# Machine Learning-Based Approaches to Link Prediction in Complex Networks

Project Proposal

Introduction to Network Analysis 2019/20, Faculty of Computer and Information Science, University of Ljubljana

Jernej Vivod*, Grega Dvoršak†
vivod.jernej@gmail.com*
grega.dvorsak@gmail.com†

## I. Introduction

Link prediction is an important and popular topic in the field of network science and has been the subject of much research in recent years. Methods developed to solve the task of link prediction attempt to deduce future links in the network using the information about the network's current state. Link prediction can also be used to find missing links that may occur due to an imperfect data collection process and for aiding in complex network reconstruction. In this study, we evolve and evaluate a machine learning-based approach to link prediction. We categorize the features and estimate their relative discriminative importance using feature ranking algorithms such as Relief derivatives and logistic regression. We also evaluate various node embedding techniques as a means of obtaining useful representations for our task. We evaluate the use of common machine learning algorithms as well as the so-called Feature stacking method that is especially useful in the context of high dimensional data. Finally, we briefly explore some recently developed deep learning-based approaches to the link prediction problem.

## II. Related Work

An early paper tackling the problem of link prediction using supervised machine learning methods is the 2006 study by Hasan et al. [1] where the authors evaluate the performance of typical machine learning algorithms using general features as well as specific ones characteristic of co-authorship networks. We intend to vastly expand the set of features to also include community-based features, existing indices developed for this specified task as well as features generated using node embedding techniques. We also intend to evaluate additional prediction algorithms from the state of the art deep learning domain and test our approaches on a wider set of networks from different domains. Another notable example is the 2010 study by Benchettara et al. [2] which focuses specifically on bipartite social networks using mainly topological features in tandem with the Ada boost method. The feature set is augmented with the Jaccard coefficient, Adamic Adar index, the preferential attachment index, and the PageRank scores. Using node embeddings to predict links is a popular research topic but few studies focus on explicitly evaluating their use in the context of inductive machine learning algorithms. Deep learning-based methods have shown great potential and are explored in numerous recent studies [3][4][5][6].

## III. Methods

In this section, we describe the methodology of our study. We briefly describe the evaluation process and the developed evaluation framework. We propose methods of feature engineering to describe node pairs and briefly introduce node embedding techniques that can yield useful feature-like representations of nodes. We describe the methods used to infer probabilities of links between nodes in a network, starting with classical approaches and useful baselines which represent an important sanity check for analyzing the performance of any other method. We introduce the so-called feature stacking ensemble method that is well suited for high-dimensional data. We describe methods of model parameter optimizations suitable for our specific scenario as well as several methods of inferring the relative importance of features which is a crucial step in making the results interpretable. finally, we briefly state some recent deep learning-based approaches to the problem of link prediction.

### A. The Evaluation Process and Framework

The canonical approach to evaluating prediction and classification methods is to split the dataset into training and test sets, where the proportion of samples in each set depends on the amount of data available. The training data is used to induce a model or hypothesis that is then used to make predictions on the withheld test set. Since we know the true values of the information we want to predict we can make statistical inferences and evaluate the performance of our method. Optionally a third, separate part of the data is withheld to serve as the so-called validation set used to optimize the hyperparameters of our method.

We construct our training and test sets from a given network by sampling node pairs in such a way as to minimize the information leakage between the training and test sets. We begin by randomly sampling a set number of node pairs that are not connected by a link in the original network. The number of samples is determined as a specified fraction of the number of links in the network. This forms our negative test set. Next, we sample a set number of connected node pairs using the same sample size to produce our positive test set. We then collect all connected node pairs not found in the test sample and designate them as our positive training set. Finally, we take the size of the positive training set as our sample size and randomly sample remaining nodes in the network that are not linked and are not yet present in the negative test set and designate them as our negative training set.

When constructing features for each node pair we again try to minimize cross-set data leakage as much as possible. We map node pairs in the positive training and test sets to their feature vector representations by first removing the link between the two nodes to simulate the approximate local network state prior to the link. There are several nuances and pitfalls to this approach which can be improved considerably if the network stores any kind of temporal information regarding the occurrence of links.

We implemented a modular and scalable framework evaluation method that allows the user to easily select considered features, prediction methods, and other evaluation parameters. The code is organized into packages and modules based on their role of classification, evaluation, feature extraction, data parsing, and results processing. The *evaluate.py* script in the *src* folder implements the main evaluation functionality. The script accepts a list of test networks for which to perform the evaluations as well as the prediction method being evaluated. The script runs the evaluation procedure specified number of times and saves the averaged classification statistics (classification accuracy, precision, recall, f-score, support) to the *results.txt* file in the *results* folder in the form of a markdown table along with associated metadata such as time, the method used, and the name of the network on which the method was evaluated. We also plot the confusion matrices, compute the AUC score, and plot the ROC curve.

### B. Feature Extraction

We use node pairs as entities we wish to describe with informative features. Insights from graph theory and the rapid advancement of network science provide us with a plethora of ways to characterize nodes and how they are embedded in the network. We utilize information about the local network topology, use any node or edge labels and we also compute some well-known indices used in link prediction. We also compute the community structure of the network and characterize the position of the given node in its community. We aim to explore many other possibilities in the final implementation. In a later section, we describe how we can try to determine the relative importance of each feature using Relief-based feature selection and ranking methods and logistic regression. This is important for making the predictions interpretable and for avoiding the issues associated with the so-called curse of dimensionality.

We also evaluate the use of node embeddings as a means of mapping the nodes to real-valued vectors that seek to best preserve their relationships, similarly to the better-known word embedding method used in natural language processing.

