Machine Learning and Edge Prediction

Michael Ramsey

December 17, 2018

Abstract

Given a snapshot of a network, can we infer the existence of missing interactions in the network? This is the problem statement for the edge prediction problem. In this paper, we explore current edge prediction methods in literature to predict edges in an early snapshot of the facebook100 networks, the hep-PH citation network, and the Enron email network. Overall, the methods based on machine learning performed the best in this study.

1 Introduction

With the recent surge in machine learning and big data, network science has benefited as well. The emergence of large-scale social networks such as facebook, twitter, and spotify have jump-started the research in the field. One of the major goals of these networks is to recommend new connections. This is the main motivation for the edge prediction problem, which can be simplified as follows. Given a network $G = \langle V, E \rangle$, with vertices V and edges set E, can we infer the existence of missing links from our observed network? [3]

There are several useful applications of these edge prediction methods. Probably the most common application, and probably one that you thought of, are recommendation systems. In the context of a social network, our goal is to recommend "friends" to people. There are many other possible applications. Post 9-11, there was a surge of research in edge-prediction with the goal of monitoring terrorist networks. [3] Based on which terrorists we know are communicating, we can try and predict other terrorists that may also be involved with the terrorist groups. For citation networks, we can recommend additional papers that may be relevant to yours. Peng et al. describes several other edge prediction applications in bioinformatics to find protein-protein interaction in gene expression networks. I believe that it is the numerous applications that has driven the research in edge-prediction.

In this paper, I will be analyzing three different networks: a subset of 30 from the facebook100 dataset (actually 30 different networks), the Enron network dataset, and the hep-Ph citation network. For the facebook and hep-Ph datasets, our goal is to recommend new friends/papers. For the Enron dataset, our goal is more along the lines of terrorist monitoring networks. Using the network, our goal is to predict who was involved with the scandal, based on who we know is currently involved. I describe more of this later in the procedure section.

Currently, there are two main approaches for edge prediction: (1) Similarity-based approaches and (2) Learning-based approaches. For similarity-based approaches, we have the following procedure [1]:

1. Compute features for every non-edge pair.
2. Order the scores of that feature for every non-edge pair.
3. Predict that the top arbitrary percentage of non-edge pairs are connected.

For learning-based approaches, we apply a probabilistic model or a classifier to predict edges. In other words, we view the edge prediction problem as a binary classification problem. Not only can we use the topology of the network as features to classify node pairs, but we can also include meta-data into the models for added features. For these types of methods, we have the following outline [1]:

1. Compute features for every edge and non-edge pair.
2. Train your probabilistic/classifier model.
3. Predict edges via your probabilistic/classifier model.

2 Procedure

For the facebook100 dataset, I decided to analyze only a subset of 30. The 30 networks that I analyzed were the smallest in size networks out of the 100. This choice was made to make sure that I didn't run into any computation or memory storage problems with my laptop.

For the 30 networks, I split each edge set into a training set and a validation set by uniform sampling. I generated the network only for those edges in the training set. Since I removed edges from the network, this will change the structure and topology of the network. We know from class that there are issues associated with uniform sampling of edges in a network, including sparsity and bias in degree distribution. A large part of the paper will be investigating the performances of our edge prediction methods based on the proportion of edges that are included in the training set. For all edges in the edge set, I computed all of the features listed in Section 2 with respect to the network generated by the training set. The training set percentages that I chose to include in this report are 30%, 50%, 70% 90% and 97%. The remaining edges were placed in the validation set. All reported prediction accuracies are based on performance of the validation set.

We note here that generally for social networks, the total number of non-edges in the network is much larger than the total number of edges in the network. For example, in the snapshot for "American University", for every 1 edge that exists in the network, there are 46 non-edges that don't exist. This data-imbalance would cause our learning-based algorithms to have skewed results. Rather than using the whole non-edge set to train the models, I decided to uniformly sample from the non-edge set, so that the distribution of edges to non-edges is approximately 1-1. This procedure made sure that my data was balanced. The non-edges were also split in the same proportions as the sampled edges.

