PSG COLLEGE OF TECHNOLOGY

19OH01-Social and Economic Network Analysis

DEPARTMENT

of

COMPUTER SCIENCE AND ENGINEERING

Link Prediction in Social Network

Team Members:
Adhavan Alexander (18z201)
Mitran.R (18z230)
Sivaraj (18z251)
Suthan Atithya R.S (18z257)
Vishnu P (18z263)

Problem Statement

Currently with the rapid development, online social network has been a part of people's life. A lot of sociology, biology, and information systems can use the network to describe, in 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 many scientific fields. Link prediction is to predict whether there will be links between two nodes based on the attribute information and the observed existing link information. Link prediction not only can be used in the field of social network but can also be applied in other fields. As in bioinformatics, link prediction can be used to discover interactions between proteins in the field of electronic commerce, link prediction can be used to create the recommendation system and in the security field, link prediction can help to find the hidden terrorist criminal gangs. Link prediction is closely related to many areas.

Important Terminologies

- **Common Neighbours:** Common neighbours captures the idea that two strangers who have a friend in common are more likely to be introduced than those who don't have any friends in common.
- **Resource allocation index**: Resource Allocation Index is obtained by calculating the average score for each cluster and then by averaging those scores.
- **Jaccard coefficient**: The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets.
- **Adamic Adar index**: Adamic Adar is a measure used to compute the closeness of nodes based on their shared neighbours.
- **Preferential attachment**: Preferential attachment means that the more connected a node is, the more likely it is to receive new links.

Dataset Description

This dataset consists of 'circles' (or 'friends lists') from Facebook. Facebook data was collected from survey participants using this Facebook app. The dataset includes node features (profiles), circles, and ego networks.

Facebook data has been anonymized by replacing the Facebook-internal ids for each user with a new value. Also, while feature vectors from this dataset have been provided, the

interpretation of those features has been obscured. For instance, where the original dataset may have contained a feature "political=Democratic Party", the new data would simply contain "political=anonymized feature 1". Thus, using the anonymized data it is possible to determine whether two users have the same political affiliations, but not what their individual political affiliations represent.

Dataset Statistics	
Nodes	4039
Edges	88324
Nodes in largest WCC	4039
Edges in largest WCC	88234
Nodes in larges SCC	4039
Edges in largest SCC	88234
Average clustering coefficient	0.6055
Number of Triangles	1612010
Fraction of Closed Triangles	0.2647
Diameter	8
90-Percentile	4.7

Tools Used

TensorFlow, Sklearn, Numpy, Pandas, Networkx, Keras.

Challenges Faced

1. The created data for link prediction was unable to fit within the model we created, after normalizing the data, the data fitted into the model with good accuracy.
2. Initially we chose logistic regression model for prediction but accuracy was not satisfying it was near 65%, so we decided to move towards artificial neural network(ANN) and sigmoid activation function for non-linearity, which raised to accuracy to 97.58%.
3. After model creation we used binary cross entropy in which we got 95% accuracy, using sparse categorical entropy accuracy improved to 97.58%.
4. At the initial stage of input we used only 3 features that is common neighbors, resource allocation index(RAI) and Jaccard Coefficient, which was not efficient enough to predict the node link, so we have included two more attributes: Adamic Adar index and preferential attachment.

Contribution of Team Members

Roll No	Name	Contribution
18z201	Adhavan Alexander	**Frontend & Report**
18z230	Mitran.R	**Model Training**
18z251	Sivaraj.J	**Model Training**
18z257	Sudhan Adithya R S	**Backend Development**
18z263	Vishnu P	**Frontend & Report**

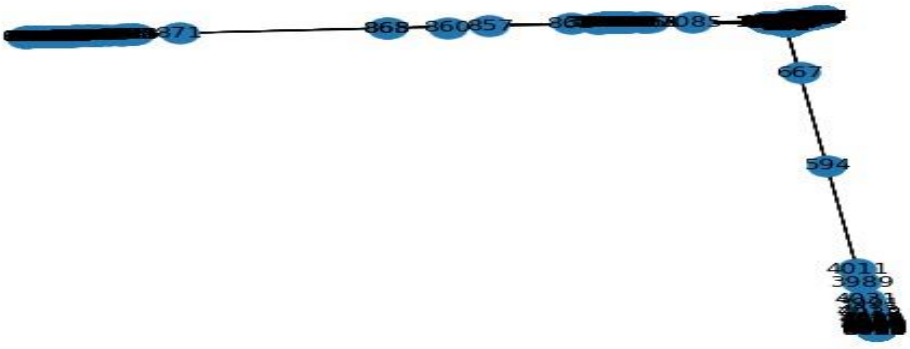
Annexure 1 Code:

