Link Prediction in Social Network using Artificial Neural Network

Anjali Sharma
M.Tech Scholar
Department of Computer Science
Sagar Institute of Research & Technology
Excellence, Bhopal, India

Sneha Soni
Assistant Professor
Department of Computer Science
Sagar Institute of Research & Technology
Excellence, Bhopal, India

Kalpana Rai, PhD
HOD
Department of Computer Science
Sagar Institute of Research & Technology
Excellence, Bhopal, India

ABSTRACT
The prediction process from prior information of the event helps to know the evolution of social network and assists the companies in effective decisions making during a typical recommendation system. Social network connection prediction is an efficient technique for the analysis of the evolution of social organizations and formation of the social network relations. Link prediction is a crucial research direction within the field of complex networks and data processing. In this work, we proposed a technique which is based on node similarity measure and Artificial Neural Network concept. We consider the features or information which are available in data set then assign the score to all pairs of nodes and apply Artificial Neural Network concept to the system. The proposed technique will improve the accuracy of Link Prediction.

Keywords
Social Networks, Link Prediction, Neighbourhood Tightness, Machine Learning, Supervised Algorithm, Unsupervised Algorithm, SVM, ANN.

1. INTRODUCTION
Social Networks (SNs) have attracted many users and have become an integrated a part of the individual's daily practices. The rapid climb of SNs like Twitter and Facebook has generated great deal of knowledge that sets direction for research in social relationships. The knowledge network represented by Facebook is predicated on information transmission, sharing, and exchange. The prediction process from prior information of the event helps to know the evolution of social network and assists the companies in effective decisions making during a typical recommendation system [1]. Social network connection prediction is an efficient technique for the analysis of the evolution of social organizations and formation of the social network relations [2].Link prediction is a crucial research direction within the field of complex networks and data processing . Some complex physical processes like local stochastic process also are wont to measure the similarity between network nodes and improve the accuracy of the link prediction . In other words two linked nodes during a network may have a possible relationship. By analyzing whether there's a possible relationship can help to seek out potential links and tightness measures the intensity of the connection .

Currently with the rapid development, online social network has been a neighborhood of people's life. tons of sociology, biology, and knowledge systems can use the network to explain , during which nodes represent individual and edges represent the relationships between individuals or the interaction between individuals. Therefore, the study of complex networks has been the important branch of the many scientific fields. Link prediction is a crucial task in link mining. Link prediction is to predict whether there'll be links between two nodes supported the attribute information and therefore the observed existing link information. Link prediction not only are often utilized in the sector of social network but also can be applied in other fields. As in bioinformatics, link prediction are often wont to discover interactions between proteins [1]; within the field of electronic commerce, link prediction are often wont to create the advice system [2]; and within the security field, link prediction can help to seek out the hidden terrorist criminal gangs [3]. Link prediction is closely associated with many areas. Therefore, in recent years there are tons of correlation algorithms proposed to unravel the matter of link prediction.

1.1 PRELIMINARIES
Formulizing a drag during a graph may be a popular technique in computing and may be used for various applications like mapping, link prediction and cluster formation. the matter of link prediction in SNs are often defined as an instance of SN where we aim to predict the links along side their likelihood score for possibilities of connections. We model the network as a graph (G), G = (V , E), where V is that the collection of vertices, E shows the gathering of links, and e =(a , b) ∈ E represents an interaction between nodes a and b. Each non-existent link are often represented as (j , k) ∈ U − E, where j , k ∈ V and U denotes the universal set.

CommonNeighbors

$$SS = |\Gamma(x) \cap \Gamma(y)|$$

where CN is common neighbors approach for link prediction and Γ (x) shows neighbor set for x ∈ V. The term SS represents the potential of the set and the number of common neighbors is equal to number of paths with two lengths between two nodes.

```
In [2]: def CommonNeighbors(u, v, g):
            u_neighbors = set(g.neighbors(u))
            v_neighbors = set(g.neighbors(v))
            return len(u_neighbors.intersection(v_neighbors))
        def common_neighbors(g, edges):
            result = []
            for edge in edges:
                node_one, node_two = edge[0], edge[1]
                num_common_neighbors = 0
                try:
                    neighbors_one, neighbors_two = g.neighbors(node_one), g.neighbors(node_two)
                    for neighbor in neighbors_one:
                        if neighbor in neighbors_two:
                            num_common_neighbors += 1
                    result.append((node_one, node_two, num_common_neighbors))
                except:
                    pass
            return result
```

