such heuristics by comparing the performances of similarity-based heuristics with performances of embedding-based approaches on the same datasets. In one hand, from Table. 1 and Table 3, AA, RA and CCLP –which heuristically assign high weights to nodes with high degrees or cluster coefficients – show better precision, accuracy on FB15K, PB, NS, USAir, and Yeast compared to other graphs. GAT also shows better precision and accuracy these graphs than other graphs. This may indicate that GAT learns similar heuristics as AA, RA and CCLP. In the other hand, WLNM considers the role of each neighbour equally like HPI, LLHN, and PA. WLNM, HPI, LLHN and PA show better performance scores on Power, Router, and WN18 graphs, confirming that they are heuristically compatible. The auto-encoder based VGAE and LVGAE approaches assign equal weights to all neighbours of a node when aggregating neighbourhood information and show competitive performance on Power and Router graphs. In addition, LVGAE aggregates hop-1 neighbours of a node using a simple linear model whereas VGAE aggregates upto hop-2 neighbours of a node using 2-layer GCN model. These indicate the possibility of learning HPI, LLHN and PA heuristics by LVGAE and AA, RA, CCLP by VGAE approach.

Table 4: False positive rate(FPR), false negative rate (FNR) and accuracy(ACC) scores. Graph-wise best metrics (lowest FPR, lowest FNR, highest ACC) are indicated in bold fonts. – denotes the failed experiment.

App.	Metric	Ecoli	NS	PB	Power	Router	USAir	Yeast	FB15K	WN18	YAGO 3-10
AA	FPR	0.0007	**0.0001**	**0.000**	0.0003	**0.0001**	0.0028	**0.0003**	**0.0001**	**0.0001**	**0.0001**
	FNR	0.962	0.898	0.991	0.977	0.996	0.932	0.956	0.998	0.997	0.998
	ACC	0.519	0.551	0.504	0.512	0.502	0.533	0.522	0.499	0.499	0.499
PA	FPR	0.0007	0.0007	0.0004	0.0018	**0.0001**	0.0019	**0.0003**	**0.0001**	**0.0001**	**0.0001**
	FNR	0.967	0.942	0.993	0.988	0.996	0.952	0.956	0.997	0.999	0.999
	ACC	0.516	0.529	0.503	0.505	0.502	0.523	0.522	0.499	0.499	0.498
RA	FPR	0.0002	**0.0001**	**0.000**	0.0003	**0.0001**	**0.0000**	**0.0003**	**0.0001**	**0.0001**	**0.0001**
	FNR	0.984	0.854	0.995	0.977	0.994	0.973	0.958	0.998	0.999	0.999
	ACC	0.508	0.573	0.502	0.511	0.503	0.514	0.521	0.499	0.499	0.498
HPI	FPR	0.0060	0.0058	0.0345	0.0012	0.0048	0.1175	0.0075	0.0067	0.001	0.0152
	FNR	0.753	0.160	0.863	0.877	0.974	0.531	0.412	0.904	0.920	0.953
	ACC	0.620	0.917	0.551	0.561	0.510	0.676	0.790	0.543	0.538	0.514
LLHN	FPR	**0.0001**	**0.000**	0.0008	0.0003	0.0026	0.0114	0.0007	0.0013	**0.0001**	0.0002
	FNR	0.999	0.826	0.999	0.985	0.998	0.988	0.994	0.999	0.995	0.999
	ACC	0.501	0.587	0.499	0.507	0.500	0.500	0.503	0.498	0.501	0.498
CCLP	FPR	0.0006	**0.0001**	**0.000**	**0.000**	**0.0001**	0.0019	0.0003	**0.0001**	**0.0001**	**0.0001**
	FNR	0.959	0.927	0.991	0.992	0.992	0.919	0.952	0.998	0.999	0.989
	ACC	0.520	0.536	0.504	0.504	0.504	0.539	0.524	0.494	0.499	0.499
WLNM	FPR	0.1005	0.1642	0.2059	0.1310	0.1125	0.128	0.1334	0.3767	0.1466	0.4108
	FNR	0.345	0.178	0.304	**0.323**	**0.048**	0.128	0.148	0.240	0.259	**0.145**
	ACC	0.777	0.829	0.745	0.773	**0.92**	0.872	0.859	0.692	0.797	0.722
SEAL	FPR	0.1774	0.0343	0.2214	0.2079	0.2323	0.0938	0.1044	0.2232	0.3492	0.1314
	FNR	0.231	0.070	0.123	0.591	0.083	0.079	0.154	0.251	0.449	0.190
	ACC	0.796	**0.948**	0.828	0.601	0.843	**0.914**	0.871	0.763	0.601	**0.839**
GAT	FPR	0.1828	0.0723	0.1909	0.2109	0.1328	0.1365	0.1025	0.1909	0.1848	0.1444
	FNR	**0.073**	**0.069**	**0.032**	0.461	0.428	**0.046**	**0.015**	**0.032**	0.3014	0.259
	ACC	**0.872**	0.93	**0.888**	0.664	0.72	0.909	**0.941**	**0.889**	0.7569	0.7983
Node2Vec	FPR	0.1457	0.0255	0.0826	0.1141	0.0541	0.1905	0.1454	0.1884	0.1594	0.1654
	FNR	0.221	0.161	0.195	0.334	0.791	0.209	0.202	0.279	**0.210**	0.237
	ACC	0.816	0.907	0.861	**0.776**	0.577	0.800	0.826	0.7663	**0.815**	0.7988
VGAE	FPR	0.2455	0.0358	0.1306	0.0243	0.0352	0.2427	0.2399	0.1881	–	–
	FNR	0.257	0.661	0.316	0.891	0.861	0.151	0.289	0.304	–	–
	ACC	0.749	0.652	0.777	0.542	0.552	0.803	0.736	0.753	–	–
LVGAE	FPR	0.1661	0.0971	0.2129	0.1478	0.1443	0.2464	0.1406	0.178	–	–
	FNR	0.477	0.319	0.435	0.368	0.400	0.082	0.172	0.514	–	–
	ACC	0.678	0.792	0.676	0.742	0.728	0.836	0.844	0.651	–	–

