THE USE OF NEGATIVE SAMPLING IN THE EVALUATION OF LINK PREDICTION ALGORITHMS

Julian Robinson

Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

Thesis Advisor: Mehmet Koyuturk

Department of Computer and Data Sciences

Case Western Reserve University

January, 2023

The Use of Negative Sampling in the Evaluation of Link Prediction Algorithms

Case Western Reserve University
Case School of Graduate Studies

We hereby approve the thesis[1] of

Julian Robinson

for the degree of

Master of Science

Mehmet Koyuturk

Committee Chair, Advisor — Date
Department of Computer and Data Sciences

Michael Lewicki

Committee Member — Date
Department of Computer and Data Sciences

Soumya Ray

Committee Member — Date
Department of Computer and Data Sciences

Date of Defense:

7/22/2022

[1]We certify that written approval has been obtained for any proprietary material contained therein.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to acknowledge and thank my advisor Mehmet Koyuturk Ph.D. for his help during my M.S. study and research. I would also like to thank Serhan Yilmaz for his continuous advice, knowledge, enthusiasm, and patience during this process. In addition, I would like to thank Michael Lewicki Ph.D. and Soumya Ray Ph.D. for being a part of my thesis defense committee.

I would like to thank Case Western Reserve University and specifically the Computer and Data Sciences Department for the opportunity to take part in and execute the research that I have.

Lastly, I would like to thank my parents and siblings for their constant love, guidance, and support during this process and always.

ABSTRACT

The Use of Negative Sampling in the Evaluation of Link Prediction Algorithms

JULIAN ROBINSON

Link prediction is a constantly growing field, but the evaluation of newly developed algorithms requires a lot of computational resources that can be prohibitively expensive to perform on large networks. To resolve this issue, a possible approach is to reduce the computational complexity by randomly sampling the negative edges. Here, we investigate the effect of negative sampling on the evaluation of link prediction algorithms, propose models to estimate the sampling error based on the number of negative edges sampled, and suggest minimum values bounding the error to a desired amount. Across a wide-array of real networks, we show that the suggested values can appropriately bound the error and can speed up the evaluation process 1000x times for large networks having 10^6 nodes with minimal error. We anticipate that these results and our estimated model can help researchers keep the evaluation of link prediction methods accessible on large, real-world networks.

1 Introduction

Link prediction on a network is an integral part to understanding its properties, uses, and has a wide variety of applications in areas of Computer and Data Science, Biology, and more. In general, it is the task of determining whether a relationship exists between two or more entities within a network. Link prediction, therefore, has become a very useful tool in the areas of biological networks, social networks, and other applications.

1.1 Problem Statement

Link prediction is a classification task used to determine whether a relationship exists between two vertices in a graph. Given a network G = (V, E), where V is the set of all vertices in G and E is the set of all true edges between vertices within G. These edges, E, will henceforth be referenced as observed links. The objective of link prediction is to identify unobserved links within G. A link prediction method is applied over the graph resulting in a trained model. Using this model on all the negative edges of the graph, edges are predicted and said to be the predicted edges. This number of total possible edges is calculated similarly to the famous "handshake problem"[30] (Equation 1.1).

$$|\text{MaxE}| = \frac{|V|(|V|-1)}{2} \tag{1.1}$$

1.2 Applications of Link Prediction

In the context of biological networks, link prediction can be used on homogeneous networks such as human protein-protein interaction networks to determine if two proteins may have a previously unknown interaction. After link prediction is performed, these interactions may be confirmed in a clinical setting. In the context of social networks, link prediction can be used to determine whether a connection between two users should be present based off their own and others' interactions.

Both biological and social networks can be very large in both the number of nodes and number of edges in the network. Due to the size of the network, link prediction can be very computationally inefficient.

Our work builds off many preexisting link prediction methods. In this section, we introduce some of these methods and metrics, their uses, and drawbacks.

1.3 Link Prediction Methods

1.3.1 Topology Based

One of the simplest genres of link prediction methods is topology-based. The general idea of topology-based link prediction methods is that similar nodes are more likely to form a link[29]. These methods determine an index that indicates the similarity between two nodes. Topology-based methods can be divided into three different categories: neighbor-based, path-based, and random walk-based.[29]. In our work, we use neighbor-based topology methods. Neighbor-based methods, in the context of a social network, say that two users who are "close" in the network

will have colleagues in common and will travel in similar circles. This then means that they are more likely to interact in the future[13]. Examples of neighbor-based methods are Common Neighbors, Jaccard Similarity Coefficient, Preferential Attachment, and Adamic Adar. Common Neighbors gives the number of nodes that both nodes share an edge with. Jaccard Similarity Coefficient builds off the Common Neighbors method. Jaccard Similarity normalizes the number of shared neighbors by total number of neighbors. Preferential Attachment revolves around the idea that a new link is more likely to form between nodes of high degree[3]. Adamic Adar is based on the idea that in a social network, users that interact with highly connected users are more likely to form a link[1].

1.3.2 Embedding Based

Another general category of link prediction is embedding based. An embedding is a low-dimensional representation of a higher-dimensional object. Networks are high-dimensional objects, so it is difficult to apply mathematical evaluations on them. So, an embedding representation of the network can be constructed in order to apply these methods. There are two main ways to embed features of a network: Edge Embedding and Node Embedding. An edge embedding gives a vector representation for each edge in the network. A node embedding gives a vector representation for each node in the network. In the context of link prediction, node embedding is more commonly used as it allows for the characterization and visualization of edges that do not yet exist in the network. In other words, node embeddings can be combined to form node-pair embeddings for nodes that do and

do not already exist in the network. Evaluation can then be performed on these node-pair embeddings.

Node embeddings of networks are calculated in a variety of ways. Some node embedding methods like DeepWalk[18] and Node2Vec[7] use short random walks to learn a vector representation of the network. LINE[26] uses the first-order proximity (local pairwise proximity between vertices) and second-order proximity (assumption that nodes with shared neighbors are likely to be similar) to calculate an embedding. CNE[10] uses derived (generally topological) information about the network to construct a distribution over edges that is then used to construct an embedding. Generally, after the node embeddings are created, each node pair embedding is constructed by using a binary operator on each pair of node embeddings[7]. After the node pair embeddings are constructed, a prediction model is trained (often logistic regression) using the embeddings of the positive edges in the graph and a subset of the negative edges. The now trained model is then used to predict links.

1.4 Evaluating Link Prediction Algorithms

1.4.1 What is the Ground Truth?

In order to evaluate link prediction methods, they need to be tested and evaluated on labeled data. In the absence of a gold standard set of labels, these training and testing sets are typically obtained by sampling the edges uniformly at random. So, in a supervised learning setting, link prediction is performed by first removing k edges from E at random creating a new network G'=(V, E'). This is a training graph

and the removed k edges form the testing graph. Note that |E'| + k = |E|. Figure 1.1 shows an example of a train-test split.

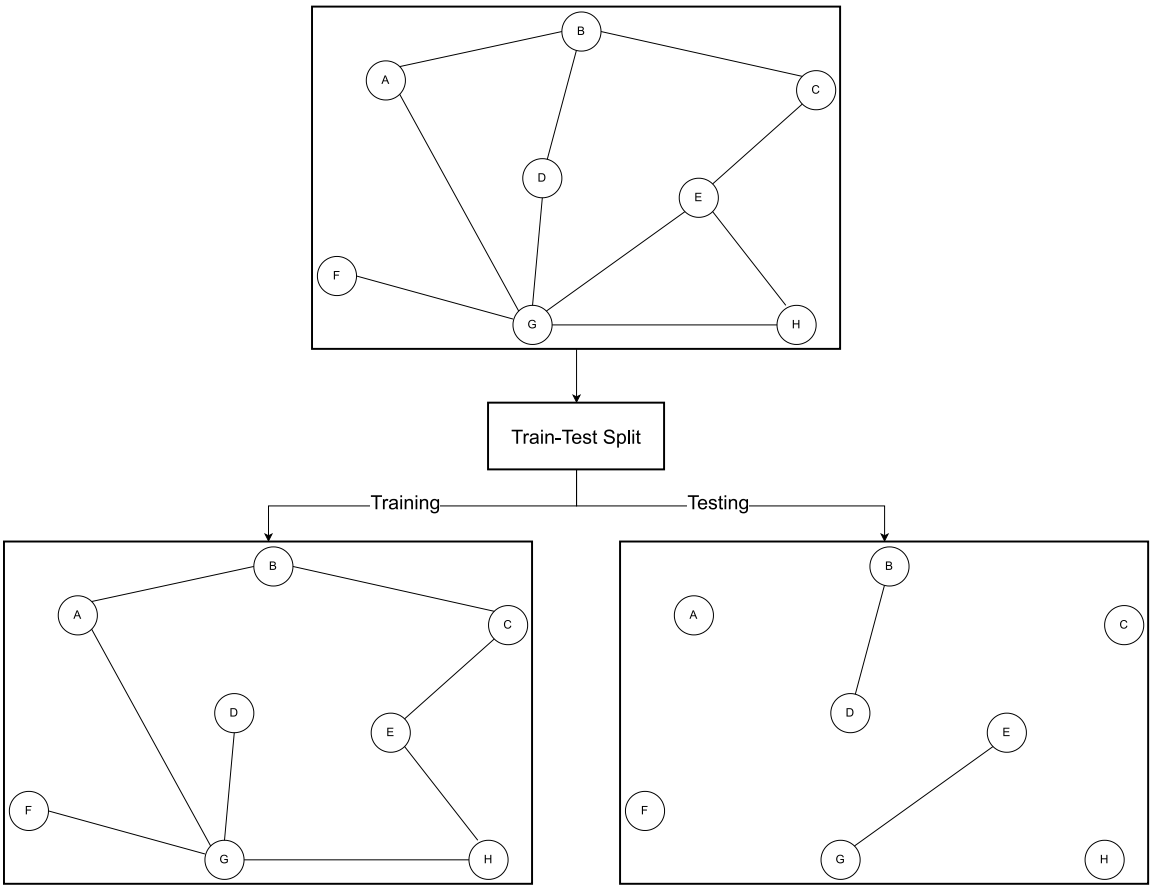


Figure 1.1. **Train-Test Split Diagram.** Edges (80%) in initial graph are randomly selected to be in the training graph. Remaining edges (20%) are selected to be in the testing graph. Note: In an ideal situation, few or no nodes are left unconnected in the training graph.