Jaccard's coefficient

Jaccard's coefficient is a normalized version of common neighbor that takes into account the total number of neighbors for both vertices. Node intersection divided by union given as follows:

$$S_{xy}^{Jaccard} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

```
[6]: def JaccardCoefficent(u, v, g):
         u_neighbors = set(g.neighbors(u))
         v_neighbors = set(g.neighbors(v))
         return len(u_neighbors.intersection(v_neighbors)) / float(len(u_neighbors.union(v_neighbors)))
```

Preferential attachment

Preferential attachment is based on the idea that highly connected nodes are more likely to form links mathematically represented in Equation. The probability of links generated by x and y is proportional to the product of two node degrees.

$$S_{xy}^{PA} = |\Gamma(x)||\Gamma(y)|$$

```
: def PreferentialAttachment(u, v, g):
      return len(g.neighbors(u))*len(g.neighbors(v
```

Resource Allocation

Resource Allocation (RA) states that contribution of different degree values of the common neighbour nodesis also different and smaller nodes affect the nodes higher than the degree. The calculation formula is as follows:

$$S_{xy}^{RA} = \sum_{z \epsilon (\Gamma(x) \cap \Gamma(y))} \frac{1}{|K(z)|}$$

where z is the common neighbour node set of x and y, K (z) shows the degree of node z, 1 /|K(z)| denotes assigning weights according to the degree value of node z.

```
: def ResourceAllocation(u, v, g):
      u_neighbors = set(g.neighbors(u))
      v_neighbors = set(g.neighbors(v))
      ra = 0
      for i in u_neighbors.intersection(v_neighbors):
          ra += 1 / float(len(g.neighbors(i)))
      return ra
```

AdamicAdar

Adamic Adar (AA) describes that the difference between the AA index and the algorithm is that the weights assigned to the common neighbor nodes are different. Logarithmic constraints are applied to weights.

$$S_{xy}^{AA} = \sum_{z \epsilon (\Gamma(x) \cap \Gamma(y))} \frac{1}{|\log K(z)|}$$

```
ı [4]: def AdamicAdar(u, v, g):
           u_neighbors = set(g.neighbors(u))
           v_neighbors = set(g.neighbors(v))
           aa = 0
           for i in u_neighbors.intersection(v_neighbors):
               aa += 1 / math.log(len(g.neighbors(i)))
           return aa
```

2. LITERATURE REVIEW

According to [1], Currently online social network with rapid development has become a neighborhood of people's life. supported user interest social network will change over time with different nodes and edges. Predicting new relation and missing relation between nodes during a social network are often identified using link prediction. Where new links and nodes are often identified with attribute information. Machine learning approaches are used with a gaggle of features to increase the performance using supervised learning setup.

In [2] , Link prediction could also be a way to forecast future new or missing relationships between entities supported this network information. Graph theory and network science are theoretical concepts that have influenced the link prediction research. Although previous reviews clearly outlined the link prediction research, it had been focused on describing prediction approaches only. However, analysis of related studies identified other components that influence link prediction.

According to [3], Link prediction may be a crucial task in social network analysis, the core of which is to predict the establishment of future links between nodes in social network. A weighted network model supported time information is proposed. Because the preference of nodes within the network will change with external factors, the time decay factor is introduced into the algorithm through the logistic function. The link weights between node pairs are determined by the link establishment time information, and thus the weighted network is used to reinforce the effect of supervised learning link prediction.

3. PROBLEM DEFINITION

The first challenge in supervised link prediction is extreme class skewness. The problem of class skewness in supervised learning is well known in machine learning. The poor performance of a learning algorithm in this case results from both the variance in the models estimates and the imbalance in the class distribution. Different quite links or edges between the nodes exist during a social network. for instance , social contacts, phone-calls or hyper-references. On analysis of social networks, there are often many information about the linkage between the nodes that aren't discovered or unknown at a given point of your time . Link Prediction is that the problem of predicting links that either dont yet exist at the given time t or exist, but unknown up to the present time.

