Feature Engineering on Link Prediction

Astrit Tola
UTD NetID:axt200006
astrit.tola@utdallas.edu

Dona Hasini Gammune
UTD NetID:dvg190000
DonaHasini.Gammune@UTDallas.edu

Brighton Nuwagira
UTD NetID:bxb210001
brighton.nuwagira@utdallas.edu

Mahesh Ranpati Dewage
UTD NetID:mkr200000
manjulamahesh.ranpatidewage@utdallas.edu

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Link prediction is a technique that aims to predict the probability of a connection between two nodes in a graph using the information from the observed graph data. This method finds widespread use in diverse fields. By studying the patterns of existing connections in a network, link prediction algorithms provide valuable insights into possible future links between nodes. This is useful in understanding complex systems better and has practical uses in various areas. To address this task, graph machine learning methods are commonly employed.

In social network analysis, link prediction techniques help identify future connections between individuals, like friendships and collaborations. For example, in online social networks, predicting future connections can improve personalized recommendations and targeted marketing.

Link prediction is also essential in security domains. It can uncover hidden patterns and connections that indicate malicious activities or criminals. Identifying potential links between individuals in security networks helps dismantle criminal networks and protect communities from threats.

In biological and bioinformatics, link prediction algorithms infer potential interactions between proteins in biological networks, which helps us understand cellular processes and diseases better. Similarly, predicting brain network connections in neuroscience enhances our understanding of brain disorders and cognitive functions.

Link prediction techniques are even applied in meteorology to improve weather forecasts. By analyzing historical weather data, we can identify hidden patterns and enhance our ability to prepare for extreme weather events.

Predicting future citation links between academic papers is a valuable task that enhances our understanding of scholarly communication, research trends, and the evolution of scientific knowledge. Forecasting future citation links between academic papers involves using computational methods to predict which papers are likely to cite each other. Academic papers often reference or cite others to provide context, support claims, or acknowledge prior research. As a result, the citation network formed by these references forms a valuable source of information about the relationships and influence between different papers.

The link prediction problem focuses on predicting connections that have yet to occur but may likely form in the future. The idea is to leverage the existing citation network and patterns of past citations to infer potential future connections.

The applications of predicting future citation links are significant in academia and research. It can help researchers and scholars identify potential, influential papers or emerging trends. It can also aid in recommending related works during literature searches and improve the efficiency of information retrieval. Moreover, understanding citation dynamics for academic publishers and journals can help improve publication strategies and impact factor predictions. Various techniques from network analysis and machine learning, including graph-based similarity, random walk-based methods, latent space models, deep learning models, etc., can be employed for this task.

In this project, we aim to develop a robust model, incorporating a neural network component, that can accurately predict future citation links between academic papers. We employed the publicly available CiteSeer Benchmark data set, comprising 3312 scientific publications categorized into six distinct fields. The data set includes a citation network with 4732 links representing the citations between publications. We employ the Citesser benchmark dataset and the 'networkx' package for graph operations. The process involves introducing equivalent negative edges alongside existing positive edges in the Citeseer graph. We extract a feature set from the original graph using five feature extraction functions, namely Shortest path, Jaccard Index, Salton Index, Sorensen Index, and Common Field, applied to a selected training set. We employ two methods to predict link existence within the test set: our custom implementation of a Neural Network (NN) and the XGboost algorithm. By applying these steps, we aim to gain valuable insights into citation network dynamics and predict future citation links between academic papers.

II. Background Work

"Semi-Supervised Classification with Graph Convolutional Networks" by Thomas Kipf and Max Welling (2016) may fo-

cus on semi-supervised classification. However, it has laid the foundation for graph convolutional networks in link prediction tasks [11].

"Representation Learning on Graphs: Methods and Applications" by William L. Hamilton, Rex Ying, and Jure Leskovec (2017) provides a comprehensive survey of representation learning methods on graphs, including those applicable to link prediction using neural networks [13].

"Heterogeneous Graph Attention Network" by Chuxu Zhang et al. (2019) introduces the concept of using attention mechanisms for link prediction in heterogeneous information networks through the HAN model [14].