After the model has been trained, the k removed edges along with the set of negatives are predicted by the model. Therefore, the number of edges predicted in this evaluation phase can be found in Equation 1.2.

$$|\text{E''}| = \frac{|V|(|V|-1)}{2} - |E'| \tag{1.2}$$

This process of link prediction is illustrated in Figure 1.2.

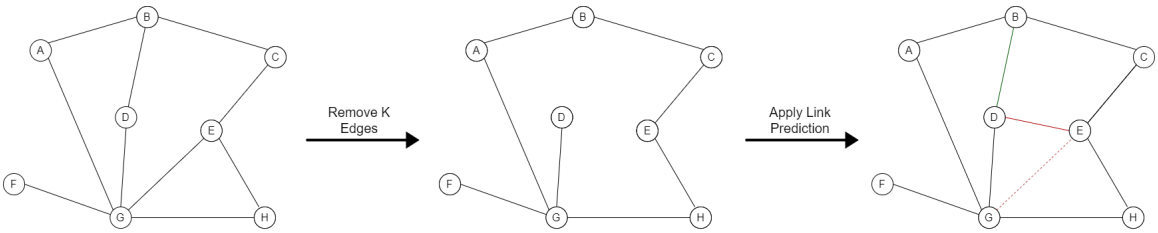


Figure 1.2. **Workflow of Evaluation of Link Prediction.** Green edges are True Positive. Solid red edges are False Positive. Dashed red edges are False Negative

1.4.2 Evaluation Metrics

The effectiveness of a given link prediction method can be calculated in a multitude of different ways, some of which are mentioned later in this section. In general, the predicted edges are compared to the observed edges to determine how effective the given metric was on predicting edges in the given graph.

In Figure 1.3, the predicted edge status (positive or negative) is mapped vs the actual edge status. Edges that are predicted positive and are actually positive are defined as True Positives (TP). Edges that are predicted positive and are actually negative are defined as False Positives (FP). Edges that are predicted negative and are actually positive are defined as False Negatives (FN). Edges that are predicted negative and are actually negative are defined as True Negatives (TN).

Accuracy. One may at first assume that accuracy (Equation 1.3), the ratio of correctly predicted edges to total edges, would be the best metric to determine the effectiveness of a method, but for link prediction, the number of positive and negative samples must be taken into account. In a network setting, there are generally

Confusion Matrix		Predicted	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Figure 1.3. **Confusion Matrix.**

far more negative samples than positive samples. Consider a classifier that predicts all node pairs as negative. As the number of nodes in a graph increases, the accuracy approaches 1. The classifier would have scored well, but it is clear that its predictions are useless.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{1.3}$$

AUROC. Area under receiver operating characteristic curve, AUROC, is in general a better indicator of the performance of a metric. The ROC curve is defined as the plot of the true-positive rate versus the false-positive rate. The calculations for true positive rate (TPR) and false-positive rate (FPR) are seen in Equations 1.4 and 1.5

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1.4}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{1.5}$$

where TP, TN, FP, and FN are the number of true positives, true negatives, false positives, and false negatives respectively. An example of an ROC curve can be seen in Figure 1.4.

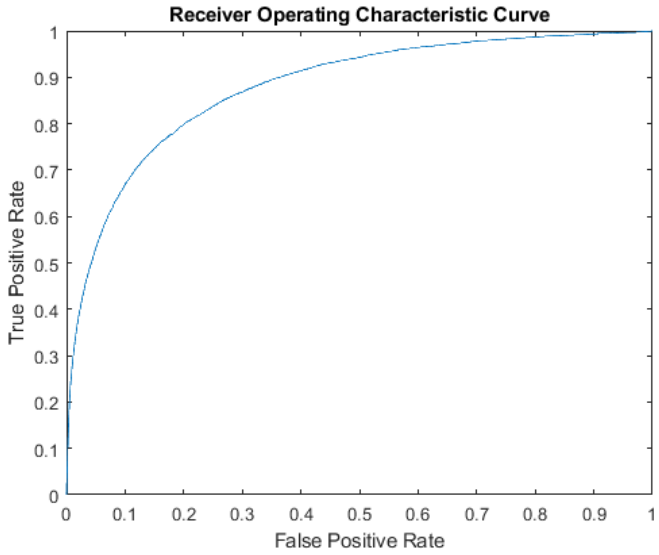


Figure 1.4. **Sample ROC Curve.** True Positive Rate (TPR) vs False Positive Rate (FPR)

The AUROC is therefore the area underneath this curve. A high AUROC means the index performed well. The maximum value of 1 would therefore be achieved if the entire area under the curve filled the 1 by 1 grid. For classification models that output a score rather than a prediction, the TPR and FPR are calculated for different decision thresholds between 0 and 1.

Precision. Precision is another metric commonly used in link prediction, and is defined as the number of true positives over the total predicted positive. This calculation can be seen in 1.6.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1.6}$$

where TP is the number of true positives, and FP is the number of false positives. Precision is very similar to Recall, its sibling metric. Recall, while not directly used in this research, is calculated as the number of true positives divided by the total of actual positives.

Odds Ratio. Odds ratio is a statistic that is used to measure the association between two events. The odds ratio is defined as the odds that an outcome will occur given a particular event, compared to the odds of the outcome occurring without the particular event[25]. While commonly used in medical applications, the odds ratio is useful in link prediction contexts as a metric that tells the odds that a pair of nodes is linked given the result of a specific link prediction algorithm[35]

The odds ratio calculation can be seen in 1.7.

$$\text{Odds Ratio} = \frac{\text{TP}/\text{FN}}{\text{FP}/\text{TN}} = \frac{\text{TP} \cdot \text{TN}}{\text{FP} \cdot \text{FN}} \tag{1.7}$$

F1 Score. F1-Score is a another statistic commonly used in link prediction[4 17 23]. It combines both the precision and recall metrics by taking their harmonic mean. The calculation of this can be seen in Equation 1.8.

$$F_1 = \frac{2}{\text{Recall}^{-1} + \text{Precision}^{-1}} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{1.8}$$

The highest F1 Score possible is 1 indicating both a perfect precision and recall score.

1.5 Use of Negative Sampling for Evaluation

Link prediction on large networks can be prohibitively computationally expensive[36]. The computational complexity of a link prediction problem is largely dependent on the number of vertices in the network, as the prediction index must be calculated $|MaxE|$ times. In order to combat this issue, negative sampling can be performed[14 22 28]. Negative Sampling, also referred to as 'test set sampling', is the process of omitting some of the observed negative edges in order to reduce the number of computations that must be performed.

Removal of negative edges can, of course, reduce the information about a network that we have resulting in a trade-off between computational time and effectiveness of the method. One paper varies the negative sampling percentage, p, to see its effect on the variance in the AUROC of various link prediction methods on their networks[33]. This research concluded that an amount lower than 1 percent of the initial negative edges results in a variance too large to consider the results viable. This research looked at networks of very similar node amounts. They used four networks of node size 1829, 3215, 13873, and 16922. These node sizes are relatively small. Their results do not extend to networks of larger node sizes. Therefore, the results brought forth in their research are not generalizable to all contexts.

In the research presented in this paper, we look to repeat their approach on larger networks. The general focus of the aforementioned paper was on the ratio of negative sampling. We also look to vary the number of negative sampled edges irrespective of the total number of possible edges. In addition to repeating their approach on larger networks, we look to characterize the error due to sampling based on the variance when different samplings are used. Doing so will allow us

to determine how many negative edges are needed to be sampled to bound the sampling error. Lastly we look to characterize the sampling error relative to the train-test error. Doing so will allow us to determine how many edges are needed to be sampled to bound the relative error.

1.5.1 Metrics and Negative Sampling

Not all evaluation metrics are safe to be used with negative sampling. For instance, when negative sampling is performed, the precision metric becomes biased towards positive samples (see Figure 1.5 for an illustration of the sampled and unsampled runs). We expect that if a metric is unaffected by negative sampling, both lines would overlap. Since in the case of precision these lines do not overlap, it is clear that the precision is biased and, without an adjustment to correct this bias, cannot be used as a reliable metric for link prediction with negative sampling. Similarly, odds ratio and F1-Score, which depend on the number of predictions, cannot safely be used for evaluation of link prediction with negative sampling without making an adjustment to correct the bias.

AUROC, however, is row normalized when constructing the TPR and FPR. This means that it both TPR and FPR are ratios that are dependent on the number of sampled edges not true edges. So, AUROC is unaffected by negative sampling. Thus, in this work, we will consider AUROC, and leave the other metrics as future work.

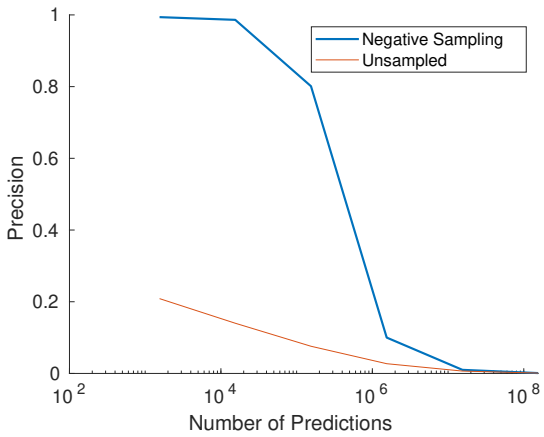


Figure 1.5. **Precision Bias due to Negative Sampling.** Precision vs Number of Predictions made for unsampled and negative sampling ratio of 10^{-3} for a single network.

2 Methods

The link prediction evaluation framework used in this research is in Figure 2.1. The input is a network represented as an edgelist where each entry is a tuple of nodes. The network is split 80/20 into a train and test set. The training set is used to train the model that will be used during testing. The positives of the test set are used directly in testing. Negative sampling is performed on the original network to obtain the negative samples that will be used in the testing phase. Starting with the entire network, the entire negative set is taken as all node pairs not in the edge list. Using the sampling ratio or the number of negatives to be sampled, this set is randomly sampled in order to determine the negative samples to be used in the testing phase. Lastly, the performance of the model is determined in the evaluation phase.

2.1 Datasets

The results of this work are reported using 18 different networks of varying node-size, edgesize and general topology. The networks used in this research can be found in Table 2.1. The networks chosen have node sizes on the order of between 10^3 and 10^6. Data preprocessing took the form of relabeling nodes to integers and constructing the network as an undirected edgelist.