Once I obtained my feature set for all edges and non-edges, I implemented the similarity-based prediction methods and the learning-based prediction methods. The hep-PH dataset was handled in a similar manner as the facebook30 dataset. For the Enron email networks, I have time stamps of every interaction. Rather than sampling uniformly from the edge set to construct a validation set, our goal is to predict future edges in the network. More will be discussed on this later.

3 Feature Selection

In order to predict edges in the network, we must be able to identify measures that distinguish between nodes that are connected and nodes that are not connected. Several different kinds of features have been used in the edge prediction literature. Overall, we can group these features in the following categories: neighbor-based features, node-based features, path-based features, random-walk based features, and metadata-based features. I describe several of these features that I used in my edge prediction models.

3.1 Neighbor-based Features:

We make the assumption that nodes are more likely to connect with nodes that have similar neighbors. To establish notation, let $\Gamma(x)$ be the set of neighbors for node x, and let $|\Gamma(x)|$ denote the number of neighbors for node x.

- **Neighbor Sum (NS):** [2] A naive feature, defined as the number of neighbors for node x plus the number of neighbors for node y. That is

$$\mathbf{NS}(x, y) = |\Gamma(x)| + |\Gamma(y)| \tag{1}$$

- **Common Neighbors:** [2] For two nodes x and y, we define the common neighbors metric as the number of common neighbors for nodes x and y. That is

$$\mathbf{CN}(x, y) = |\Gamma(x) \cap \Gamma(y)| \tag{2}$$

- **Preferential Attachment (PA):** [1] We make the assumption that new nodes are more likely to connect to high degree nodes than low degree ones.

$$\mathbf{PA}(x, y) = |\Gamma(x)| \cdot |\Gamma(y)| \tag{3}$$

- **Sorensen Index (SI):** [1] A similar measure to common neighbors, but we normalize by $|\Gamma(x)| + |\Gamma(y)|$ to put more weight on nodes with fewer connections. We have

$$\mathbf{SI}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)| + |\Gamma(y)|} \tag{4}$$

- **Salton Cosine Similarity (SC):** [1] Another similarity metric similar to the Sorensen Index.

$$\mathbf{SC}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{\sqrt{|\Gamma(x)| \cdot |\Gamma(y)|}} \tag{5}$$

- **Hub Promoted (HP):** [1] Defines the topological overlap between two nodes.

$$\mathbf{HP}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{min(|\Gamma(x)|, |\Gamma(y)|)} \tag{6}$$

- **Hub Depressed (HD):** [1] Defined similarly to (HP),

$$\mathbf{HD}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{max(|\Gamma(x)|, |\Gamma(y)|)} \tag{7}$$

- **Resource Allocation (RA):** [1] An additional metric that measures similarity based on the number of neighbors, but also penalizes high degree nodes.

$$\mathbf{RA}(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{|\Gamma(z)|} \tag{8}$$

3.2 Node-based Features:

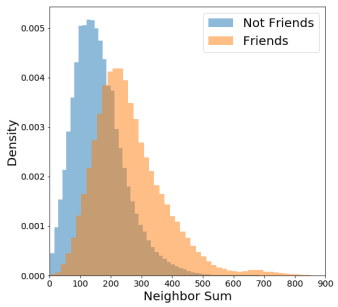
These are features that depend on certain attributes of the node. For each node pair, we compute

- **Local Clustering Coefficient Sum (CS):** [2] For nodes x and y, let $C(x)$ and $C(y)$ denote the local clustering coefficient for nodes x and y respectively. Then this feature is simply