"Representation Learning for Attributed Multiplex Heterogeneous Network" by Hongchang Gao et al. (2018) presents a method for representation learning in attributed multiplex heterogeneous networks, proving useful for link prediction tasks within such networks [12].

Collectively, these papers demonstrate how neural networks and relevant mechanisms can effectively learn graph representations and patterns, thus significantly advancing link prediction capabilities across various network types.

III. Problem Definition

Let $G = (V, E)$ be an un-directed graph of N nodes so that $V = \{v_1, v_2, ..., v_N\}$ is the set of nodes and $E \subseteq V \times V$ is the set of observed links. We denote the adjacency matrix as $A \in \{0, 1\}^{N \times N}$, where $A_{i,j} = 1$ indicates that there is an edge between nodes v_i and v_j. We denote the feature matrix of nodes as $X \in \mathbb{R}^{N \times F}$, where F indicates the number of node features and x_i represents the feature vector of node v_i [8], [9].

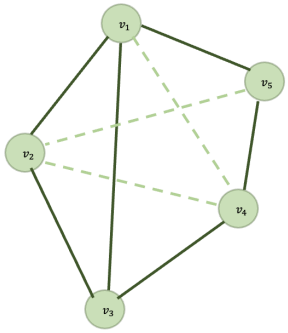

Fig. 1. The Link Prediction Graph (v_1v_4, v_2v_4, and v_2v_5 are missing links)

IV. Theory And Conceptual Study

A. Deep neural network

A deep neural network is a collection of different layers. The input layer captures and passes the input signals to the next layer. The hidden layers perform nonlinear transformations of the inputs and pass the result to the final layer and the output layer, which delivers the final results.

A model with an input layer, k number of hidden layers, and an output layer can be written as

$$X \to \boldsymbol{h}^{(1)}(X) \to \cdots \to \boldsymbol{h}^{(k)}\{\boldsymbol{h}^{(k-1)}\} \to a\{\boldsymbol{h}^{(k)}\} \to \hat{\boldsymbol{Y}}$$

implying the target f is assumed have the form

$$f(X;\theta) = a\{\boldsymbol{h}^{k}\{\boldsymbol{h}^{(k-1)}\{\ldots\{\boldsymbol{h}^{(1)}(X)\}\ldots\}\}\}$$

with specified activation function a. Parameter θ consists of biases and weights. Figure 2 represents a simple neural network with one hidden layer.

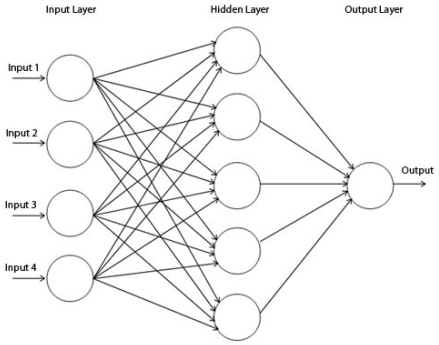

Fig. 2. A simple neural network with one hidden layer

We built a deep neural network using the following steps:

- Choosing activation function
- Forward propagation
- Choosing loss (cost) function
- Backward propagation
- Parameter tuning

1) Activation function: The activation function takes the sum of the weighted inputs as input and computes the output of a neuron. The sigmoid function is one of the popular activation functions used in neural networks. It can transform any real number to a value between 0 and 1. This property is beneficial as it enables the neural network to learn and capture non-linear relationships in the data. The sigmoid function is mathematically represented as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

derivative of sigmoid would be

$$\frac{\partial \sigma}{\partial x} = \sigma(x)(1 - \sigma(x))$$

2) Forward Propagation: During forward propagation, information flows in a forward direction through the network. The input information is first propagated through the layers, reaching the hidden units at each layer, where computations and transformations occur. Finally, the process culminates in producing the predicted output $\hat{\boldsymbol{y}}$, representing the network's prediction for a given input.