The link prediction problem is additionally deals with the matter of getting missing links from a known network, during a number of elds. It involves prediction of additional links that aren't directly visible currently, are likely to exist during a network supported observable data. It considers a static picture of the network, instead of taking network evolution and network dynamics. It also considers speci c prop-erties of the nodes within the network, instead of computing the facility of prediction methods that focuses on the graph structure.

4. PROPOSED WORK

The link prediction algorithms supported user input as maximum path to be traversed predicts the probability of formation of link between any two nodes of the network by traversing all the paths of the network up there to certain input path length. It rest traverses the trail lengths of 2 i.e. the immediate neighborhoods of the node. we will say it runs an area algorithm on path length 2. It then produces a similarity list between every two nodes. When it traverses the graph for path length 3, it uses new paths that are made by path length 2 to urge the new path lengths of three and computes their similarity matrix

and updates the similarity matrix during a cumulative way. This process continues till the graph is traversed up to the utmost path length to be traversed given by user. Let we get a path from A to B while traversing for the trail length of n' with some similarity value, once we traverse the graph for path length value n, then we'll check all the neighbors of B(i.e. C) to urge path from A to C. We compute the similarity of every pair (A; C) and update the trail list if their no direct link between A and C within the original graph.

The inputs to the algorithm are the graph in terms of an inventory or adjacency array, the total number of nodes the graph, maximum length to be traversed, which determines what percentage time the algorithm will run. The output of the algorithm may be a similarity list containing the similarity value between every two nodes by traversing the trail lengths of given maximum user input value. By observing the similarity matrix we will predict the future links. The high similar values have more probability to make links in near future and that we can classify the values supported certain threshold value. The similarity values quite the edge value are likely to make future links. This prediction are often compared with the test data to urge the efficiency of the algorithm.

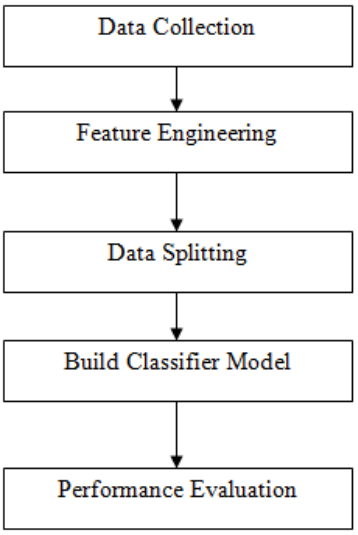


Fig 1. Proposed Block Diagram

4.1 Algorithm Steps

Step-1. Data Collection :- Facebook dataset collected from SNAP

Step-2. Feature Engineering : Collected dataset has no features to work with. Therefore extracted different features from the dataset using different link prediction techniques

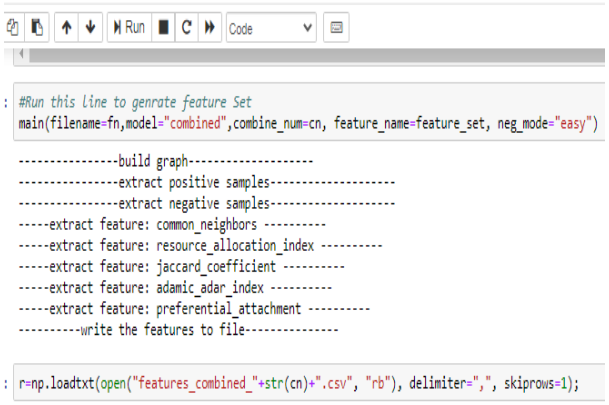
Step-3. Data Splitting: In machine learning dataset as to be split into train and test data. Where train data is used to train the model and model is tested with test data to measure the model performance.

Step-4. Build Classifier Model: we can build ML model on SVM (Supervised) and ANN(Unsupervised) algorithm on training data and make prediction on test dataset.

Step-5. Performance Evaluation: Based on Prediction we can calculate the performance of the algorithm .