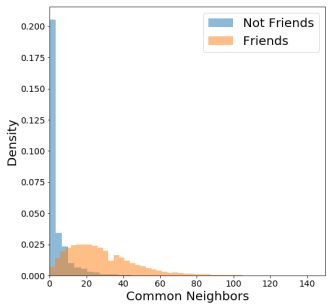
$$\mathbf{CS}(x, y) = C(x) + C(y) \tag{9}$$

- **Local Clustering Coefficient Product (CP):** [2] Like feature (CS), we define it as,

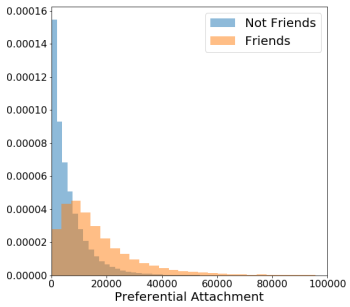
$$\mathbf{CS}(x, y) = C(x) \cdot C(y) \tag{10}$$



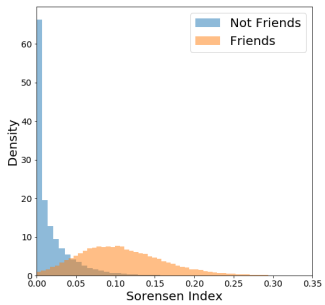
(a) Neighbor Sum



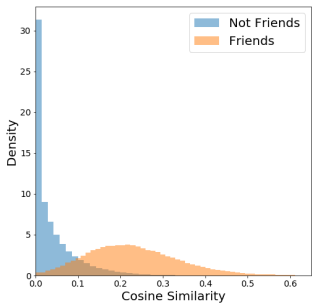
(b) Common Neighbors



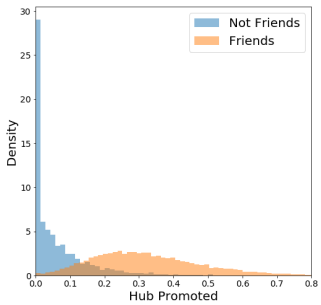
(c) Preferential Attachment



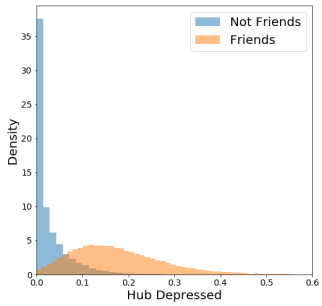
(d) Sorensen Index



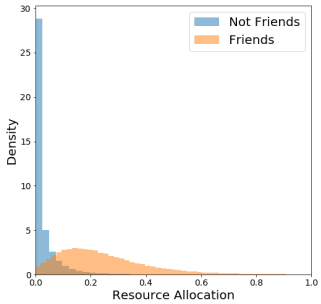
(e) Salton Cosine Similarity



(f) Hub Promoted

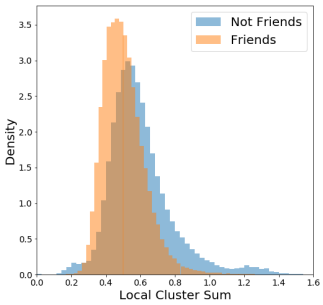


(g) Hub Depressed

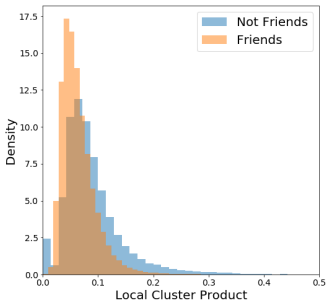


(h) Resource Allocation

Figure 1: Distributions of neighbor-based features.



(a) Cluster Coefficient Sum



(b) Cluster Coefficient Product

Figure 2: Distributions of node-based features.

3.3 Path-based Features:

Rather than using information specific to the nodes, we can use path information from each pair of nodes to compute features. These kind of features are not as trivial to compute as the node-based features. Many of them require forming matrix factorizations and inverses of the adjacency matrix. For the sake of time, I decided to implement one feature.