Table 5: Top-2 ranked similarity-based approaches with higher agreement with embedding-based approach for test link decision. Numbers in () represent the agreement percentages.

Graph	WLNM	SEAL	GAT	Node2Vec	VGAE	LGVAE
Ecoli	HPI(69), RA(69)	LLHN(80), RA(79)	HPI(70), RA(69)	RA(70), LLHN(70)	RA(74), CCLP(74)	RA(72), HPI(72)
NS	CCLP(65), AA(63)	AA(70), CCLP(68)	AA(61), PA(61)	AA(70), CCLP(68)	AA(65), RA(65)	PA(72), LLHN(72)
PB	HPI(68), PA(64)	RA(68), PA(66)	LLHN(61), RA(59)	AA(68), RA(68)	RA(76), CCLP(75)	RA(71), AA(71)
Power	HPI(63), LLHN(63)	PA(63), HPI(62)	AA(67), RA(67)	PA(63), RA(62)	PA(54), AA(51)	PA(54), LLHN(54)
Router	PA(52), LLHN(47)	PA(66), LLHN(51)	CCLP(65), RA(65)	CCLP(69), AA(68)	PA(57), CCLP(54)	PA(52), HPI(52)
USAir	AA(78), CCLP(78)	LLHN(90), HPI(88)	CCLP(77), AA(75)	RA(90), LLHN(90)	RA(81), AA(81)	HPI(69), RA(69)
Yeast	CCLP(75), PA(74)	CCLP(70), AA(69)	CCLP(75), AA(71)	CCLP(70), AA(69)	AA(72), RAI(71)	RA(69), HPI(69)
FB15K	RA(32), HPI(31)	LLHN(30), HPI(28)	HPI(28), LLHN(27)	HPI(26), AA(24)	HPI(42), CCLP(39)	HPI(38), LLHN(36)
WN18	PA(44), LLHN(42)	PA(40), HPI(32)	PA(28), AA(26)	PA(36), CCLP(31)	–	–
YAGO 3-10	PA(34), AA(26)	PA(44), AA(24)	PA(38), CCLP(32)	PA(42), RA(34)	–	–

In order to further explore their connections, we compute the percentage of agreements in link existence between the embedding-based and the similarity-based approaches. Table 5 shows the top-2 ranked similarity-based approaches when they are ranked in decreasing order of their percentage of agreements on each graph for each embedding-based approach. Overall, embedding-based approaches show higher percentages of agreements to similarity-based approaches in small/medium graphs than in large graphs. Considering all graphs, HPI, PA and LLHN are three frequent heuristics which have higher agreement to WLNM and SEAL approaches. On the other hand, AA, RA and CCLP show frequent agreements with GAT. Between auto-encoder based approaches, LVGAE has higher percentage of agreement to hop-1 neighbourhood based PA, LLHN, HPI heuristics as LVGAE aggregates hop-1 neighbourhood information of a node during node encoding. For VGAE, CCLP, AA, RA are three frequent heuristics which have higher percentage of agreement. These agreements align to the previous discussion on the nature of learned heuristics in embedding-based methods. However, low agreement percentage values (in Table 5) but high precision scores (in Table 3) for embedding-based approaches in many graphs like FB15K, Ecoli, NS suggest the existence of other learned heuristics that are not included in this study.