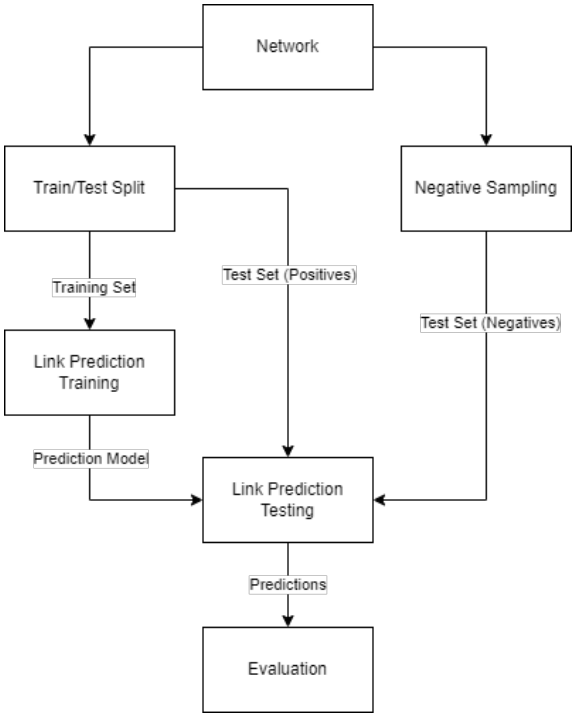


Figure 2.1. **Link Prediction Evaluation Framework.** Input is a network. Random sampling occurs during Train-Test Split and Negative Sampling steps.

2.1.1 Network Details

DrugBank DDI: []. Facebook denotes users on Facebook as nodes and edges between nodes denote users that are friends. LastFM Asia denotes users from Asian countries as nodes and edges are mutual follower relationships between them. PhosphoSite Plus 2019 and PhosphoSite Plus 2021 []. Biogrid Drosophila 2020, Biogrid Human 2010 and 2020 [] for Drosophila and Human genes respectively. Wormnet denotes genes in Caenorhabditis Elegans as nodes and []. HIPPIE []. Enron Email and EU Email denote users and emails between users as edges. Deezer-Romania, Hungary, and Croatia denote users on the streaming platform as nodes and mutual friendships as edges. Twitch Gamers denotes users on Twitch.com

as nodes and mutual follower relationships as edges. PTMcode denotes proteins as nodes and functional associations of post-translational modifications within and between proteins as edges. Amazon denotes products as nodes and common function or purposes between products as edges.

Network Name	**Nodes**	**Edges**	**Max Edges**	**Type**
DrugBank DDI[31]	1514	48514	1.15E+06	Biological
Facebook[27]	6540	12329	2.14E+07	Social
LastFM Asia[19]	7624	27806	2.91E+07	Social
PhosphoSite Plus 2019[8]	7807	10431	3.05E+07	Biological
PhosphoSite Plus 2021[8]	9354	12676	4.37E+07	Biological
Biogrid Drosophila 2020[24]	9536	62640	4.55E+07	Biological
Biogrid Human 2010[24]	9715	32347	4.72E+07	Biological
Wormnet[5]	16347	762822	1.34E+08	Biological
HIPPIE[2]	19484	773806	1.90E+08	Biological
Biogrid Human 2020[24]	25776	464003	3.32E+08	Biological
Enron Email[12]	36692	183831	6.73E+08	Social
Deezer-Romania[21]	41773	125826	8.72E+08	Social
Deezer-Hungary[21]	47538	222887	1.13E+09	Social
Deezer-Croatia[21]	54573	498202	1.49E+09	Social
Twitch Gamers[20]	168114	6797557	1.41E+10	Social
PTMcode[16]	191104	835061	1.83E+10	Biological
EU Email[11]	265009	364481	3.51E+10	Social
Amazon[32]	334863	925872	5.61E+10	Information

Table 2.1. **Network Information.**Networks used along with their respective node size, edge number, maximum edges possible given the node size and the type of network.

2.2 Link Prediction Methods

2.2.1 Topology-Based

In topology based link prediction, the method is applied to each positive edge remaining in the training graph and every negative edge (or every sampled negative edge). The result is that every node-pair is assigned a score and then all pairs are sorted by their scores in descending order. Thus, for making k predictions, the top k node-pairs with the highest scores are predicted as "positives" and the rest are predicted as "negatives".

Preferential Attachment. Degree is the number of edges a particular node has. In a link prediction context, the preferential attachment index is computed as the product of the degree of the two nodes. Node pairs of high degree are more likely to share an edge. The calculation can be seen in Equation 2.1.

$$C_{deg}(x,y) = |N(x))| * |N(y))| \tag{2.1}$$

where N(x) is the set of vertices that share an edge with x.

Adamic Adar. Adamic Adar is an index used to predict links based on their shared neighbors[1]. The index is calculated as the inverse sum of the degree of the common neighbors. Node pairs with more shared neighbors are more likely to share an edge. The calculation for Adamic Adar can be seen in Equation 2.2.

$$A(x,y) = \sum_{u \in N(x) \bigcap N(y)} \frac{1}{\log|N(u)|} \tag{2.2}$$

where N(x) is the set of vertices that share an edge with x.

Jaccard Similarity Coefficient. The Jaccard Similarity Coefficient is calculated as the number of common neighbors between two nodes normalized by the total

number of neighbors between both nodes. Nodes with more neighbors in common are more likely to share an edge. Also, nodes with high degree that also share many neighbors are deemed as less important than nodes with a higher proportion of shared neighbors to total neighbors. The calculation of the Jaccard Coefficient can be seen in equation 2.3.

$$Jaccard(x, y) = \frac{|N(x)| \cap |N(y)|}{|N(x)| \cup |N(y)|} \tag{2.3}$$

where N(x) is the set of vertices that share an edge with x.

2.2.2 Embedding-Based

Node2Vec. This prediction method combines using an embedding method to learn a mapping of nodes to a low-dimensional matrix with a binary classification algorithm. Node2Vec makes use of random walk to generate the embedding. The Node2Vec embedding process has three main steps. The first step is to calculate the edge transition probabilities. This is the probability that given a current node u at time t_0, the probability that at time $t_0 + 1$ the current node will be v. The homophily hypothesis states that nodes with common traits and features are more likely to interact with each other[34]. This means that nodes that are highly interconnected should be embedded closely together. In addition, by the structural equivalence hypothesis, nodes that perform similar roles in a network (e.g. acting as a junction point) should likewise be embedded closely together[15].

For the actual implementation of this link prediction method, the graph is first split into a train and test set. For the second step of Node2Vec, each node of the training graph is then embedded using 200 walks each of length 10 into a vector

of length 64. All of the positives from the training graph and an equal number of negatives are then mapped to features[6 9]. Each node has its own embedding, so to combine them, the Hadamard product is calculated for each pair of nodes[6]. The Hadamard product is illustrated in Equation 2.4.

$$\text{Hadamard}(u, v) = f(u) \circ f(v) \tag{2.4}$$

where $f(u)$ is the embedding for node u, and $\circ$ is the element-wise multiplication operator. These node-pairs' Hadamard products are used as the feature set along with their label as True or False in a logistic regression classifier for the third and final step. This classifier, once fit, is the model that predicts the test set as positive or negative.

3 Results and Discussion

3.1 Effect of Negative Sampling on AUC

We first compare our results to those found in previous work[33]. Figures 3.1a and 3.1b show recreations of the figures in the paper but with datasets of node size 1514 and 36692. As expected, we see that the error increases as the negative sampling percentage decreases. However, the extent to which this occurs is not consistent with that in the paper. "Condmat is stable down to 1% sampling of negative class instances while DBLP, Enron and Facebook are stable only down to 10% sampling of negative class instances."[33] Where Condmat, DBLP, Enron, and Facebook are networks of node sizes 13873, 3215, 16922, and 1829 respectively. In Figure 3.1b we see very little error even past the threshold of 1 percent which was declared in the paper.

In Figure 3.1c, we see that for a constant ratio of negative edges, the error decreases as the number of nodes in the network increases. When there is a constant number of negative edges, as the number of nodes increases, the sampling ratio is of course decreasing. So, one would expect based off previous work that the error would increase as the number of nodes increases. In Figure 3.1d, we do not see that. Instead, for a constant number of negative edges sampled, the error remains constant as the number of nodes in the network increases. This implies that the

error is dependent only on the the amount of edges sampled irrespective of the number of nodes in the network.