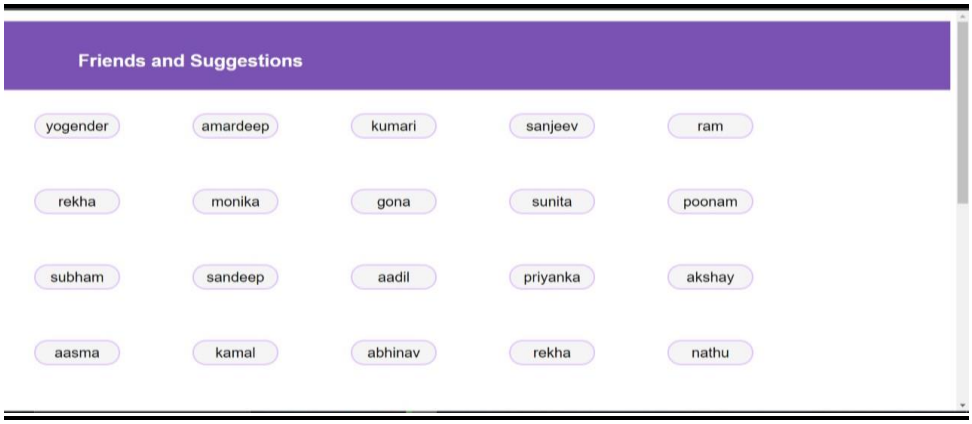
```
#Installing networkx Library
!pip install network
#Importing Required Libraries
import networkx #For handling Network graphs
import random
import csv
import numpy as np
import pandas as pd
from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import Normalizer
from tensorflow.keras.losses import categorical_crossentropy
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from tensorflow.keras.optimizers import Adam
from keras.utils import np_utils
#stroing dataset path
path='/content/drive/MyDrive/SENA/facebookdata.txt'
#Loading dataset into dataset as graph in variable
with open(path,'rb') as file:
  graph=networkx.read_edgelist(file)
networkx.draw_spectral(graph,with_labels = True)
def Generate_False_Graph(Graph,neg_no):
  """
  Input : a graph as Networkx class
          number of negative samples
  Output :neg_no number of false edges (edges which does not exist in the graph)
          and build a new graph using negative edges
  """
  Negative_graph= networkx.Graph()
  count=0
  while (count<neg_no):
    [source,target]=random.sample(Graph.nodes(),2) # pick random two nodes in the graph
    try :

      short_path=networkx.shortest_path_length(Graph,source,target)
      if(short_path>=2):# there does not exist a direct edges betwwen them
        Negative_graph.add_edge(source,target, positive="False") # adding the false edges in the new graph
created
        count += 1
    except:
```