The performance of the studied methods was also assessed in terms of average computational time (data will be made available on request). As expected, similarity-based approaches are faster as they don't require training. As for embedding-based approaches, Node2Vec requires the smallest time as it does not use deep NN like the other embedding-based methods. The computational time of auto-encoder based VGAE and LVGAE approaches are lower than NN-based WLNM, SEAL, GAT approaches as it shallow or no neural network. The computational time of SEAL is the best as it utilizes the structural and explicit features like WLNM and GAT along with latent features like Node2Vec. We also noticed that the computational time of embedding-based methods grows with the size of datasets by more amount than the similarity-based methods.

5 FUTURE RESEARCH DIRECTIONS

Although there exists numerous approaches for link prediction in graphs, it is remarkable that existing approaches are not good enough to offer satisfying solutions for any graphs from any domains. We list some issues in this section for future research to explore.

Scalability: Many of the real-world graphs are huge in size. The scalability of embedding-based approaches is achieved by sacrificing the completeness of graphs. In addition, auto-encoder based approaches often failed for large scale graphs due to high memory requirement. A valuable future research could be developing a link prediction approach in a distributed and parallel environment to make a good trade-off between scalability and graph integrity.

Interpretability: GNN-based link prediction approaches suffer from well-known 'black-boxes' problem. It is important to provide trusted explanation of these approaches when applying on real-world applications. Developing an interpretable GNN-based link prediction approach could be an important future work.

Transferability. Embedding-based link prediction approaches require high-quality labeled data and it is a costly task to obtain the labeled data. To overcome this problem, the transfer potentiality of trained model has been investigated and applied to the domains of image and language processing with performance guarantees [41], [42]. Recently, researchers also study the transferability or use of pretrained model in graph domain [43]–[45]. However, the task still require further researches to improve the effectiveness of existing embedding-based models on learning structural or feature information, etc.

Heterogenity: Most of the current embedding-based link prediction approaches are developed for homogeneous graphs. However, most of the real-world graphs are heterogeneous. A valuable future research could be designing a high-performing embedding-based link prediction approach for heterogeneous.

Robustness: Embedding-based link prediction approaches are vulnerable to adversarial attacks [46]. The attacks consider not only node/edge attributes, but also structural features of graphs. The attacks remarkably affect the prediction performance. Developing a robust graph embedding-based link prediction approach, which could effectively defend adversarial attacks, is an import future direction.

6 CONCLUSION

In this paper, we study several link prediction approaches, looking for their performances and connections. We focused on two categories of methods: similarity-based methods and embedding-based learning methods. The studied approaches were evaluated on ten graph datasets with different properties from various domains. The precision of similarity-based approaches was computed in two different ways to highlight the difficulty of tuning the threshold for deciding the link existence based on the similarity score. The experimental results show the expected superiority of embedding-based approaches. Still, each of the similarity-based approaches is competitive on graphs with specific properties. The possible links between the handcrafted similarity-based approaches and current embedding-based approaches were explored using (i) prediction performance comparison to get an idea about the learned heuristics and (ii) agreement percentage on the diverse graphs. Our

observations constitute a modest contribution to the 'black box' limitation of GNN-based methods.
Finally, we suggest few research directions of link prediction problem , including robustness, scalability, interpretability, transferability, heterogenity, and robustness.

References

[1] Z. Xu, C. Pu, and J. Yang, "Link prediction based on path entropy," *Physica A: Statistical Mechanics and its Applications*, vol. 456, pp. 294–301, 2016.

[2] Z. Shen, W.-X. Wang, Y. Fan, Z. Di, and Y.-C. Lai, "Reconstructing propagation networks with natural diversity and identifying hidden sources," *Nature communications*, vol. 5, no. 1, pp. 1–10, 2014.

[3] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[4] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[5] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2015.

[6] I. A. Kovács, K. Luck, K. Spirohn, Y. Wang, C. Pollis, S. Schlabach, W. Bian, D.-K. Kim, N. Kishore, T. Hao, *et al.*, "Network-based prediction of protein interactions," *Nature communications*, vol. 10, no. 1, pp. 1–8, 2019.