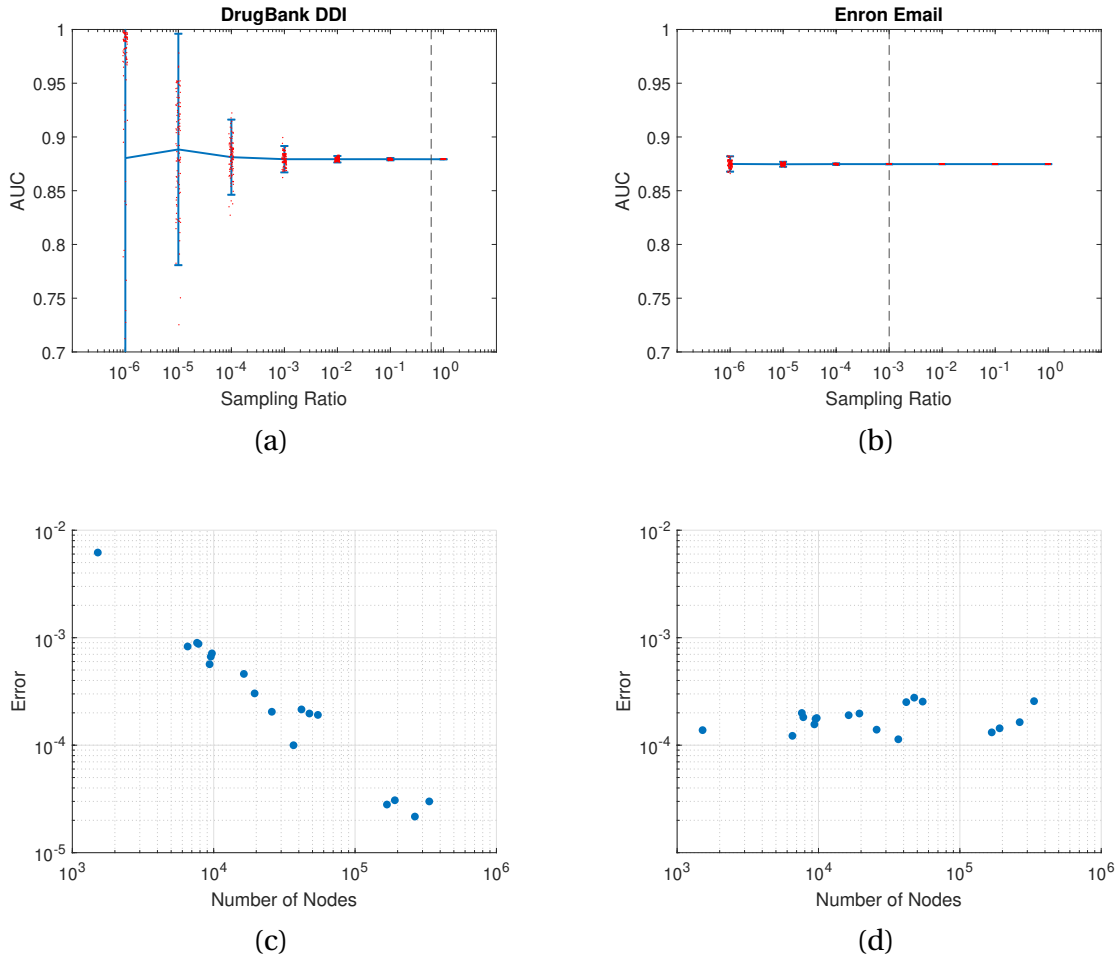


Figure 3.1. **Effect of Negative Sampling on AUC for Preferential Attachment as the Link Prediction Method.** (3.1a and 3.1b) AUC vs Sampling Ratio for DrugBankDDI and Enron Email respectively. Red points depict each iteration. Bars denote two standard deviations. (3.1c) Error vs. number of nodes for all networks with a fixed negative sampling ratio (10^{-3}). (3.1d) Error vs. number of nodes for all networks with a fixed number of negative samples (673133). This number of negatives is equivalent to the number of negatives for Enron Email with 10^{-3} sampling ratio. Vertical line in 3.1a and 3.1b at 673133.

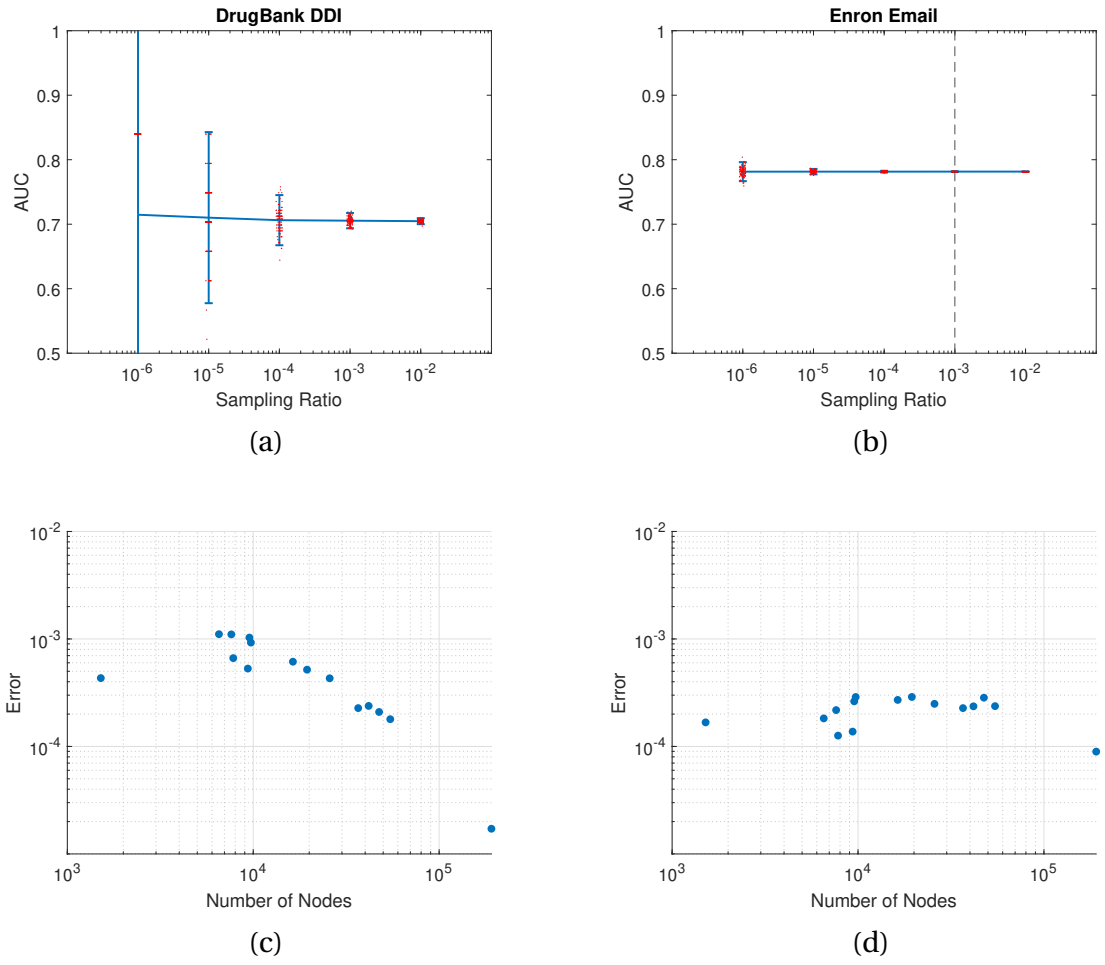


Figure 3.2. **Effect of Negative Sampling on AUC for Node2Vec as the Link Prediction Method.** Same figure as Figure 3.1 using Node2Vec as the link prediction method.

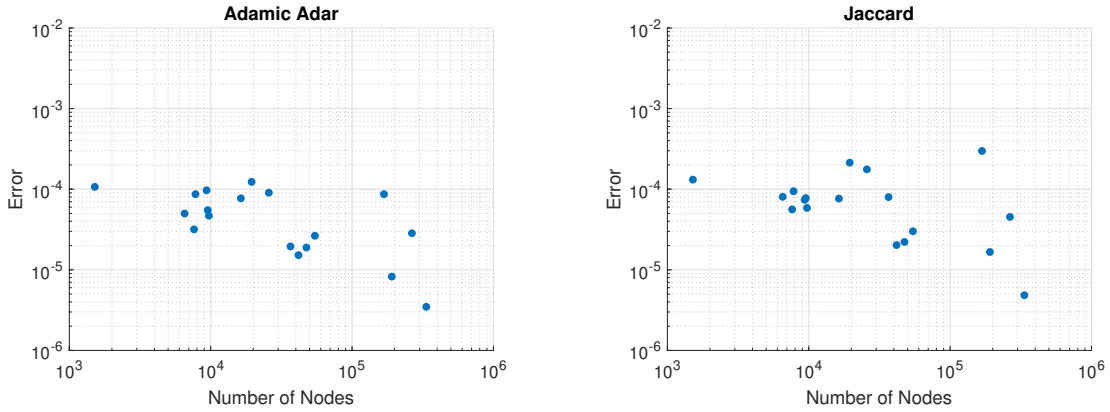
3.2 Additional Methods

The findings so far have been shown using preferential attachment as the method of link prediction. These findings, however, are generalizable to other methods as well. In an effort to display the robustness of these results, Figure 3.2 follows the form of 3.1 and shows the same conclusions, but with Node2Vec as the link prediction method. Node2Vec is considered by many to be more 'state of the art'

than the topology-based link prediction methods, so it is important to show that sampling is effective for this method.

Figure 3.3 shows the error in AUROC vs the number of nodes in the network for the networks shown in Table 2.1 for a set number of negative edges sampled. This was performed for the remaining of the four link prediction methods: Adamic Adar and Jaccard Index.

According to previous work[33], the error increases as the ratio of negative edges sampled decreases. Because the number of negative edges sampled is set, as the number of nodes in a network increases, the ratio decreases. Therefore, from their conclusions, we would expect the error to increase, but instead it stays relatively constant below 10^{-3} or decreases. So, the conclusions from Figure 3.1 extend to other link prediction methods.



(a) (b)

Figure 3.3. **Effect of Negative Sampling on AUC for Adamic Adar and Jaccard Similarity Coefficient as the Link Prediction Methods.** Error vs number of nodes for all networks with a fixed number of negative samples (673133).

3.3 Extent of the Error

In order to show that the error incurred during sampling is not significant, the error due to sampling, baseline error when varying the train-test splits, and error when performing sampling and varying the train-test splits (Combined Error) must be compared. Pseudocode for all three methods is in Algorithms 1, 2, and 3.

Algorithm 1 shows the steps taken to calculate the sampling error associated with the given method on the given network with a given sampling ratio. To isolate the sampling error itself different negative samples were used on the same train-test split (internal repeats) 10 times. This gave the error associated with just sampling. However, some train-test splits can affect the network topology much more than others, so these steps were repeated over many train-test splits (external repeats) 10 times. The average of the error for each external repeat was reported as the true sampling error.

Algorithm 1 Sampling Error

```
for nExt External Repeats do
    Graph_Train, Graph_Test ← Train-Test_Split(Graph)
    Trained_Model ← LP_Method(Graph_Train)
    for nInt Internal Repeats do
        Sampled_Edges ← Negative_Sample(Graph, kNegatives)
        Predictions ← Trained_Model(Graph_Test, Sampled_Edges)
        Scores[nInt] ← Evaluate(Graph_Test, Predictions)
    end for
    Errors[nExt] ← std(Scores)
end for
Error ← mean(Errors)
```

Algorithms 2 and 3 show the steps taken to calculate the train-test and combined error respectively. They also employ a series of external and internal repeats. The difference between these two and the sampling error algorithm is that the

train-test split varies for every iteration, so no iteration has the same train-test split. The difference between the train-test and combined error algorithms is the negative sampling that takes place in the combined error algorithm, but not the train-test error.