### C. Prediction Methods

In this subsection, we describe our approaches to solving the prediction problem after we have computed a suitable representation of our data.

#### 1) Classical Approaches and Baseline Models

The research in the field of machine learning has produced a myriad of methods that can be used to make predictions in a supervised learning setting. We evaluate the performance of important classification algorithms such as Support vector machines, Random forests, logistic regression, and an ensemble method called Feature stacking which we describe in greater detail below. It is important to critically compare any results obtained by such sophisticated methods to the outputs of baseline models such as the so-called majority classifier, which always predicts the most common label found in training data with maximal certainty as well as the random-guess or uniform classifier which predicts the labels uniformly at random. To be useful, any implemented method should be able to outperform these trivial baselines.

#### 2) The feature stacking classifier

An interesting algorithm well suited for classifying high-dimensional data is the so-called Feature stacking approach which combines several classifiers each trained on a feature subset and a final classifier that takes as its input the output of these classifiers. The authors of the paper [7] describing the Feature stacking approach suggest using logistic regression as the feature classification method since it makes the method resemble a neural network structure. We implemented the feature stacking method as a classifier conforming to the well-known Scikit-Learn [8] machine learning framework interface.

During training, the training data is converted to new features consisting of logistic regression outputs for each feature subset. This is achieved using k-fold cross-validation. Next, logistic regression is fitted to the entire training data feature subsets and is used to encode the test data. A final meta-classifier is fitted to the training data encoded using logistic regression. Test data is first encoded using a trained logistic regression model and finally classified with the meta-classifier. The features stacking method is contrasted with the more typical feature concatenation method on the diagram shown in figure 1.

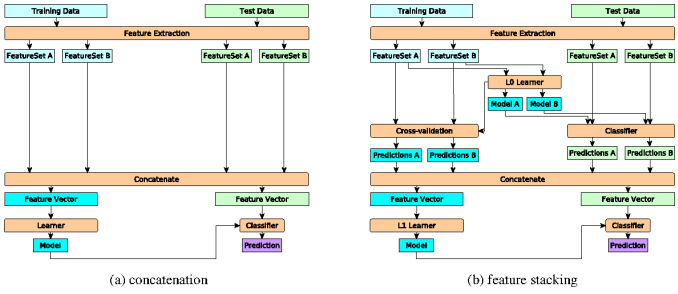


Figure 1. Diagram of the Feature stacking method. Taken from [7].

#### 3) Hyperparameter optimization

Hyperparameter optimization is an important part of model selection in which we try to find good values of model parameters that must be set before the learning process begins. The most common approach is to perform a so-called grid search by evaluating all combinations of parameters from a pool of possible values. This approach is computationally extremely expensive which is why often only a small subset of the data is used. Many novel approaches have been developed to tackle this issue. A notable example is the use of genetic algorithms. The *TPOT* tool [9] [10] implements this idea and can be used to completely automate the construction of an effective machine learning pipeline for a specific problem.

### D. Feature Ranking and Feature Selection

In most scenarios, not all features are equally important in discriminating between the labels, or, in our case, the presence or absence of nodes. An over-abundance of features, many of which acting as noise, can lead to problems associated with the so-called curse of dimensionality. Important features can get obscured and conventional distance metrics can become notably unreliable - two feature vectors that are close with respect to a very discriminative feature can appear to be very far apart when all available features are considered by a typical distance

metric. Observing which features are important in predicting the occurrence of a link is also important for interpreting the predictions which can lead to important inferences about the nature of the observed network.

We employ typical feature weighting and ranking methods as well as logistic regression to estimate the discriminatory powers of features. We also use Relief-based derivatives [11] which are notably sensitive to feature interactions and are readily available through the skrelief library [12].

## References


[1] M. Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link prediction using supervised learning," 01 2006.

[2] N. Benchettara, R. Kanawati, and C. Rouveirol, "Supervised machine learning applied to link prediction in bipartite social networks," 08 2010, pp. 326–330.

[3] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang, *A Deep Learning Approach to Link Prediction in Dynamic Networks*, 04 2014, pp. 289–297.

[4] W. Gu, F. Gao, X. Lou, and J. Zhang, "Link prediction via graph attention network," 2019.

[5] M. Lim, A. Abdullah, N. Zaman, and M. Supramaniam, "Hidden link prediction in criminal networks using the deep reinforcement learning technique," *Computers*, vol. 8, p. 8, 01 2019.

[6] W. Wang, L. Wu, Y. Huang, H. Wang, and R. Zhu, "Link prediction based on deep convolutional neural network," *Information*, vol. 10, p. 172, 05 2019.

[7] M. Lui, "Feature stacking for sentence classification in evidence-based medicine," in *Proceedings of the Australasian Language Technology Association Workshop 2012*, Dunedin, New Zealand, Dec. 2012, pp. 134–138. [Online]. Available: https://www.aclweb.org/anthology/U12-1019

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] T. T. Le, W. Fu, and J. H. Moore, "Scaling tree-based automated machine learning to biomedical big data with a feature set selector," *Bioinformatics*, vol. 36, no. 1, pp. 250–256, 2020.

[10] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, ser. GECCO '16. New York, NY, USA: ACM, 2016, pp. 485–492. [Online]. Available: http://doi.acm.org/10.1145/2908812.2908918

[11] J. Vivod, "Ocenjevanje atributov s posplošitvami algoritma relief," Undergraduate Thesis, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Večna pot 113, Ljubljana, 2019.

[12] ——, "skrelief," https://github.com/Architecton/skrelief, 2020.