[7] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[8] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[9] V. Martı´nez, F. Berzal, and J.-C. Cubero, "A survey of link prediction in complex networks," *ACM computing surveys (CSUR)*, vol. 49, no. 4, pp. 1–33, 2016.

[10] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[11] H. Cai, V. W. Zheng, and K. C. C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[12] M. K. Islam, S. Aridhi, and M. Smail-Tabbone, "Appraisal study of similarity-based and embedding-based link prediction methods on graphs," in *Proceedings of the 10th International Conference on Data Mining & Knowledge Management Process*, 2021, pp. 81–92.

[13] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *The European Physical Journal B*, vol. 71, no. 4, pp. 623–630, 2009.

[14] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[15] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, "Hierarchical organization of modularity in metabolic networks," *science*, vol. 297, no. 5586, pp. 1551–1555, 2002.

[16] E. A. Leicht, P. Holme, and M. E. Newman, "Vertex similarity in networks," *Physical Review E*, vol. 73, no. 2, p. 026 120, 2006.

[17] Z. Wu, Y. Lin, J. Wang, and S. Gregory, "Link prediction with node clustering coefficient," *Physica A: Statistical Mechanics and its Applications*, vol. 452, pp. 1–8, 2016.

[18] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length, and helmholtz free energy," *Advances in neural information processing systems*, vol. 6, pp. 3–10, 1994.

[19] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[20] B. Weisfeiler and A. A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.

[21] M. Zhang and Y. Chen, "Weisfeiler-lehman neural machine for link prediction," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 575–583.

[22] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5165–5175.

[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of International Conference on Learning Representations*, 2016, pp. 4700–4708.

[24] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[25] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *NIPS Bayesian Deep Learning Workshop*, 2016.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[27] G. Salha, R. Hennequin, and M. Vazirgiannis, "Keep it simple: Graph autoencoders without graph convolutional networks," 2019.

[28] H. Salgado, A. S. Zavaleta, S. G. Castro, D. M. Zárate, E. D. Peredo, F. S. Solano, E. P. Rueda, C. B. Martı´nez, and J. C. Vides, "Regulondb (version 3.2): Transcriptional regulation and operon organization in escherichia coli k-12," *Nucleic acids research*, vol. 29, no. 1, pp. 72–74, 2001.

[29] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[30] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, p. 036 104, 2006.

[31] R. Ackland *et al.*, "Mapping the us political blogosphere: Are conservative bloggers more prominent?" In *BlogTalk Downunder 2005 Conference, Sydney*, 2005.

[32] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[33] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[34] M. S. Handcock, D. R. Hunter, C. T. Butts, S. M. Goodreau, and M. Morris, "Statnet: An r package for the statistical modeling of social networks," *Web page http://www. csde. washington. edu/statnet*, 2003.

[35] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," *Machine Learning*, vol. 94, no. 2, pp. 233–259, 2014.

[36] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *7th Biennial Conference on Innovative Data Systems Research*, 2013.

[37] C. Von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork, "Comparative assessment of large-scale datasets of protein- protein interactions," *Nature*, vol. 417, no. 6887, pp. 399–403, 2002.

[38] L. Pan, T. Zhou, L. Lü, and C. K. Hu, "Predicting missing links and identifying spurious links via likelihood analysis," *Scientific Reports*, vol. 6, no. 1, pp. 1–10, 2016.

[39] S. Tsugawa and H. Ohsaki, "Effectiveness of link prediction for face-to-face behavioral networks," *Plos one*, vol. 8, no. 12, e81727, 2013.

[40] Y. Yang, R. N. Lichtenwalter, and N. V. Chawla, "Evaluating link prediction methods," *Knowledge and Information Systems*, vol. 45, no. 3, pp. 751–782, 2015.

[41] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith, "Linguistic knowledge and transferability of contextual representations," *arXiv preprint arXiv:1903.08855*, 2019.

[42] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability for a generic convnet representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1790–1802, 2015.

[43] R. Levie, W. Huang, L. Bucci, M. M. Bronstein, and G. Kutyniok, "Transferability of spectral graph convolutional neural networks," *arXiv preprint arXiv:1907.12972*, 2019.

[44] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1150–1160.

[45] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," *arXiv preprint arXiv:1905.12265*, 2019.

[46] L. Sun, Y. Dou, C. Yang, J. Wang, P. S. Yu, L. He, and B. Li, "Adversarial attack and defense on graph data: A survey," *arXiv preprint arXiv:1812.10528*, 2018.