Social Network Analysis: Supervised Link Prediction

Qi Ying Lim, Jason Wang, Ruifan Yang

December 9, 2017

Abstract

Given knowledge of co-authorships between academics in the past, we want to try to predict future co-authorships. In our project, we extract 6 different features from the social network that we believe will influence prospective co-authorships between academics. By comparing the performance of these features in the positive and negative example, we derive an initial insight on the ranking of these features for our link prediction problem. Next, feed these features into a binary classifier that produce the main results for our task. We use five different supervised learning algorithms as our classifier, and then compared their results to see which performed the best. The five used are Random Forest, Support Vector Machine, Ababoost Algorithm, K Nearest Neighbors, and Feedforward Neural Network. From the Random Forest model we were also able to extract an actual ranking on these features according to their prediction ability. We saw that the total accuracy for the five models were very similar, approximately 98%. However, the Abadoost algorithm was significantly the best in predicting the true positives. On the other hand, Neural network had the best performance overall, with a high total accuracy and a much lower running time compared to all the other algorithms.

Keywords— Link Prediction, Artificial Intelligence, Supervised Learning, Machine Learning, Binary Classification, Features Selection, Similarity Features

1 Introduction and Prior Work

1.1 The Link Prediction Problem

Consider the snapshot of a social network at time t represented by a graph $G = (V, E)$ where V and E are sets of vertices and edges respectively. The goal of the link prediction problem is to predict how the social network evolves, more specifically, the appearances of new edges between two vertices at a future time t', where $t' > t$.

Our project is inspired by a previous research done by Hasan et al. In their research, they applied the link prediction problem to the realm of academia, specifically, computer science, where a co-authorship graph represents the social network. In this graph, each vertex corresponds to an author, and an edge between two vertices means that the two authors have co-authored an academic paper before some time t. We implement their methodology on a different data set – papers written in the field of 'fluid dynamics'. We found that restricting our dataset to a more specific and obscure academic field made data processing and feature extraction a more time efficient task.

We extract features from this graph, and use these features for various types of supervised learning techniques for building binary classifier models, to predict prospective links between authors.

1.2 Why should we care?

There are many practical problems that fall into the category of the link prediction problem. For the project we studied (Hasan, 2006), the authors were concerned with applying their link prediction techniques to the problem of analyzing terrorist neworks, as link prediction techniques present us with the tools to anticipate future activities by terrorist groups. Another application of this problem would be in advertisement recomendations. Given knowledge of present interactions of users with advertisements on a website, we want to be able to measure the probability that a user clicks on an advertisement in the future. The higher the probability, the more the particular advertisement should appear on the user's browser.

2 Dataset

We were able to use the Microsoft Academic Graph API (MAG) to get all the necessary data for our study. We experimented with different datasets corresponding to different fields, and found that the academic network falling in the discpline of fluid dynamics was sufficiently large for meaningful analysis, and yet sufficiently compact so that our implementation was doable on our personal computers.

2.1 Data Processing

The data returned by MAG was in a JSON file format, where each entitity represents a paper containing the term 'fluid dynamics' in its field list. For each entity, we were given:

- the title of the paper
- the list of authors who worked on the paper
- the list of keywords associated with the paper
- the list of fields associated with the paper
- the year in which the paper was published

From this file we generated three separate datasets. The first was our vertex set, simply a list of all the authors present. The second one was a list of nested tuples: ((author1, author2) year), which represented our graph's edge set. We placed a pair of authors into a tuple if their names had appeared together on some paper, and included the year in which the particular paper was published. The last dataset included all the necessary attributes to compute our feature set. This was in the form of a json file:

```
{ author name:
        {"num_collaborations": int
        "num_papers": int
        "keywords": list
        "fields": list
    }
}
```

With our vertex set and edge set, we generated a graph using the python library NetworkX. However, on visualizing the data, we saw that our graph had many disconnected components, so we chose to take the biggest connected component for our analysis. For this biggest connected component, Table lists its properties.

Table 1: Properties of our graph

Property	**Values**
Number of nodes	795
Number of edges	137799
IsConnected?	true
Average degree of Nodes	346.6541

3 Similarity Features

Most of our choice of features, excluding field match count, were taken from Hasan et al's study.

We first assign scores to each pair of nodes with no edge in the training set according to different similarity features. Then, we feed these similarity scores to a binary classifier to predict whether a link will occur in the future for each pair of node.

3.1 Proximity Features

This relates to a measure of how ”close” two nodes in our graph are to each other. We used two metrics in this feature.