Algorithm 2 Train-Test Error

```
for nExt External Repeats do
    for nInt Internal Repeats do
        Graph_Train, Graph_Test ← Train-Test_Split(Graph)
        Trained_Model ← LP_Method(Graph_Train)
        Predictions ← Trained_Model(Graph_Test)
        Scores[nInt] ← Evaluate(Graph_Test, Predictions)
    end for
    Errors[nExt] ← std(Scores)
end for
Error ← mean(Errors)
```

Algorithm 3 Combined Error

```
for nExt External Repeats do
    for nInt Internal Repeats do
        Graph_Train, Graph_Test ← Train-Test_Split(Graph)
        Trained_Model ← LP_Method(Graph_Train)
        Sampled_Edges ← Negative_Sample(Graph, kNegatives)
        Predictions ← Trained_Model(Graph_Test, Sampled_Edges)
        Scores[nInt] ← Evaluate(Graph_Test, Predictions)
    end for
    Errors[nExt] ← std(Scores)
end for
Error ← mean(Errors)
```

Figure 3.4 contains four bar graphs that show error with sampling, error without sampling, and the error with both sampling and train-test splitting performed in tandem for two different networks. These charts show that the combined error is not much larger than the error without sampling. Combined with the above mentioned fact that the mean AUROC is not affected by sampling, we conclude

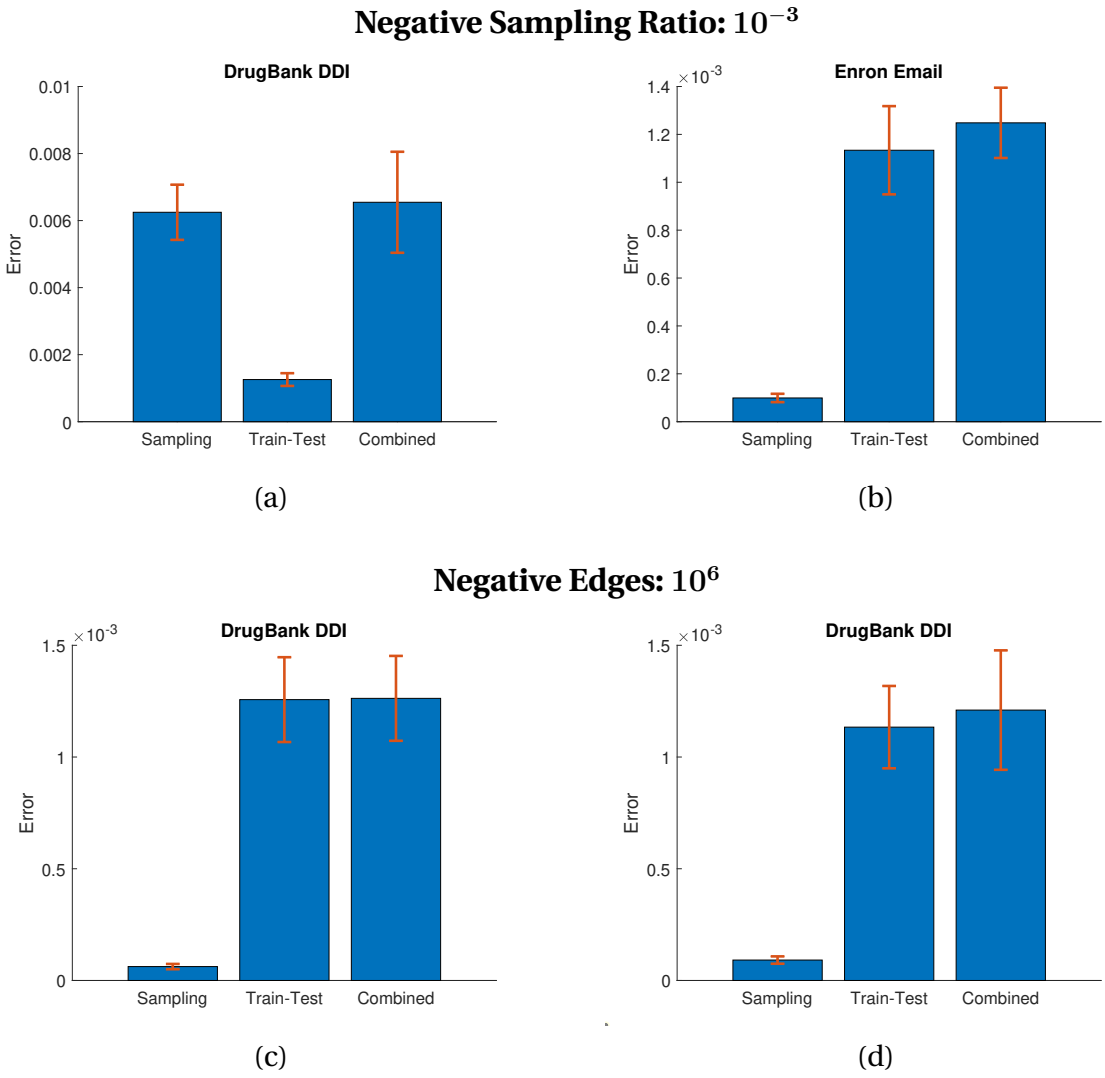


Figure 3.4. **Sampling, Train-Test, and Combined Errors for given networks and sampling strategy.** (3.4a) DrugBank DDI with sampling ratio 10^{-3}. (3.4b) Enron Email with sampling ratio 10^{-3}. (3.4c) DrugBank DDI with 10^6 negative samples. (3.4d) Enron Email with 10^6 negative samples.

that sampling is an effective way to reduce computational complexity while maintaining the effectiveness of the link prediction task.

The bar graphs above showed the error for two selected networks. The conclusions brought forth are, of course, extendable to all of the networks referenced thus

far. As mentioned before, the error due to sampling was calculated separately from the train-test and combined errors. In order to determine exactly how much the error will increase as negative sampling occurs, the ratio between the combined and train-test errors should be examined. As many of the networks used are quite large, the train-test error cannot be determined due to runtime and space constraints. Because the combined error procedure calculates the error due to sampling and train-test splitting, the sum of the sampling error and train-test error gives a rough estimate of the combined error (Equation 3.1).

$$\text{Error}_{\text{Sampling}} + \text{Error}_{\text{Train-Test}} \approx \text{Error}_{\text{Combined}} \tag{3.1}$$

So, the difference of the combined and sampling error also gives estimate of the train-test error (Equation 3.2).

$$\text{Error}_{\text{Train-Test}} \approx \text{Error}_{\text{Combined}} - \text{Error}_{\text{Sampling}} \tag{3.2}$$

We then say the ratio of combined error to the difference of combined and sampling error (Equation 3.3) provides a good estimate of the increase in error due to sampling.

$$\delta_{Error} \approx \frac{\text{Error}_{\text{Combined}}}{\text{Error}_{\text{Combined}} - \text{Error}_{\text{Sampling}}} \tag{3.3}$$

3.4 Finding Suggested Number of Sampling Edges

In order to understand how much sampling can be done without a large increase in error, a mathematical estimation of needed negative samples is useful.

Figure 3.5 shows a series of figures used to determine the minimum number of negative edges needed to ensure the error stays below a certain threshold. As has been stated, without performing negative sampling on some of the large networks, it is infeasible to perform link prediction.

Because some of the networks are so large, it is not realistic to perform baseline train-test iterations. The difference between the combined and sampled error provides a rough estimate of the baseline train-test error. In Figure 3.5a, the baseline train-test error is graphed vs the estimated train-test error for small networks. The points lie along the line $y = x$, meaning that the estimate given in Equation 3.2 is an accurate substitution for the true baseline error for this number of negative edges sampled. Therefore, this estimate can be used on larger networks as well. In Figure 3.5b, the estimated train-test error and sampling error are graphed for all networks for a fixed number of negatives. The estimated train-test error decreases while the sampling error increases. So, for the In Figure 3.5c as the number of negative edges sampled increases, the sampling error predictably decreases linearly (in log-log scale).

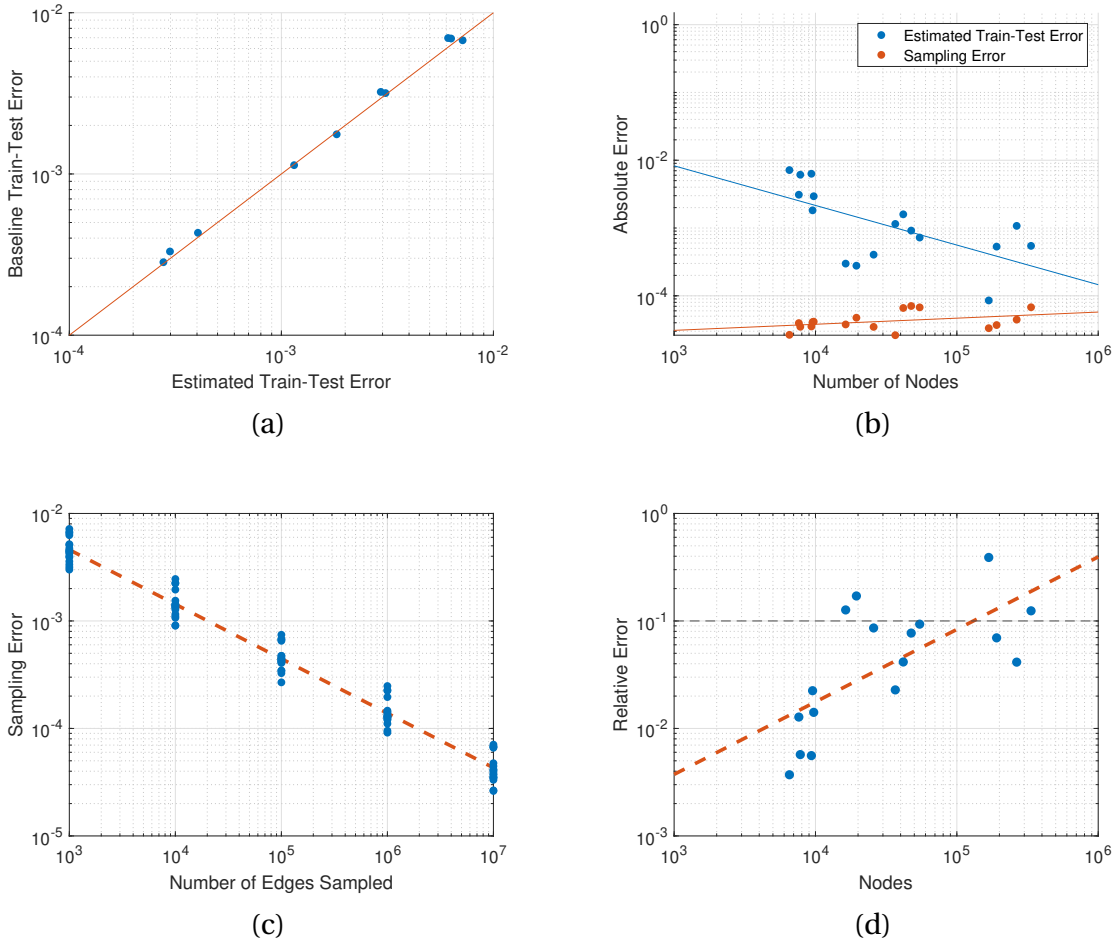


Figure 3.5. **Estimating the Absolute and Relative Error Associated with Negative Sampling** (3.5a) Baseline Train-Test Error vs. the Estimated Train-Test Error for a subset of networks (with node sizes less than or equal to that of Enron Email, see Table 2.1 for details). The red line indicates $y = x$ (estimated train/test error equal to the measured one). (3.5b) depicts the estimated baseline error and the sampling error with their corresponding best fit lines (indicated by blue or red colors) for all networks. 10^7 negative edges were used for link prediction. (3.5c) shows the sampling error and the corresponding best fit line with respect to the number of edges sampled for all networks. (3.5d) shows the relative error vs.. the number of nodes for all networks (for 10^7 sampled negative edges). The black horizontal line indicates the 10% error level.