5. EXPERIMENTAL & RESULT ANALYSIS

This section presents the experimental results of the work done and the proposed approach. For performing experimental analysis we can use jupyter notebook which is an IDE of python mainly used for building machine learning models. For Loading Facebook link dataset from SNAP and manipulating the dataset we are using pandas or numpy package of python and after loading data we apply various features extraction techniques to calculate the similarity index between the nodes which is shown in figure 2.



```
#Run this line to genrate feature Set
main(filename=fn,model="combined",combine_num=cn, feature_name=feature_set, neg_mode="easy")

----------build graph----------
----------extract positive samples----------
----------extract negative samples----------
-----extract feature: common_neighbors ----------
-----extract feature: resource_allocation_index ----------
-----extract feature: jaccard_coefficient ----------
-----extract feature: adamic_adar_index ----------
-----extract feature: preferential_attachment ----------
----------write the features to file----------

r=np.loadtxt(open("features_combined_"+str(cn)+".csv", "rb"), delimiter=",", skiprows=1);
```

Figure 2. Apply Feature Extraction Techniques on Dataset

By applying various feature extraction techniques on dataset we can calculate the similarity index of each nodes in the dataset and the similarity index are shown in figure 3.

common_neighbors	resource_allocation_index	jaccard_coefficient	adamic_adar_index	preferential_attachment
1	0.54895519	0.434782609	5.458317958	1088
5	1.118861666	0.5625	13.67728512	5561
1	0.733931803	0.396551724	18.86115972	24480
2	0.747321234	0.595121951	23.7772022	26676
7	0.876087375	0.725	28.29148203	29624
1	0.417479654	0.34375	11.14620167	10744
2	0.28334699	0.338709677	4.794651	1680
8	0.527100324	0.432038835	17.30741609	20634
0	0.279717854	0.192982456	5.016146472	4399
0	0.677649547	0.223214286	6.807583297	4386
1	0.406245979	0.262135922	10.91762405	16644
1	0.370682667	0.278481013	9.061676502	9976

Figure 3. Similarity index

After calculating the similarity index we can split the data into two part with a ratio of 75% data for training and remaining 25% is for testing and normalize it , steps for these are shown in figure 4.

```
np.random.shuffle(r);
```

```
train_l=int(0.75*l)
X_train=r[0:train_l,0:b-1]
Y_train=r[0:train_l,b-1]
X_test=r[train_l:l,0:b-1]
Y_test=r[train_l:l,b-1]
X_train = normalize(X_train, axis=0, norm='max')
X_test = normalize(X_test, axis=0, norm='max')
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Figure 4. Splitting the Data

After splitting the data we can train the SVM algorithm on training dataset and once the model is ft we can test the performance of the model on testing data and the result we have get are shown in figure 5.

```
def mySvm(training, training_labels, testing, testing_labels):
    #Support Vector Machine
    start = datetime.datetime.now()
    clf = svm.SVC()
    clf.fit(training, training_labels)
    print ("+++++++++ Finishing training the SVM classifier ++++++++++++")
    result = clf.predict(testing)

    print ("SVM accuracy:", accuracy_score(testing_labels, result))
    #keep the time
    finish = datetime.datetime.now()
    print ((finish-start).seconds)
```

```
#Run this to for SVM classification
mySvm(X_train,Y_train,X_test,Y_test)
```

```
+++++++++ Finishing training the SVM classifier ++++++++++++
SVM accuracy: 0.6219281663516069
0
```

Figure 5. Performance Measure of SVM

After evaluating the performance of supervised algorithm we can train the unsupervised ANN algorithm for the same and the performance result we have got are shown in figure 6.

```
def ANN(training, training_labels, testing, testing_labels):
    clf = MLPClassifier(solver='adam', alpha=1e-5,hidden_layer_sizes=(15,9), random_state=1)
    start = datetime.datetime.now()
    clf.fit(training, training_labels)
    print ("+++++++++ Finishing training the ANN classifier ++++++++++++")
    result = clf.predict(testing)

    print ("ANN accuracy:", accuracy_score(testing_labels, result))
    #keep the time
    finish = datetime.datetime.now()
    print ((finish-start).seconds)
```