Keyword Match Count For an author we were able to access a list of keywords used in the author's paper, that describe its main ideas. Between two authors, their 'keyword proximity' is length of the intersection of both author's keyword lists. The more keywords in the intersection would mean that the authors tend to study similar ideas and concepts, which makes them better candidates for future collaboration.

Field Match Count We were also able to access the list of fields, that is, the disciplines that the author's previous paper collectively fall into. Much similar to the keyword match count, the larger the intersection of fields between two authors, the more likely that their research interests are pointed in similar directions, increasingly likelihood of future collaboration.

3.2 Aggregated Features

These features relate to a single node in the graph. When we calculating the similarity scores, we use an aggregate function to convert two aggregate scores of single nodes to the score of a pair of nodes. Here, we use summation as our aggregation function. That is, $score(x, y) = aggregateScore(x) + aggregateScore(y)$.

Sum of Papers This feature measures how many papers an author has published in the training time period. This directly relate to how prolific an author is. Instinctively, two more prolific authors have a higher likelihood of co-authoring a paper in the future, compared to two authors who, perhaps, publish papers infrequently.

Sum of Neighbours This feature measures how likely in general an author is likely to collaborate. An author who likes to work alone, or in small groups will naturally have a lower rate, hence likelihood, of collaboration in the future.

Sum of Keywords Count This feature measures how many specific domain an author has worked in. If an author has a wide range of publications, he/she is more likely to work with new authors.

3.3 Topological Features

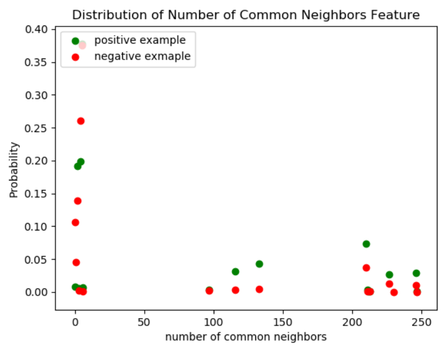
Here, we predict the future link based on the graph structure itself. According to the characteristics of the features, they can be divided into neighbor-based metrics, path-based metrics, and random-walk based metrics.

In the following section we will denote by $\Gamma(x)$ the neighbor set of node x, and $|\Gamma(x)|$ is the number of neighbors of node x.

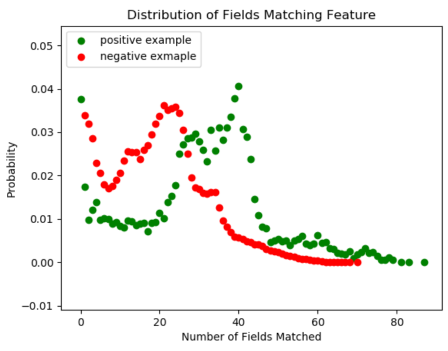
Common Neighbors The most straightforward measure of similarity between two nodes is to define $score(x, y) = |\Gamma(x) \cap \Gamma(y)|$, the number of common neighbors node x and node y have.

Jaccard's coefficient Jaccard's coefficient is a commonly used similarity metric measures the probability that both x and y have some feature given that either x or y has the feature. Here, we define $score(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$

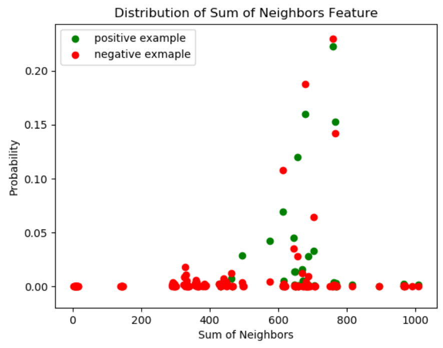
Shortest Path This metric computes the smallest number of steps required to walk from node x to node y in the training graph. If two authors are more "close" to each other in a graph, they are more likely to co-author together in the future. For this metric, we set the weights of each edge to be the reciprocal of the number of times the corresponding two authors it connects has co-authored a paper together.