3.4.1 Bounded Absolute Sampling Error

The best-fit line for sampling error seen in 3.5b is of the form

$$\log_{10}(\text{error}_{\text{abs}}(n)) = \beta + \alpha \log_{10}(n) \tag{3.4}$$

Where n is the number of nodes and α and β are parameters of the fit line. Given that the number of negatives was fixed at 10^7, we can say that:

$$\text{error}_{\text{abs}}(n, t : 10^7) = \beta_{\text{abs}} n^{\alpha_{\text{abs}}} \tag{3.5}$$

where t is the number of negatives sampled, $\beta_{\text{abs}} = 10^{\beta}$ and $\alpha_{abs} = \alpha$. From Figure 3.5c, we can see that as the number of edges sampled increases by a factor of 100, the sampling error approximately decreases by a factor of 10. So, we have:

$$\log_{10}(\text{error}_{\text{abs}}(t)) \propto \frac{\log_{10}(t)}{2} \tag{3.6}$$

In other terms,

$$\text{error}_{\text{abs}}(t) \propto \frac{1}{\sqrt{t}} \tag{3.7}$$

The general case for the absolute error is therefore,

$$\text{error}_{\text{abs}}(n, t) = \beta_{\text{abs}} n^{\alpha_{\text{abs}}} \frac{\sqrt{t_0}}{\sqrt{t}} \tag{3.8}$$

where t_0 is equal to 10^7 and satisfies Equation 3.4.

In order to determine the number of negative edges needed to sample to achieve the desired error, we define the desired error as γ. Thus, solving $\gamma = error_{abs}$ for the desired t, we get the following equation:

$$t_{\text{desired}}(n,\gamma) = \left(\beta_{\text{abs}} n^{\alpha_{\text{abs}}} \frac{\sqrt{t_0}}{\gamma}\right)^2 \tag{3.9}$$

Where γ is the desired absolute error. This equation selects t that would result the absolute sampling error to be equal to γ on average. Substituting in our empirically determined values for $\alpha_{abs} \approx 0.066$, $\beta_{abs} \approx 4.42 \times 10^{-5}$ and $t_0 = 10^7$, we get:

$$\begin{aligned} t_{\text{desired}}(n,\gamma) &= \left(4.42 \times 10^{-5} n^{0.066} \frac{10^{3.5}}{\gamma}\right)^2 \\ &= \left(\frac{0.14 n^{0.066}}{\gamma}\right)^2 \\ &\approx \frac{0.02 n^{0.13}}{\gamma^2} \end{aligned} \tag{3.10}$$

In order to ensure that all error is below γ, we introduce a buffer, multiplying the number of negatives by a factor of 3 (chosen based off the mean absolute error from fitted line).

$$t_{\text{desired}}(n,\gamma) = \frac{0.06 n^{0.13}}{\gamma^2} \tag{3.11}$$

Thus, for the results in this paper, the suggested values for negative edges were given according to the above equation (with a buffer of 3x) to ensure the values are truly within the range desired. Figure 3.6 shows the suggested number of sampled edges vs. the number of nodes in the network for γ values of 10^{-4}, 10^{-3}, and 10^{-2}.

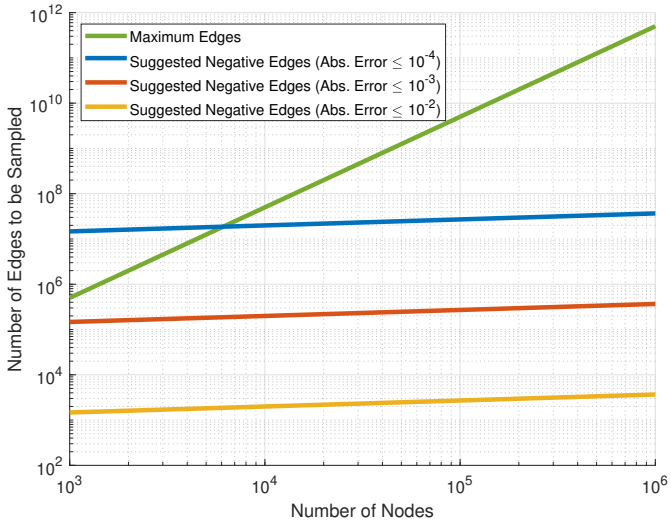


Figure 3.6. **Suggested Negative Edges Given Expected Absolute Error.** Number of Negative Edges to be Sampled vs Number of Nodes in the Network. Green line signifies the maximum number of edges in a network given the number of nodes (Equation 1.1). Other lines signify the suggested number of negative edges to sample in order to limit the absolute error to the specified value.

Figure 3.7 shows the absolute sampling error vs. number of nodes for the three different values of gamma. For networks where the maximum number of edges surpasses the suggested number of edges, the maximum number of edges is used. For each value, the corresponding values are close to, yet below the line indicating the maximum error allowed.

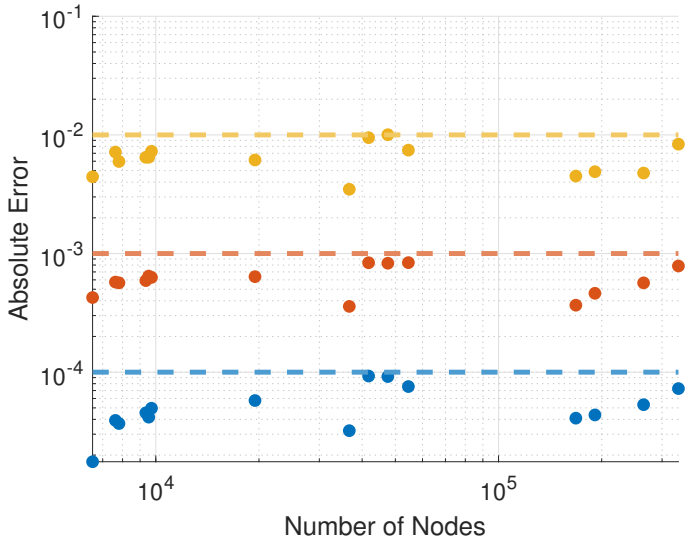


Figure 3.7. **Absolute Error using Suggested Number of Negative Edges.** Absolute error vs number of nodes for Preferential Attachment as the link prediction method for γ values of 10^{-4}, 10^{-3}, and 10^{-2}.

3.4.2 Bounded Relative Error

The best-fit line for sampling error seen in 3.5d is also of the form

$$\log_{10}(\text{error}_{\text{rel}}(n)) = \beta_{\text{rel}} + \alpha_{\text{rel}}\log_{10}(n) \tag{3.12}$$

Using a similar procedure to find the optimal number of negative edges to sample, we arrive at an equation of the same form as Equation 3.9 (Equation 3.13).

$$t_{\text{desired}}(n,\delta) = \left(\beta_{rel} n^{\alpha_{rel}} \frac{10^{3.5}}{\delta}\right)^2 \tag{3.13}$$

The number of negatives to be sampled vs the number of nodes in the network is seen in Figure 3.8.

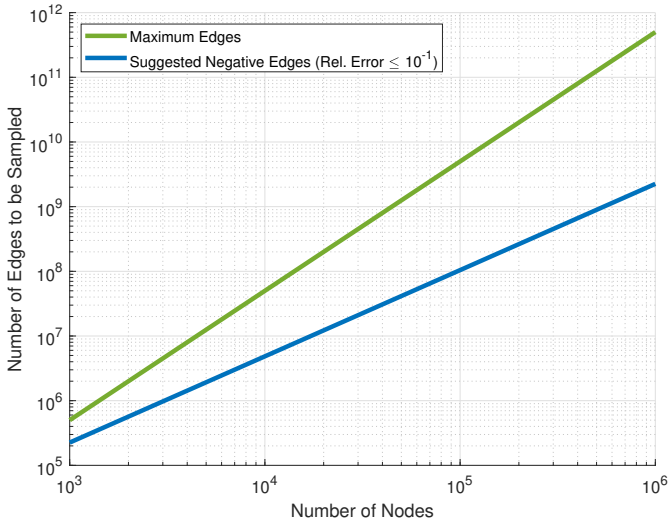


Figure 3.8. **Suggested Negative Edges Given Expected Relative Error.** Number of edges to be sampled vs number of nodes. Red line shows the maximum number of negative edges in the network given the number of nodes. Blue line shows the suggested number of negative edges to sample given the number of nodes.