The forward propagation equations for a neural network with one hidden layer can be formulated as:

$$\begin{aligned} \boldsymbol{Z}_1 &= \boldsymbol{W}_1^T \boldsymbol{X} + \boldsymbol{b}_1 \\ \boldsymbol{A}_1 &= \sigma(\boldsymbol{Z}_1) \\ \boldsymbol{Z}_{out} &= \boldsymbol{W}_{out}^T \boldsymbol{A}_1 + \boldsymbol{b}_{out} \\ \boldsymbol{O} &= \sigma(\boldsymbol{Z}_{out}) \end{aligned}$$

where, $\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{b}, a, \boldsymbol{O}$ represents corresponding input, weights, biases, activation function, and output.

3) Calculating loss: Next, we compare the result from the forward propagation with the actual output. The goal is to minimize the difference between predicted and actual output. This difference is quantified using a loss function (cost function) that measures the dissimilarity between predicted and actual output. Commonly used loss functions in different problems include mean squared error (MSE) for regression tasks and cross-entropy loss for classification tasks.

4) Backward propagation: Now, we minimize the loss by updating weights and biases while traveling back. This process is known as "Backward Propagation". We employ a standard algorithm called "Gradient Descent", which iteratively adjusts the weights and biases in the opposite direction of the gradients, gradually reducing the loss. We update weights using the rule

$$\boldsymbol{W} \rightarrow \boldsymbol{W} - \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}}$$

Here we consider two cases,

(a) *Updating weights and bias at output layer*: Our goal is to update the weights ($\boldsymbol{W}_{out}$) and bias ($\boldsymbol{b}_{out}$) of the output layer. Let's take sigmoid as our activation and MSE as our loss function (figure 3). The values of $\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}_{out}}$ and $\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}_{out}}$ can be calculated as

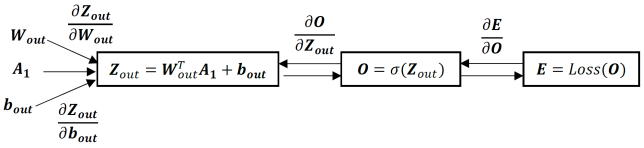

Fig. 3. updating weights and bias at output layer

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}_{out}} = \frac{\partial \boldsymbol{Z}_{out}}{\partial \boldsymbol{W}_{out}} \times \left[\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{Z}_{out}} \cdot \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}} \right]^T$$

and

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}_{out}} = \frac{\partial \boldsymbol{b}_{out}}{\partial \boldsymbol{W}_{out}} \times \left[\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{Z}_{out}} \cdot \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}} \right]^T$$

where,

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}} = -(\boldsymbol{Y} - \boldsymbol{O}), \quad \frac{\partial \boldsymbol{O}}{\partial \boldsymbol{Z}_{out}} = \boldsymbol{O}(1 - \boldsymbol{O})$$
$$\frac{\partial \boldsymbol{Z}_{out}}{\partial \boldsymbol{W}_{out}} = \boldsymbol{A}_1, \quad \frac{\partial \boldsymbol{Z}_{out}}{\partial \boldsymbol{b}_{out}} = \boldsymbol{1}$$

$\times$ represent dot product and $\cdot$ represent element wise multiplication.

Then we can update weights using

$$\boldsymbol{W}_{out} \rightarrow \boldsymbol{W}_{out} - \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}_{out}}, \quad \boldsymbol{b}_{out} \rightarrow \boldsymbol{b}_{out} - \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}_{out}}$$

(b) *Updating weights and bias at hidden layers*: Our goal is to update the weight($\boldsymbol{W}_{out}$) and bias($\boldsymbol{b}_{out}$) of the hidden layer (figure 4). The values of $\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}_1}$ and $\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}_1}$ can be calculated as