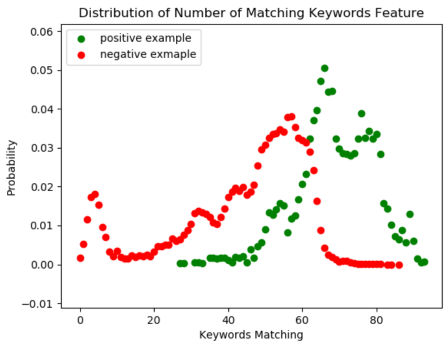
(a) Common Neighbours



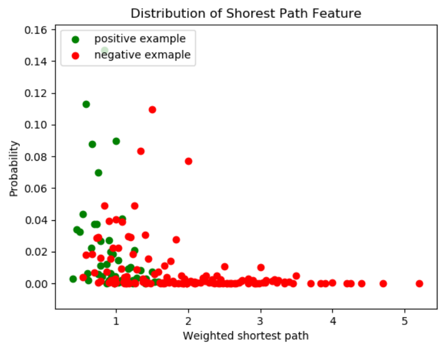
(b) Field Match Count



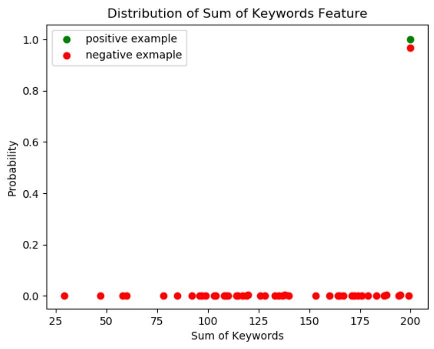
(c) Sum of Neighbours



(d) Keyword Match Count



(c) Weighted Shortest Path



(d) Sum of Keyword Count

Figure 1: Evaluation of Features

4 Feature Evaluation

Figure 1 shows the results for each feature. One of the more pronounced separations between the positive and negative examples is in the weighted shortest path feature. As seen in Figure 1(d),the positive example points congregate about the lower weighted shortest path values, as anticipated.

The other two features with a clear distinction are the field match and keyword match count. Figure 1(b) and Figure 1(d) show that the positive examples do tend do have higher field match and keyword match count as compared to the negative examples.

While not as clear as the previous 3 features, the plot 1(c) of the sum of neighbours feature does indicate that the positive examples seem to fall in the upper-end of the spectrum, that, authors who have had collaborated with more people before will in turn be more likely to collaborate in the future.

These are simply observations from the plots. To obtain a deeper and more meaningful insight on these features, we turn our implementation of several classifiers.

5 Classifiers

We randomly choose roughly half set of pairs in our training set for model training and the other half as testing. Since these two halves are independent, it's fine to split it that way. And the labels for each set is 0 if there is no edge in the testing set and 1 if there is edge. And all of the classifiers are implemented by scikit-learn v0.19.1 package.

5.1 Random Forest

The first model we used is random forest with max depth = 7, number of tree = 20, and minimum sample in the leafs = 20. The parameters are tuned with consideration of prevention of over-fitting, test error reduction and running time. Random forest is a ensemble method, voting the outputs of multiple small decision trees, which is able to reduced variance of our model. The construction of decision trees is based on Gini Index. A input vector is fed into each of the small decision trees, and then the output is the class that has most votes from the decision trees. One good thing about Random Forest is that it would not over-fit since it's doing out of bag sampling each time and use different bootstrap sample as training data to construct trees. Therefore, at the beginning, we used random forest to train the data with all our six features Moreover, we can utilize random forest to rank our six features by its gini importance. It is calculated by the summation of gini impurity reduction of each nodes split, (Louppe, 2017) and the result is shown in fig2
As you can see feature K, P, F, which are number of matching keywords, shortest path and number of matching fields, are features with top three importances, whereas CN, SK, and SN, which are number of common neighbors, sum of keywords and sum of neighbors are not importance in our classification model. And the result is actually consistent with the distributions of features in our dataset.
Next, we are using the top three features, K, P, F to train our various classifiers with standardized data. We standardized our features value to standard normal distribution with mean 0 and standard deviation 1 to rescale the range of feature values into the same.

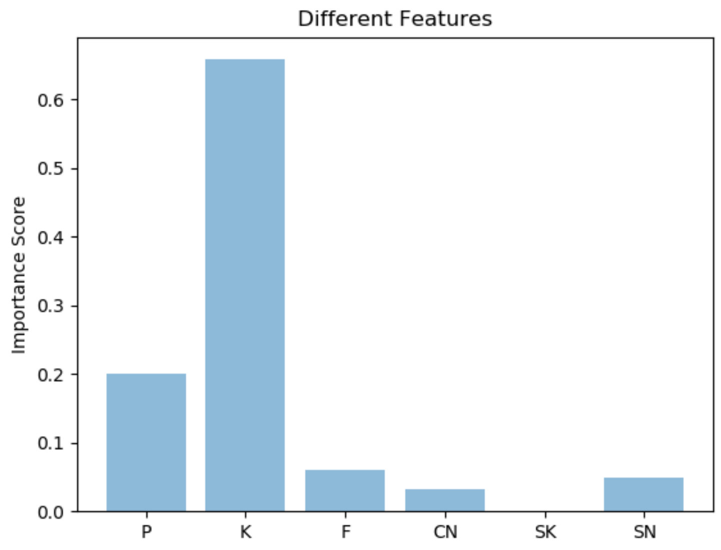


Figure 2: Ranking of features

The 3D visualization of the resulting dataset distribution with those three features is shown in figure 3. There is a clear separation between positive examples and negative example, which means that classifiers are able to draw decision boundary to correctly classify data points.