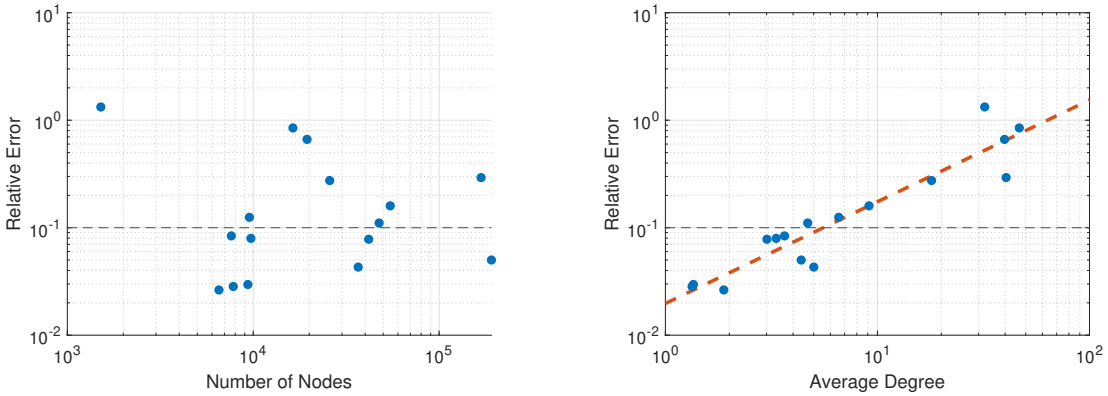
Using this model to perform link prediction, and setting δ at 10^{-1} we arrive at Figure 3.9a. Clearly, the relative error has not been bounded at 10^{-1} as many points are above the threshold line. The points that are above the threshold line, in general, have high average degree. So, plotting relative error vs average degree, we see a clear upward trend.

We can then represent the relative error as a function of average degree. We say

$$\text{error}_{\text{rel}}(d, n : n_0,\ t : t_0) = \beta^0_{\text{rel}} d \tag{3.14}$$

where d is the average degree of the network. So, the relative error is proportional to the average degree.

$$\text{error}_{\text{rel}} \propto d \tag{3.15}$$



(a) (b)

Figure 3.9. **Relation of Error to Average Degree.** (3.9a) Relative error vs. nodes. (3.9b) Relative error vs. average degree. Y values are the same.

The original relative error equation then needs an adjustment factor relative to average degree (Equation 3.16 and Figure 3.10):

$$\text{factor}_{\text{adj}}(d) = \beta'_{\text{rel}} d = \frac{\text{error}_{\text{rel}}(d, n : n_0, t : t_0)}{\text{error}_{\text{rel}}(n : n_0, t : t_0)} \tag{3.16}$$

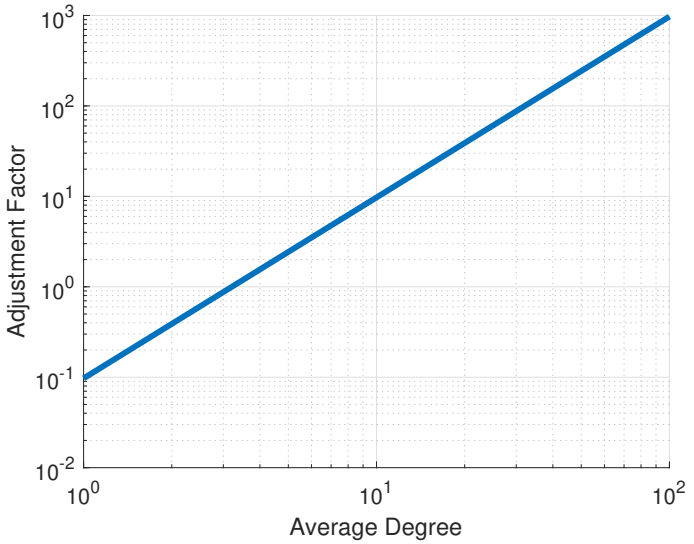


Figure 3.10. **Adjustment Factor given Average Degree.**

Thus, substituting gives the following β'_{rel}:

$$\beta'_{rel} = \frac{\beta^0_{rel}}{\beta_{rel}} \tag{3.17}$$

In all, we say

$$\begin{aligned} \text{error}_{\text{rel}}(d, n, t) &= \text{error}_{\text{rel}}(n,\ t) * \text{factor}_{\text{adj}}(d) \\ &= \left(\beta_{rel} n^{\alpha_{rel}} \frac{\sqrt{t_0}}{\sqrt{t}}\right) \left(\frac{\beta^0_{\text{rel}}}{\beta_{\text{rel}}} d\right) \\ &= \frac{\sqrt{t_0}}{\sqrt{t}} \beta^0_{rel} n^{\alpha_{rel}} d \end{aligned} \tag{3.18}$$

Next, we combined the constant factors into a single variable $\beta^{0'}_{rel}$:

$$\begin{aligned} \beta^{0'}_{rel} &= \sqrt{t_0} \beta^0_{rel} \\ \text{error}_{\text{rel}}(d, n, t) &= \frac{\beta^{0'}_{rel} n^{\alpha_{rel}} d}{\sqrt{t}} \end{aligned} \tag{3.19}$$

Solving the above equation for t that gives a relative error equal to δ, we get the following:

$$\text{desired}_{t_{\text{rel}}}(n, d, \delta) = \left(\frac{\beta^{0'}_{\text{rel}} n^{\alpha_{\text{rel}}} d}{\delta}\right)^2 \tag{3.20}$$

Substituting in our empirically determined values for $\alpha_{\text{rel}} \approx 0.667$ and $\beta^{0'}_{rel} \approx 0.027$, we get:

$$\text{desired}_{t_{\text{rel}}}(n, d, \delta) = \left(\frac{0.027 n^{0.667} d}{\delta}\right)^2 \tag{3.21}$$

Similarly to absolute error, in order to ensure the error is below δ, we introduce a buffer factor and multiply the desired t_{rel} by 3 (chosen based on the mean absolute error of the points to the fitted line). We finally arrive at Equation 3.22 which

specified our suggested amount of negative edges t_{rel} that bounds the relative error by δ:

$$\text{desired}_{t_{\text{rel}}}(n, d, \delta) = 3\left(\frac{0.027n^{0.667}d}{\delta}\right)^2 \tag{3.22}$$

Using this new equation for the suggested number of edges setting δ to 10^{-1}, we get Figure 3.11. For networks where the maximum number of edges surpasses the suggested number of edges, the maximum number of edges is used. We can see that the error is bounded by the 10% error threshold provided.

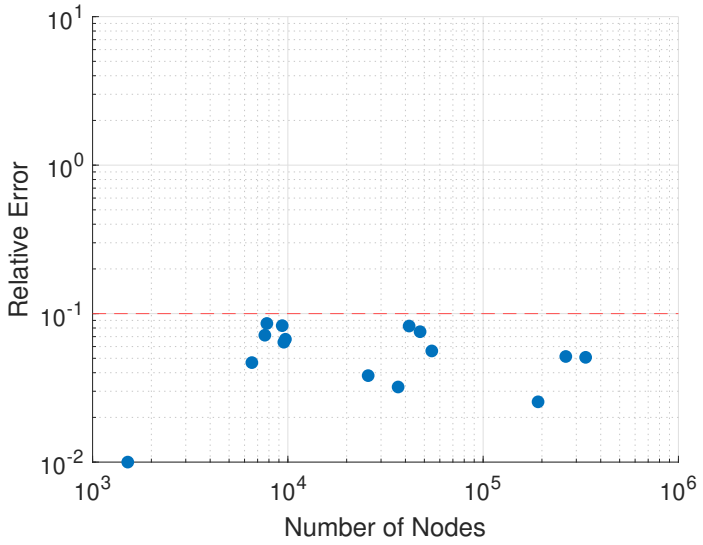


Figure 3.11. **Relative Error using Suggested Number of Negative Edges.** Relative error vs number of nodes for Preferential Attachment as the link prediction method for δ value of 10^{-1}.

Table 3.1 shows the suggested amount of edges needed to stay below the absolute or relative error bound. Clearly, sampling can be performed on large networks to a great extent without sacrificing performance of the model as the number of edges required are generally below the maximum number of edges.

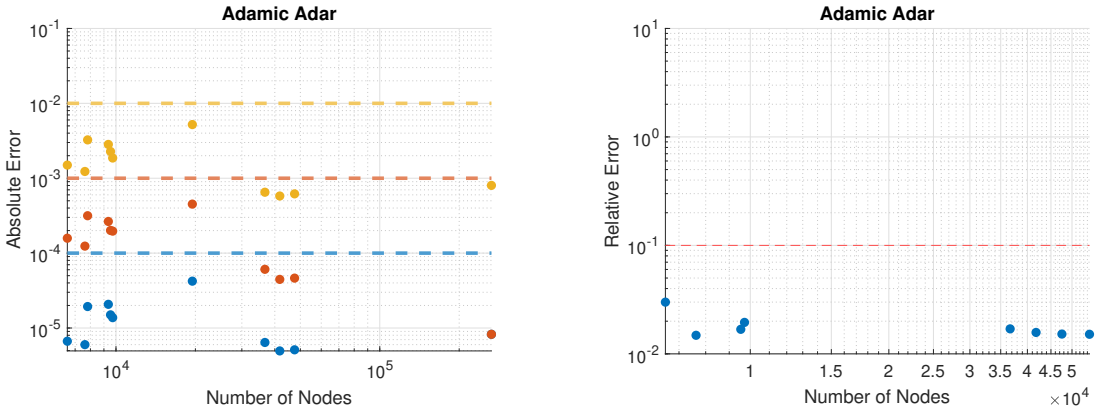
Network Name	**Max Edges**	**Absolute Error** ($\leq 10^{-4}$)	**Relative Error** ($\leq 10^{-1}$)
DrugBank DDI	$1.10 \cdot 10^6$	$1.10 \cdot 10^6$	$1.10 \cdot 10^6$
Facebook	$2.14 \cdot 10^7$	$1.88 \cdot 10^7$ (≈1.1x)	$9.51 \cdot 10^4$ (≈220x)
LastFM Asia	$2.90 \cdot 10^7$	$1.92 \cdot 10^7$ (≈1.5x)	$4.37 \cdot 10^5$ (≈67x)
PhosphoSite Plus 2019	$3.05 \cdot 10^7$	$1.92 \cdot 10^7$ (≈1.6x)	$6.05 \cdot 10^4$ (≈500x)
PhosphoSite Plus 2021	$4.37 \cdot 10^7$	$1.97 \cdot 10^7$ (≈2.2x)	$7.92 \cdot 10^4$ (≈550x)
Biogrid Drosophila 2020	$4.54 \cdot 10^7$	$1.98 \cdot 10^7$ (≈2.3x)	$1.91 \cdot 10^6$ (≈24x)
Biogrid Human 2010	$4.72 \cdot 10^7$	$1.98 \cdot 10^7$ (≈2.4x)	$5.03 \cdot 10^5$ (≈94x)
Wormnet	$1.33 \cdot 10^8$	$2.12 \cdot 10^7$ (≈6.3x)	$1.33 \cdot 10^8$
HIPPIE	$1.89 \cdot 10^8$	$2.17 \cdot 10^7$ (≈8.7x)	$1.81 \cdot 10^8$
Biogrid Human 2020	$3.32 \cdot 10^8$	$2.25 \cdot 10^7$ (≈15x)	$5.40 \cdot 10^7$ (≈6.1x)
Enron Email	$6.73 \cdot 10^8$	$2.36 \cdot 10^7$ (≈28x)	$6.69 \cdot 10^6$ (≈100x)
Deezer-Romania	$8.72 \cdot 10^8$	$2.40 \cdot 10^7$ (≈36x)	$2.88 \cdot 10^6$ (≈300x)
Deezer-Hungary	$1.13 \cdot 10^9$	$2.44 \cdot 10^7$ (≈46x)	$8.28 \cdot 10^6$ (≈140x)
Deezer-Croatia	$1.49 \cdot 10^9$	$2.49 \cdot 10^7$ (≈60x)	$3.77 \cdot 10^7$ (≈39x)
Twitch Gamers	$1.41 \cdot 10^{10}$	$2.89 \cdot 10^7$ (≈490x)	$3.32 \cdot 10^9$ (≈4.3x)
PTMcode	$1.83 \cdot 10^{10}$	$2.94 \cdot 10^7$ (≈620x)	$4.60 \cdot 10^7$ (≈400x)
EU Email	$3.52 \cdot 10^{10}$	$3.07 \cdot 10^7$ (≈1100x)	$7.04 \cdot 10^6$ (≈5000x)
Amazon	$5.61 \cdot 10^{10}$	$3.17 \cdot 10^7$ (≈1800x)	$3.89 \cdot 10^7$ (≈1400x)