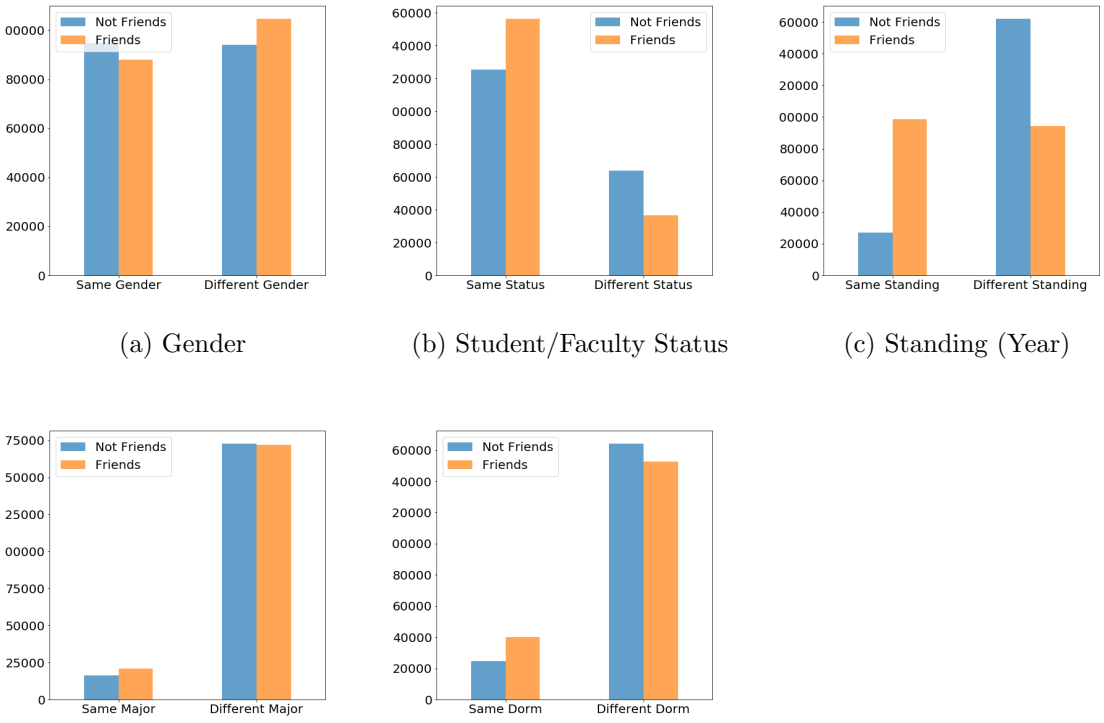
- **Local Path (LP):** For nodes x and y, we define the (LP) of x and y as

$$\mathbf{LP}(x, y) = A^2(x, y) + \alpha A^3(x, y) \tag{11}$$

where α is a tuning parameter. This metric considers all node-pairs that are length 2 away and also puts partial weight on length 3 paths. I considered 3 different values of $\alpha = .1, .01, .001$.

3.4 Network-Specific Features

For the Facebook dataset, we are given additional attributes for the nodes. Specifically for each node, we know gender, student/faculty status, academic year, major, and dorm. I did not expect gender and major to have a large effect on the model.



(a) Gender (b) Student/Faculty Status (c) Standing (Year)

(d) Major (e) Dorm

Figure 3: Distributions of meta-data features for facebook30 data.

3.5 Other Features

Another category of metrics are random-walk based metrics. The motivation for these features comes from the idea that social interactions can be modeled by a random walk in the network. [1] I chose not to include these features in my model because many of these features require matrix decomposition's and factorizations. With networks of this scale, I would have to be careful with the numerics of these computations, should I implement them in the future.