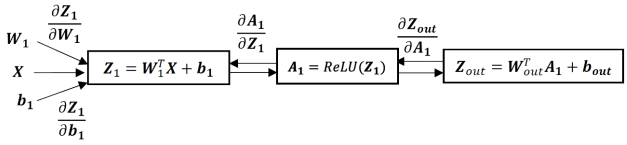

Fig. 4. updating weights and bias at output layer

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}_1} = \frac{\partial \boldsymbol{Z}_1}{\partial \boldsymbol{W}_1} \times \left[\frac{\partial \boldsymbol{A}_1}{\partial \boldsymbol{Z}_1} \cdot \frac{\partial \boldsymbol{Z}_{out}}{\partial \boldsymbol{A}_1} \times \left[\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{Z}_{out}} \cdot \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}} \right] \right]^T$$

and

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}_1} = \frac{\partial \boldsymbol{Z}_1}{\partial \boldsymbol{b}_1} \times \left[\frac{\partial \boldsymbol{A}_1}{\partial \boldsymbol{Z}_1} \cdot \frac{\partial \boldsymbol{Z}_{out}}{\partial \boldsymbol{A}_1} \times \left[\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{Z}_{out}} \cdot \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}} \right] \right]^T$$

where,

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{O}} = -(\boldsymbol{Y} - \boldsymbol{O}), \quad \frac{\partial \boldsymbol{O}}{\partial \boldsymbol{Z}_{out}} = \boldsymbol{O}(1 - \boldsymbol{O}),$$
$$\frac{\partial \boldsymbol{Z}_{out}}{\partial \boldsymbol{A}_1} = \boldsymbol{W}_{out}, \quad \frac{\partial \boldsymbol{A}_1}{\partial \boldsymbol{Z}_1} = \boldsymbol{A}_1(1 - \boldsymbol{A}_1),$$
$$\frac{\partial \boldsymbol{Z}_1}{\partial \boldsymbol{W}_1} = \boldsymbol{X}, \quad \frac{\partial \boldsymbol{Z}_1}{\partial \boldsymbol{b}_1} = \boldsymbol{1}.$$

we can update weights using

$$\boldsymbol{W}_1 \rightarrow \boldsymbol{W}_1 - \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{W}_1}, \quad \boldsymbol{b}_1 \rightarrow \boldsymbol{b}_1 - \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{b}_1}$$

B. XGBoost

Extreme Gradient Boosting (XGBoost) is a widely acclaimed and potent machine learning technique renowned for its efficiency, speed, and remarkable predictive capabilities. Falling into the ensemble learning category, XGBoost harnesses the collective wisdom of multiple base models, usually decision trees, to generate a final accurate and robust prediction. However, owing to its potency, optimizing XGBoost through hyperparameter tuning can be intricate and computationally demanding, particularly for deep and intricate models. As with any machine learning algorithm, the efficacy of XGBoost heavily relies on the quality and relevance of the features utilized and the careful calibration of hyperparameters tailored to the specific problem at hand.

C. Local similarity indices

(a) Shortest path distance.
The possibility of friends of a friend becoming friends suggests that the path distance between nodes in a social network can influence link formation. Smaller path distances imply a higher likelihood of link creation. However, it is essential to consider the small-world phenomenon [1], as only a few vertices separate most node pairs in social networks. Consequently, this feature may only sometimes perform well in link prediction

tasks. In a study by Hasan et al. [2] focusing on link prediction in a biological co-authorship network, this feature ranked, on average, at 4 out of 9 features used. A similar finding of limited effectiveness for this feature was also noted in another study. [3].

(b) Jaccard Coefficient

An alternative to the non-normalized common neighbors metric is the Jaccard Coefficient, which offers normalization by considering the ratio of the size of common neighbors to the total number of unique neighbors between two nodes. This normalization enhances the comparability of similarity scores across different node pairs. Note that:

$$\text{Jaccard-coefficient}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \quad (1)$$

The Jaccard Coefficient quantifies the likelihood of selecting a common neighbor between two vertices, x and y, if neighbors are randomly chosen from the combined neighbor sets of x and y. Thus, a higher score is obtained when there are more common neighbors.