Table 3.1. **Suggested Negative Edges.** Maximum edges in the network, suggested number of negative edges for $\gamma = 10^{-4}$, and suggested number of negative edges for $\delta = 10^{-1}$. The numbers inside the parenthesis indicate the reduction rate in the computational resources required (i.e. the max number of edges divided by the suggested number of edges for bounded absolute or relative errors).

3.4.3 Investigating the Suggested Values on Additional Link Prediction Methods

Figure 3.12 shows the absolute and relative errors associated with using Adamic Adar as the link prediction method. Compared to the results for preferential attachment, the error values are significantly lower than their expected γ and δ values. This corroborates what is seen in Figure 3.3a. The error there is lower than the error seen in Figure 3.1d. Because of this, we say that the error estimates are accurate

only with respect to the specific method being used. So, for each method of link prediction, different α and β values must be used.



(a) (b)

Figure 3.12. **Adamic Adar Absolute and Relative Error Predictions** (3.12a) Absolute error vs number of nodes for Adamic Adar as the link prediction method for γ values of 10^{-4}, 10^{-3}, and 10^{-2}. (3.12b) Relative error vs number of nodes for Adamic Adar as the link prediction method for δ value of 10^{-1}

4 Limitations and Future Work

4.1 Mathematical Proof

This set of experiments shows that negative sampling during link prediction can be performed without greatly increasing the error associated with the prediction method. This empirical demonstration is sufficient for this small set of networks, methods, and metric, but it would be very useful to mathematically prove that the error is bound by the number of negatives sampled and not the ratio.

4.2 Refining the Error Prediction Equations

The error prediction proved to be quite accurate in bounding the relative and absolute errors. The derivation of these equations was done using the results of the link prediction experiments run on the networks in Table 2.1. Because of this, they are relatively crude and not proven to work in the general case. So, it is very important that a refined equation is derived. It was also shown that the error prediction was only specific to the link prediction method which the regression was performed on. So, it would be useful to determine how the prediction equations differ from method to method. Understanding this may give insight into a general equation that can be used regardless of the method being used, potentially using the method as a parameter.

4.3 Additional Methods

It has been proven that negative sampling during link prediction does not drastically increase the error, but does decrease the run-time and space required for five methods of link prediction. While these results are promising, there are still many areas of link prediction that could be visited. There are many more topology-based methods whose results drastically depend on the structure of the network. As mentioned before, in this research, we only used neighbor-based topology methods. As there are also path-based and random walk-based, we would like to explore how error is affected by different types of prediction methods. In addition to looking at topology-based methods, Node similarity methods should be visited as well. Node similarity methods construct feature sets based off the likeness of each node's attributes. Examples of similarity methods are Euclidean distance which uses the shortest path length between two nodes, Pearson coefficient which uses the covariance of the common neighbors between two nodes, and Cosine similarity which computes the cosine between two feature vectors to predict edges. There are also many more embedding, probabilistic, and Neural Network based link prediction methods that should be visited.

4.4 Evaluation Metrics

As mentioned before, AUROC was the only metric used during these experiments. This was chosen because negative sampling results in a bias in some metrics that cannot be easily quantified and accounted for when making comparisons between sampling and train-test error.

In order to more comprehensibly understand how negative sampling affects link prediction, it will be important to determine how to unbias these metrics.

5 Conclusions

From this work, we conclude that negative sampling can be performed in order to reduce the runtime and space needed to perform link prediction without sacrificing the performance of the link prediction method. In essence, we say the combined error associated with sampling and train-test splits is similar to the error associated with just train-test splits. These results were performed on 18 networks and generated using Preferential Attachment, Adamic Adar, Jaccard Similarity Coefficient, and Node2Vec as link prediction methods. These results apply to AUC as a metric, but we expect them to extend to other metrics that are also unbiased after negative sampling is performed as well.

We also proposed a model that suggests the number of negative edges needed to be sampled in order to limit the absolute or relative error. This model applies to preferential attachment, but we expect that a similar model could be constructed for other link prediction methods.

References

[1] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[2] Gregorio Alanis-Lobato, Miguel A Andrade-Navarro, and Martin H Schaefer. Hippie v2. 0: enhancing meaningfulness and reliability of protein–protein interaction networks. *Nucleic acids research*, page gkw985, 2016.

[3] Albert-Laszlo Barabâsi, Hawoong Jeong, Zoltan Néda, Erzsebet Ravasz, Andras Schubert, and Tamas Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical mechanics and its applications*, 311(3-4): 590–614, 2002.

[4] Mustafa Bilgic, Galileo Mark Namata, and Lise Getoor. Combining collective classification and link prediction. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, pages 381–386. IEEE, 2007.

[5] Ara Cho, Junha Shin, Sohyun Hwang, Chanyoung Kim, Hongseok Shim, Hyojin Kim, Hanhae Kim, and Insuk Lee. Wormnet v3: a network-assisted hypothesis-generating server for caenorhabditis elegans. *Nucleic acids research*, 42(W1):W76–W82, 2014.

[6] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[7] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[8] Peter V Hornbeck, Bin Zhang, Beth Murray, Jon M Kornhauser, Vaughan Latham, and Elzbieta Skrzypek. Phosphositeplus, 2014: mutations, ptms and recalibrations. *Nucleic acids research*, 43(D1):D512–D520, 2015.

[9] Md Kamrul Islam, Sabeur Aridhi, and Malika 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*, pages 81–92, 2021.

[10] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. Conditional network embeddings. *arXiv preprint arXiv:1805.07544*, 2018.

[11] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

[12] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[13] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[14] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 243–252, 2010.

[15] Francois Lorrain and Harrison C White. Structural equivalence of individuals in social networks. *The Journal of mathematical sociology*, 1(1):49–80, 1971.

[16] Pablo Minguez, Ivica Letunic, Luca Parca, Luz Garcia-Alonso, Joaquin Dopazo, Jaime Huerta-Cepas, and Peer Bork. Ptmcode v2: a resource for functional associations of post-translational modifications within and between proteins. *Nucleic acids research*, 43(D1):D494–D502, 2015.

[17] Rushabh Patel, Yanhui Guo, Adi Alhudhaif, Fayadh Alenezi, Sara A Althubiti, and Kemal Polat. Graph-based link prediction between human phenotypes and genes. *Mathematical Problems in Engineering*, 2022, 2021.

[18] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623732. URL http://doi.acm.org/10.1145/2623330.2623732.

[19] Benedek Rozemberczki and Rik Sarkar. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, page 1325–1334. ACM, 2020.

[20] Benedek Rozemberczki and Rik Sarkar. Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings, 2021.

[21] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*, pages 65–72. ACM, 2019.

[22] Jerry Scripps, Pang-Ning Tan, Feilong Chen, and Abdol-Hossein Esfahanian. A matrix alignment approach for link prediction. In *2008 19th international conference on pattern recognition*, pages 1–4. IEEE, 2008.

[23] Naoki Shibata, Yuya Kajikawa, and Ichiro Sakata. Link prediction in citation networks. *Journal of the American society for information science and technology*, 63(1):78–85, 2012.

[24] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic acids research*, 34(suppl_1):D535–D539, 2006.

[25] Magdalena Szumilas. Explaining odds ratios. *Journal of the Canadian academy of child and adolescent psychiatry*, 19(3):227, 2010.

[26] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*. ACM, 2015.

[27] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, August 2009.

[28] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi. Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1100–1108, 2011.

[29] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. Link prediction in social networks: the state-of-the-art. *Science China Information Sciences*, 58 (1):1–38, 2015.

[30] Eric W Weisstein. Handshake problem. URL https://mathworld.wolfram.com/HandshakeProblem.html.

[31] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.

[32] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[33] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, 45(3):751–782, 2015.

[34] Y Connie Yuan and Geri Gay. Homophily of network ties and bonding and bridging social capital in computer-mediated distributed teams. *Journal of computer-mediated communication*, 11(4):1062–1084, 2006.

[35] Jing Zhao, Lili Miao, Jian Yang, Haiyang Fang, Qian-Ming Zhang, Min Nie, Petter Holme, and Tao Zhou. Prediction of links and weights in networks by reliable routes. *Scientific reports*, 5(1):1–15, 2015.

[36] Tao Zhou. Progresses and challenges in link prediction. *Iscience*, 24(11): 103217, 2021.