5.2 Support Vector Machine

Support Vector Machine is originally a linear classifier, which fits a line to separate the dataset with margin as big as possible. However, our dataset might not be separable in high dimensions, so we used the Gaussian kernel to detect any nonlinearity in our dataset.

5.3 Adaboost Algorithm

Adaboost is another powerful ensemble methods in classification problem. It starts with a weak classifier, for our case, a small decision tree. And then, it improves the models iteration by iteration by ensembling multiple weak classifier with modifications each time. In our model, we use decision tree with max depth = 3 as weak classifier and number of estimator = 15, to reduce running time.

5.4 K Nearest Neighbors

K Nearest Neighbors is a simple but useful algorithm. For our binary classification problem, it find the k nearest data point from the current data point, and choose the labels that majority of the neighbors have. After parameter tunning, we choose k equals to 3 in our algorithm. If the k is too small, it would over-fitting and if the k is too large, it would under-fitting.

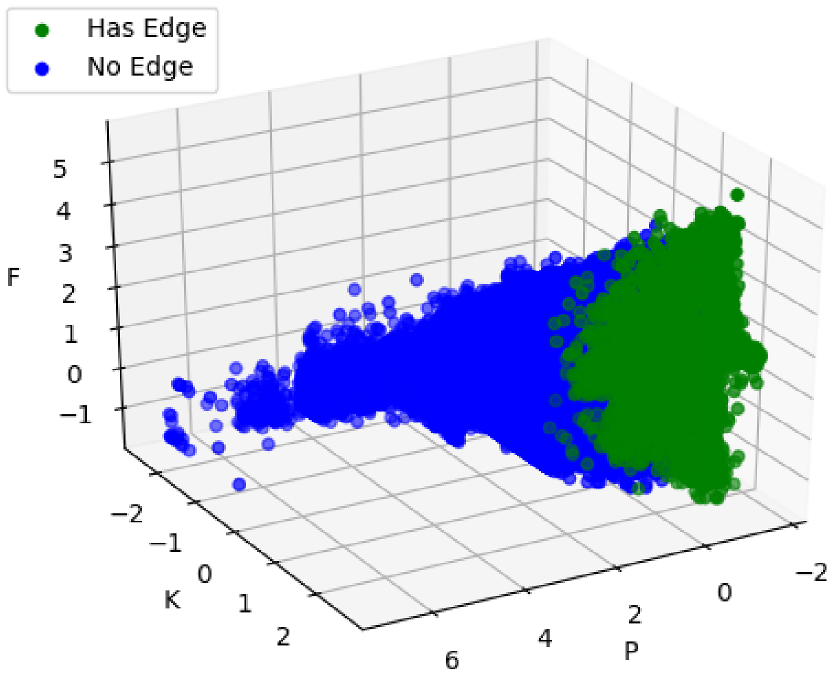


Figure 3: Visualization of data with feature K, P, F

5.5 Feedforward Neural Network

Lastly, we use a feed forward neural network with two hidden layers, which contain five and two neurons respectively. We used quasi-Newton methods in gradient descent optimization and ReLu as activation function. And we used regularization penalty as 10^{-4}.

6 Results and Discussion

For the random forest model, we have 0.985 accuracy rate with confusion matrix show in Table 2: The true positive rate is 0.477, which is much higher than random guessing, 0.0244. The running time

	Predict NO Edge	Predict Has Edge
No Edge	151902	196
Has Edge	1988	1819

Table 2: Confusion matrix of random forest

of predicting 155905 test data is 306 seconds.

The result for our SVM model is shown in the Table3: The accuracy rate is 0.9836, and the true

	Predict NO Edge	Predict Has Edge
No Edge	151908	190
Has Edge	2340	1467

Table 3: Confusion matrix of SVM

positive rate is 0.385 with random baseline as 0.0244.