4 Results

4.1 Similarity-based Results

After I obtained the feature set for the networks, I ordered the validation set by each feature, and took the top 50% of node-pairs to predict as edges. Keep in mind that in this validation set, there is approximately a 1-1 ratio of edges to non-edges in the validation set. So if the similarity score is performing perfectly for predicting edges, then it should predict that the top 50% of node-pairs are edges correctly. I display these prediction results in Tables 2 and 3 for the facebook30 networks and the hep-Ph network.

Feature	Acc 30	Acc 50	Acc 70	Acc 90	Acc 97
Common Neighbors	0.8789	0.8726	0.8790	0.8816	0.8814
Preferential Attachment	0.7219	0.7213	0.7221	0.7232	0.7238
Neighbor Sum	0.7023	0.7016	0.7023	0.7027	0.7034
Sorensen Index	0.8846	0.8759	0.8846	0.8880	0.8891
Cosine Similarity	0.8877	0.8799	0.8876	0.8900	0.8909
Hub Promoted	0.8767	0.8707	0.8766	0.8782	0.8793
Hub Depressed	0.8779	0.8698	0.8780	0.8809	0.8812
Resource Allocation	0.8955	0.8873	0.8957	0.8988	0.8983
Local Cluster Sum	0.4014	0.4124	0.4005	0.3970	0.3978
Local Cluster Product	0.4085	0.4201	0.4076	0.4039	0.4047
Local Path - .001	0.8775	0.8730	0.8776	0.8798	0.8798
Local Path - .01	0.8775	0.8730	0.8776	0.8798	0.8798
Local Path - .1	0.8775	0.8730	0.8776	0.8798	0.8798

Table 1: Similarity measure results for each feature for the facebook30 networks. Columns two through 6 contain the prediction accuracies for each feature, with the column indicating the percentage of edges that were in the network.

We see that the two features involving clustering coefficients perform poorly in the facebook30 networks. We also see that the preferential attachment and neighbor sum features don't perform as well as the remaining features. This is some evidence to suggest that the facebook30 networks don't form edges via a preferential attachment mechanism (Recall that preferential attachment is the process by which new edges form with higher probability on nodes that already have high degree). The features that performed the best are the ones involving common neighbors. One additional interesting effect is that the percentage of the network that the graph was trained on does not seem to effect the accuracies in a large way. It seems that even when the graph is created with only 30% of the edges, we get good edge prediction accuracy. We see similar trends for the hep-PH network in Table 2.

For the Enron email data, I decided to perform edge prediction differently. December 2, 2001 is the date that Enron filed for bankruptcy. The criminal investigation of Enron began on January 2, 2002. My goal here was to predict the future associations of the emails, knowing the previous email interactions of the Enron employees. For example in Table 3, The January column indicates prediction accuracy for all future interactions of the Enron employees, knowing all the information up to but not including January 2002. Similarly, the February column indicates prediction accuracy for all future interactions of the Enron employees, knowing all the information up to but not including February 2002.

The prediction accuracy is not as high as in the facebook30 or hep-PH networks. It is possible that after the criminal investigation began, the distribution of communications between the Enron employees changed. Some may be paranoid and "go dark", while other individuals may start communicating when the previously haven't. Again, we see that the features involving the clustering coefficients perform poorly compared to the neighbor-based features. All 3 local path features also performed well.

Feature	Acc 30	Acc 50	Acc 70	Acc 90	Acc 97
Common Neighbors	0.9968	0.9929	0.9885	0.9843	0.9824
Preferential Attachment	0.7387	0.7475	0.7502	0.7525	0.7528
Neighbor Sum	0.7243	0.7292	0.7319	0.7331	0.7334
Sorensen Index	0.9968	0.9929	0.9885	0.9843	0.9824
Cosine Similarity	0.9964	0.9925	0.9883	0.9842	0.9823
Hub Promoted	0.9964	0.9925	0.9883	0.9842	0.9823
Hub Depressed	0.9968	0.9929	0.9885	0.9843	0.9824
Resource Allocation	0.9968	0.9929	0.9885	0.9843	0.9824
Local Cluster Sum	0.5440	0.4824	0.4368	0.4158	0.4057
Local Cluster Product	0.7774	0.5673	0.4806	0.4370	0.4229
Local Path - .001	0.9724	0.9361	0.9511	0.9597	0.9633
Local Path - .01	0.9724	0.9361	0.9511	0.9597	0.9633
Local Path - .1	0.9724	0.9361	0.9511	0.9597	0.9633

Table 2: Similarity measure results for each feature for the hep-PH network. Columns two through 6 contain the prediction accuracies for each feature, dependent on the percentage of edges that are provided to the network.