```
# Run this for ANN classification
ANN(X_train,Y_train,X_test,Y_test)
```

```
+++++++++ Finishing training the ANN classifier ++++++++++++
ANN accuracy: 0.6465028355387523
1
```

Figure 6. Performance Measure of ANN

5.1 Performance Comparison

The validity of the model can be observed using accuracy of the model along with “false positive” and “false negatives”.

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of data points}}$$

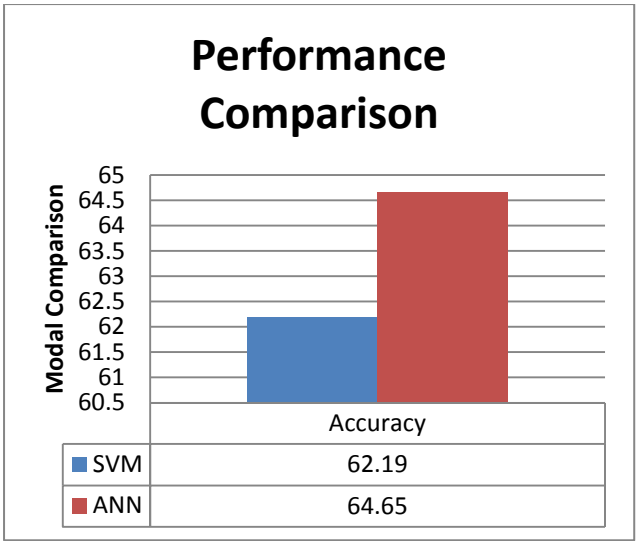


Figure 7. Performance comparison of model

6. CONCLUSION

There are various different approaches for link prediction problems in social networks but they are generic algorithms with little focus on social networks. In this we used various features engineering techniques and combined them to improve the prediction accuracy. In this we can build machine learning model based on supervised algorithm (SVM) and Unsupervised Algorithm (ANN) on social graph dataset and than we evaluate the performance and we found that ANN perform better as compared to SVM.

7. REFERENCES

[1] Ramya Bv, Dr. N Sandeep Varma, R Indra" Recommendations In Social Network Using Link Prediction Technique " In 2020, IEEE .

[2] Herman Yuliansyah , Zulaiha Ali Othman , And Azuraliza Abu Bakar " Taxonomy Of Link Prediction For Social Network Analysis: A Review " In IEEE 2020.

[3] Zhipeng Yang; Xiaohui He, "Link Prediction In Weight Social Networks Based On Time Series", In IEEE 2020.

[4] Li Bian, Henry Holtzman, "Online Friend Recommendation through Personality Matching and Collaborative Filtering", UBICOMM 2011: The Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies

[5] Xiao Yu, Ang Pan, Lu-An Tang, Zhenhui Li, Jiawei Han, "Geo-Friends Recommendation in GPS-Based Cyber-Physical Social Network", 2017 International Conference on Advances in Social Networks Analysis and Mining

[6] Mir Saman Tajbakhsh, Vahid Solouk," Semantic GeoLocation Friend Recommendation System; LinkedIn User Case", 6th Conference on Information and Knowledge Technology (lKT 2014), May 28-30, 2016, Shahrood University of Technology, Tehran, Iran.

[7] Zhibo Wang, Student Member, IEEE, Jilong Liao, Qing Cao, Member, IEEE, Hairong Qi, Senior Member, IEEE, and Zhi Wang, Member, IEEE," Friendbook: A Semantic-Based Friend Recommendation System for Social Networks", IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 14, NO. 3, MARCH 2015

[8] Tofik R. Kacchi, Prof. Anil V. Deorankar, "Friend Recommendation System based on Lifestyles of Users", International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB16)

[9] Huansheng Ning, Senior Member, IEEE, Sahraoui Dhelim, and Nyothiri Aung, "PersoNet: Friend Recommendation System Based on Big-Five Personality Traits and Hybrid Filtering", IEEE TRANSACTIONS, VOL. 6, NO. 3, JUNE 2019

[10] https://www.verywellmind.com/the-big-five-personalitydimensions-2795422 01/10/2019

[11] Fateme Keikha, Dr. Mohammad Fathian, Dr. Mohammad Reza Gholamian, "Comparison and Evaluation of Recommendation Systems on Social Networks", Journal of Basic and Applied Scientific Research J. Basic. Appl. Sci. Res., 3(10)52-58, 2013.

[12] W. Youyou, D. Stillwell, H. A. Schwartz, and M. Kosinski, "Birds of a feather do flock together: Behavior-based personality-assessment method reveals personality similarity among couples and friends," Psychol. Sci., vol. 28, no. 3, pp. 276–284, 2017.