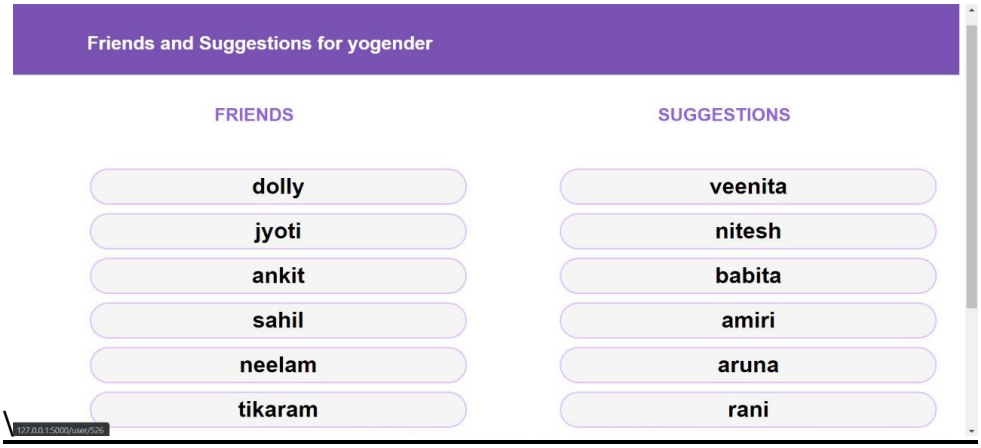
```
        pass
    return Negative_graph
-------------------------------------------------------------------------------
def Extract_Sample(Graph,pos_no,neg_no):
    """
    Input : a graph as Networkx class
    Output : List of Positive and Negative samples
    """
    no_edges=Graph.number_of_edges()
    # Poistive samples
    pos_graph = networkx.Graph()
    positves=random.sample(Graph.edges(),pos_no)
    pos_graph.add_edges_from(positves, positive="True")
    networkx.write_edgelist(pos_graph, "positive_graph.txt", data=['positive'])
    # Negative Samples
    neg_graph=Generate_False_Graph(Graph,neg_no)
    networkx.write_edgelist(neg_graph, "negatvie_graph.txt", data=['positive'])
    return pos_graph.edges(),neg_graph.edges()
-------------------------------------------------------------------------------
def Extract_feature(Graph,positives,negatives):
    """
    Input: Graph as network class
           Positive dataset(Nodes having edges among them)
           Negative Dataset(Nodes having shortest path greater than 2)
    Output:Features of the two nodes in the given dataset
           Features: Common Neighbours, Jacaard Coefficient, Resource allocation index, Adamic Adar Coefficient,
Preferential Attachment
    """
    data = []
    feature_name = [common_neighbors,networkx.resource_allocation_index,networkx.jaccard_coefficient,
networkx.adamic_adar_index, networkx.preferential_attachment]
    label = ["label"] + ["1" for i in range(len(positives))] + ["0" for i in range(len(negatives))]
    for func in feature_name:
        preds = func(Graph, positives)
        feature = [func._name_] + [i[2] for i in preds]
        preds = func(Graph, negatives)
        feature = feature + [i[2] for i in preds]
        data.append(feature)
    data.append(label)
    col_names = []
    records = []
    for col in data:
        col_names.append(col[0])
        records.append(col[1:])
    records = np.array((records)).T
    csvfile = pd.DataFrame(records, columns = col_names).to_csv('features.csv')
#Extracting 5000 samples for each dataset
x,y=Extract_Sample(graph,5000,5000)
Extract_feature(graph, x, y)
dataset=pd.read_csv("features.csv")
x=dataset.iloc[:,:-1].values
```

```
y=dataset.iloc[:,-1].values
#Normalizing the dataset
t = Normalizer().fit(x)
x = t.transform(x)
--------------------------------------------------------------------------------
considering as coordinate with color chosen for that label corresponding to the record
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
y_train = np_utils.to_categorical(y_train).astype('float32')
y_test = np_utils.to_categorical(y_test).astype('float32')
#Building the base model
model = Sequential()
model.add(Input(len(x[0])))
model.add(BatchNormalization(axis=-1))
model.add(Dense(32, activation='relu'))
model.add(BatchNormalization(axis=-1))
model.add(Dense(32, activation='relu'))
model.add(BatchNormalization(axis=-1))
model.add(Dense(2, activation='softmax'))
#Compiling the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
#Model training
model.fit(X_train, y_train, batch_size=200, epochs=1000)
model.evaluate(X_test,y_test)
model.save('/content/drive/MyDrive/SENA/model.h5')
```

Annexure 2 Output Snapshots:







References:

https://www.analyticsvidhya.com/blog/2020/01/link-prediction-how-to-predict-your-future-connections-on-facebook/

https://stellargraph.readthedocs.io/en/stable/demos/node-classification/gcn-node-classification.html

https://www.tensorflow.org/js/tutorials/training/nodejs_training

https://www.hindawi.com/journals/sp/2015/172879/

https://www.sciencedirect.com/topics/computer-science/link-prediction

http://snap.stanford.edu/data/ego-Facebook.html

https://archive.siam.org/meetings/sdm06/workproceed/Link%20Analysis/12.pdf

https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2163-9

https://neo4j.com/docs/graph-algorithms/current/labs-algorithms/linkprediction/

https://papers.nips.cc/paper/2018/file/53f0d7c537d99b3824f0f99d62ea2428-Paper.pdf

SmallSEOTools

PLAGIARISM SCAN REPORT

Words	564	Date	April 15,2021
Characters	6778	Excluded URL	

0% Plagiarism	100% Unique	0 Plagiarized Sentences	17 Unique Sentences