Feature	January	February	March	April	May
Common Neighbors	0.7546	0.8000	0.7656	0.7132	0.6786
Preferential Attachment	0.6044	0.6429	0.6302	0.6765	0.6071
Neighbor Sum	0.6190	0.6536	0.6458	0.6985	0.6071
Sorensen Index	0.7216	0.7986	0.7656	0.6838	0.6786
Cosine Similarity	0.7269	0.7760	0.7527	0.6953	0.6667
Hub Promoted	0.7538	0.7760	0.7527	0.7109	0.6667
Hub Depressed	0.6996	0.7986	0.7656	0.6985	0.6429
Resource Allocation	0.7070	0.8000	0.7656	0.7059	0.7143
Local Cluster Sum	0.4762	0.5036	0.4427	0.4485	0.6429
Local Cluster Product	0.4725	0.5357	0.4792	0.4706	0.5714
Local Path - .001	0.7216	0.7429	0.7448	0.6985	0.6429
Local Path - .01	0.7216	0.7429	0.7448	0.6985	0.6429
Local Path - .1	0.7216	0.7429	0.7448	0.6985	0.6429

Table 3: Similarity measure results for each feature for the Enron email network. Columns two through 6 contain the prediction accuracies for each feature, where the network was given all edges up to but not including that month, 2002.

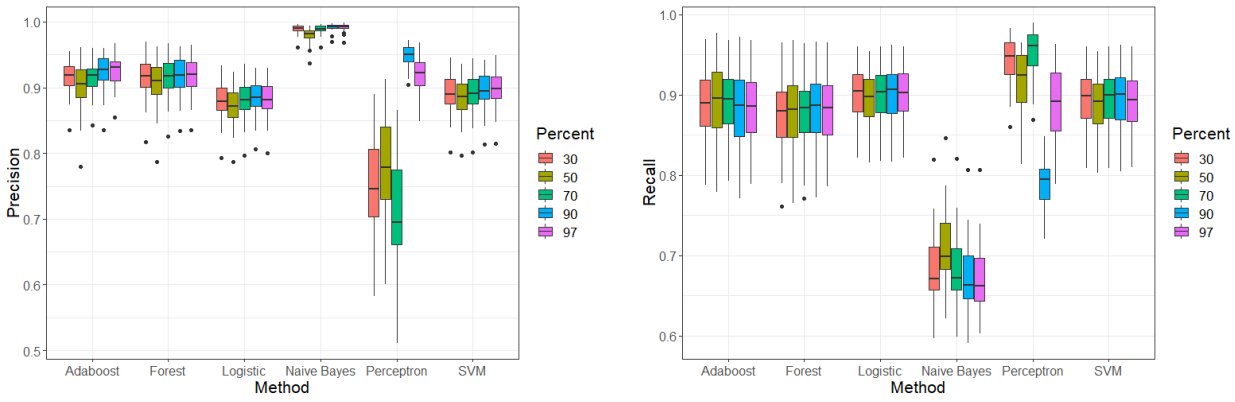
4.2 Learning-based Results

One drawback from the similarity-based methods is that you can only use one piece of data to predict edges. We see from the similarity results that many features do a good job at predicting edges. We turn to machine learning, and model the edge-prediction problem as a binary classification problem. For this analysis, I chose to implement logistic regression, support vector machine (SVM), multi-layer perceptron, random forest, adaboost, and naive bayes.