(c) Sorensen Index

The Sorensen Index (in equation 2), proposed by Thorvald Sorensen in 1948, is a similarity index primarily used for analyzing ecological data samples [5]. It bears a close resemblance to the Jaccard.

$$\text{Sorensen Index } (x, y) = \frac{2|\Gamma(x) \cap \Gamma(y)|}{k_x + k_y} \quad (2)$$

McCune et al. [4] demonstrated that the Sorensen Index exhibits greater robustness against outliers compared to the Jaccard index.

(d) Salton Index

To compute document similarities, the Salton index, also known as Cosine similarity, is employed [6]. This similarity index measures the similarity between two records by calculating their Cosine similarity. The Cosine similarity metric focuses solely on the orientation of the vectors and disregards their magnitudes.

$$\text{Salton Index } (x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{\sqrt{k_x.k_y}} \quad (3)$$

V. Results

TABLE I
LINK PREDICTION PERFORMANCES MEASURED BY HITS@20 (LEFT) AND AUC (RIGHT)

By Hits@20	CITESEER
Node2Vec	47.78±1.72
MVGRL	14.07±0.79
VGAE	44.04±4.86
SEAL	40.90±3.68
LGLP	57.43±3.71
GCN	55.56±1.32
GSAGE	53.67±2.94
JKNet	55.60±2.17
Our model	

By AUC	CITESEER
Node2Vec	80.00±0.68
MVGRL	61.20±0.55
VGAE	85.35±0.60
SEAL	85.82±0.44
LGLP	89.41±0.13
GCN	71.47±1.40
GSAGE	87.38±1.39
JKNet	88.58±1.78
Our model	

VI. Discussion and Conclusion

The citation network formed by academic papers is a fundamental aspect of scholarly communication and knowledge dissemination. In academic research, papers often cite other works to acknowledge the existing knowledge, provide evidence for their claims, and establish credibility. The structure of this citation network provides valuable insights into the relationships between papers and their influence on each other. Predicting future citation links offers several advantages for researchers. It can help to identify potentially influential papers, emerging research trends, and promising areas of study. Researchers can gain insights into the dynamics of knowledge dissemination and scholarly impact by discovering connections between papers likely to be cited in the future [7].

References

[1] Al Hasan, Mohammad and Chaoji, Vineet and Salem, Saeed and Zaki, Mohammed, "Link prediction using supervised learning" Applied Mechanics and Materials, vol.30, 2015

[2] Li, Min and Ou, Rui Qiu and Song, Yan Ling, "Principal-Agent Activities on Small World Network" Applied Mechanics and Materials, vol.727, pp.798–805, 2006.

[3] Liben-Nowell, David and Kleinberg, Jon, "The link prediction problem for social networks" Applied Mechanics and Materials, vol.727, pp.556–559, 2003.

[4] McCune, Bruce and Grace, James, "Analysis of ecological communities", 2002

[5] Sørensen, Thorvald Julius, "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" I kommission hos E. Munksgaard, 1948.

[6] Salton, McGill, "Introduction to Modern Information Retrieval", McGraw-Hill, Inc., New York, NY, USA, 1986.

[7] Baskadia, "The Importance of in Graph Theory by Affiliate Marketing For Beginners". https://baskadia.com/post/4mrn.

[8] Zhao, Tong, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. "Learning from counterfactual links for link prediction." In International Conference on Machine Learning, pp. 26911-26926. PMLR, 2022.

[9] Zhao, Tong. Learning to Augment Data in Graphs. University of Notre Dame, 2022.

[10] Choi, Jiho, Taewook Ko, Younhyuk Choi, Hyungho Byun, and Kim Chong-kwon. "Dynamic Graph Convolutional Networks with Attention Mechanism for Rumor Detection on Social Media." PLoS One 16 (8): e0256039, 2021.

[11] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907, 2016.

[12] Cen, Yukuo, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. "Representation learning for attributed multiplex heterogeneous network." In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 1358-1368. 2019.

[13] Hamilton, William L., Rex Ying, and Jure Leskovec. "Representation learning on graphs: Methods and applications." arXiv preprint arXiv:1709.05584 (2017).

[14] Zhang, Chuxu, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. "Heterogeneous graph neural network." In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 793-803. 2019.