The result for our Adaboost model is shown in the Table 4: The accuracy rate is 0.980, and the

	Predict NO Edge	Predict Has Edge
No Edge	150735	1363
Has Edge	1769	2038

Table 4: Confusion matrix of Adaboost

true positive rate is 0.535 with random baseline as 0.0244.

Results for our KNN model is shown in the Table 5: The accuracy rate is 0.982, and true positive rate is 0.4812 with random baseline as 0.0244. The running time is 125 sec.

Results for our Neural Network model is shown in Table 6: The accuracy rate is 0.984, and the true positive rate is 0.4592 with random baseline as 0.0244. The running time is 33 sec.

	Predict NO Edge	Predict Has Edge
No Edge	151243	855
Has Edge	1975	1832

Table 5: Confusion matrix of KNN

	Predict NO Edge	Predict Has Edge
No Edge	151717	381
Has Edge	2059	1748

Table 6: Confusion matrix of Neural Network

Since the number of positive examples in our dataset is pretty small compared to negative example, we particularly care about the accuracy of predicting a new edge if there is one in the future, which is the true positive rate. The overview is shown in Table 7.

	Total accuracy	True positive	running time(sec)
Random Forest with 6 features	0.985	0.477	306
SVM with RBF kernel	0.984	0.385	426
Adaboost	0.980	0.535	185
KNN(k=3)	0.982	0.481	125
Neural network	0.984	0.459	33

Table 7: Comparison of results

As you can see, Adaboost algorithm performs best in terms of true positive, but the most efficient one should be our neural network models, which produces relatively high true positive rate with least time.

7 Conclusion

Our project has shown that the technique of link prediction is relatively straightforward to implement, and that a relatively high prediction accuracy can be obtained with just a few features. Moreover, it is work noting that the feature ranking in itself can give much insight into the social network being studied. From our study, we can conclude that the biggest factor propelling co-authorships is the area of common interest (keyword match count), and secondly, the ”degree of separation” between two authors in the network (shortest path). This is no surprise, yet now we can confidently assert this from our results.

The ability to accurately forecast the future state of a social network is a powerful tool that can be applied to many areas. As discussed briefly in section 1.2, it, for example, can be used as a tool for national security in tracking international terrorist networks, and even as a metric to make better business decisions. Naturally, the features extracted from the network will change, but the types of features can actually be very similar. For our proximity features we counted the number of keyword matchings between two authors. For terrorist network cells, Hasan et al suggested that its proximity features could be the languages and dialects commonly used in each cell, and for general social networks, the proximity feature could be the common interests between two individuals. Therefore, the technique used here can very easily extended to analyzing many other social networks to obtain meaningful results.

7.1 Reflection

Our experience through this project started out very shaky, as we had no prior experience in social network analysis. Initially, we chose a bad dataset with little attributes, and spent much time in trying to glean meaningful results from it. We finally realized this did not work. After some research, we had realized that an essential property of any social network used in link prediction is that it had to follow the *power law*, one that described the behavior of the network. If given the experience to conduct a similar study again, we would definitely spend more time studying the nature of the problem, and be more careful with choosing a dataset to study.

7.2 Room for Improvement

Our results show that we were better able to predict true negatives, than to predict true positives. A future effort in extending this project could be in trying to improve the accuracy of predicting positive examples in our graph. We have only used a fraction of the information that the MAG provides. Aside from the information we extracted, the MAG contains information, for each academic paper, the conferences, the journal types, number of citations, the specific dates of study, etc, that might be useful in improving our results.

7.3 Github

Our dataset, the code used to process the data and to obtain our results can be found at:
https://github.com/JasontheMonster/Link-Prediction-in-Co-Authorship.

8 References

Hasan, Mohammad, et al. "Link Prediction using Supervised Learning." In Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security (2006)Print.

Liben-Nowell, David, and Jon Kleinberg. "The link-prediction Problem for Social Networks." Journal of the American Society for Information Science and Technology 58.7 (2007): 1019-31. Print.

Louppe, Gilles, et al. ”Understanding variable importances in forests of randomized trees.” Dept. of EE CS, University of Lie‘ge, Belgium.
https://pdfs.semanticscholar.org/2635/19c5a43fbf981da5ba873062219c50fdf56d.pdf.

Wang, Peng, et al. ”Link Prediction in Social Networks: The State-of-the-Art.” Science China Information Sciences 58.1 (2015): 1-38. Print.

Wasserman, Stanley, and Katherine Faust. Social Network Analysis : Methods and Applications. Cambridge ; New York: Cambridge University Press, 1994. Print.

Microsoft Academic Graph API. https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/