For the facebook30 networks, I was able to combine the edge-pairs for all schools into one model. The prediction accuracies for each school were all comparable. This tells me that the facebook30 networks have similar distributions of edges.

Recall is the proportion of true positives to true positives and false negatives. A high recall score indicates that our classifier is doing a good job at identifying pairs of nodes that share an edge. We want a high recall score in our classifier. Precision is the proportion of true positives to true positives and false positives. A low precision means that many pairs of nodes that did not share an edge, were predicted as sharing an edge. I'm going to be satisfied with a lower precision score in these models. A false positive can be used as a connection recommendation tool. An edge may not actually exist in the network, but the model thinks it should.

In Figure 4, I display boxplots for the precision and recall scores of each classifier of the facebook30 networks. I chose to look at the precision and recall scores individually for each school. Each boxplot is comprised of 30 points, one for each school.



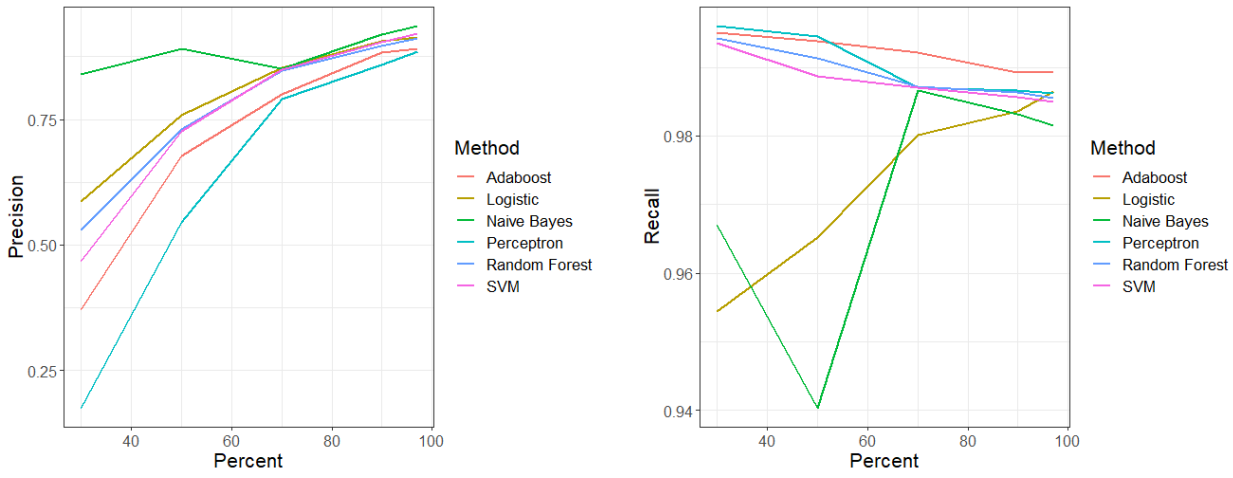
(a) Facebook Precision

(b) Facebook Recall

Figure 4: On the left, we have the precision scores for each classifier, as we vary the proportion of edges in the training set. Percent "97" corresponds to 97% of total edges are in the training set.

Surprisingly, the proportion of edges in the training set does not seem to effect the precision and recall scores. We can see that that model with the lowest precision is the perceptron model and it is also the model with the highest recall overall. Because I am looking for a model with lower precision and higher recall, I would say that the perceptron model is the best model for the facebook30 networks. The logistic regression, random forest, SVM, and adaboost classifiers also performed very well. The naive bayes model performed the worst out of them all.

Next, we look at the performance of the learning-based algorithms on the hep-PH citation network. I display the precision and recall scores for each classifier as a function of the percent of edges in the training set in Figure 5. We see in Figure 5(a), that as the percentage of edges in the training set increases, the precision scores for all classifiers generally increases. Interestingly in Figure 5(b), as the proportion of edges in the training set increases, the overall recall scores go down. For the hep-PH network, I like the performance of the adaboost classifier. It gives us one of the highest recall scores among the classifiers and also has a lower precision, allowing for extra edge predictions.



(a) hep-PH Precision

(b) hep-PH Recall

Figure 5: On the left, we have the precision scores for each classifier, as we vary the proportion of edges in the training set. Percent "97" corresponds to 97% of total edges are in the training set. Each classifier is represented by a different color line.

Next we discuss the performance of the learning-based methods on the Enron email network. Overall, the performance of the machine learning methods were poor. The only method that provided reasonable performance results was the naive bayes model. I provide a table of the accuracy, precision, and recall scores for the Naive Bayes classifier in Table 4. The columns of Table 4 have the same meaning as the similarity results in Table 3. The enron network was much smaller than the facebook30 and hep-PH networks. It is possible that there is not a large enough sample of edges to make the machine learning methods perform well. Generally naive bayes often works better on smaller datasets, like it does in this case. However, we next address other possible failures of modeling in the next sections.

	January	February	March	April	May
Accuracy	0.6840	0.7246	0.7198	0.6784	0.6481
Precision	0.9081	0.6870	0.6919	0.7185	0.8214
Recall	0.6399	0.7627	0.7399	0.6879	0.6216

Table 4: Performance of the Naive Bayes classifier as we vary the time point of edges known.

5 Discussion

Overall both the similarity-based methods and the learning-based methods performed well on the facebook30 and the hep-PH networks. I prefer use of the learning-based predictions over the similarity-based predictions for two reasons: (1) We are allowed to include meta-data into the learning-based models. (2) We get more flexibility in modeling by including multiple features in the model. Additionally, I would recommend using the false positive predictions of the learning-based models to be used as "connection recommendation" for those two nodes.

For the Enron email network, our predictions were poor. The similarity-based methods performed better than most of the learning-based methods. I previously addressed that one possible reason for the poor predictions could be due to an edge distribution change after the Enron filed for bankruptcy in December 2001. Another possible reason for poor predictions could be due to a relatively small number of edges. However, I think the most likely reason for the prediction deficiency lies with the shortcomings of the learning-based methods.

6 Future Work

The main downfall of the methods that I performed is that we assume that the networks are static. In reality, most networks are dynamic; they are constantly changing over time. In order to better predict edges for the Enron email network, I think we need to model the edges dynamically. This is the next step of edge prediction methods, and currently is being the one that is most researched. One possible technique that I would like to research is an evolving latent space model. [5] Using the framework of a recurrent neural network, we can model both the appearance and disappearance of edges in the network dynamically. I ran out of time and was unable to implement this method, however I believe it to be promising.

References

[1] W. Peng, X. BaoWen, W. YuRong, Z. XiaoYu, *Link Prediction in Social Networks: The State-of-the-Art*, (Science China January 2015, Vol. 58), available at `https://arxiv.org/pdf/1411.5118.pdf`.

[2] M. Hasan, V. Chaoji, S. Salem, M. Zaki, *Link Prediction Using Supervised Learning*, available at `https://archive.siam.org/meetings/sdm06/workproceed/Link%20Analysis/12.pdf`.

[3] D. Liben-Nowell, J. Kleinberg, *The link Prediction Problem for Social Networks*, available at `https://www.cs.cornell.edu/home/kleinber/link-pred.pdf`.

[4] L. Yao, L. Wang, L. Pan, Kai Yao, *Link Prediction Based on Common-Neighbors for Dynamic Social Network*, (ScienceDirect, Procedia Computer Science 83 (2016) 82 89), available at `https://www.sciencedirect.com/science/article/pii/S1877050916301259`.

[5] S. Gupta, G. Sharma, A. Dukkipati, *Evolving Latent Space Model for Dynamic Networks*, available at `https://arxiv.org/pdf/1802.03725.pdf`.