Project 2

Pranav Menon

April 25, 2024

1 Question

The project deals with the impression network formed between 143 nodes, and we analyse how to find the missing links between two nodes

This means that if two nodes A and B never meet each other during the experiment then we predict if A can find B impressionable or nor.

2 Algorithm

In the algorithm, we use the matrix method and linear combinations which help us to predict the links between two nodes successfully.

- Iterate over the entire matrix. If we find a missing link between two nodes then we prepare to use the method of linear combination.
- As we know that humans behave linearly, hence the solution to the system of linear equations $AX = B$ can be used to predict the missing links between two nodes.
- If we find that the missing link attains a positive value, then we take the missing link as 1, else 0. Please note that the matrix method gives out different values for each node - the strength of that node, but since finally we need to predict if the edge exists or not, we just give binary outputs.

3 Code and Implementation:

3.1 Importing libraries

```
import numpy as np
import networkx as nx
import pandas as pd
import math
import matplotlib.pyplot as plt
import random
```

All these libraries will be used for computing the final leader of the impression network.

3.2 Preprocessing and making the graph

```
G = nx.DiGraph()
for index, row in df.iterrows():
row_list = row.tolist()
t = row_list[1]
t = t[:11].lower()
for i in range(2, 31):
    if(not isinstance(row_list[i], str)):
        continue
    x = row_list[i]
    x = x[-11:]
    x = x.lower()
    G.add_edge(t, x)
```

This makes the plot for the graph.

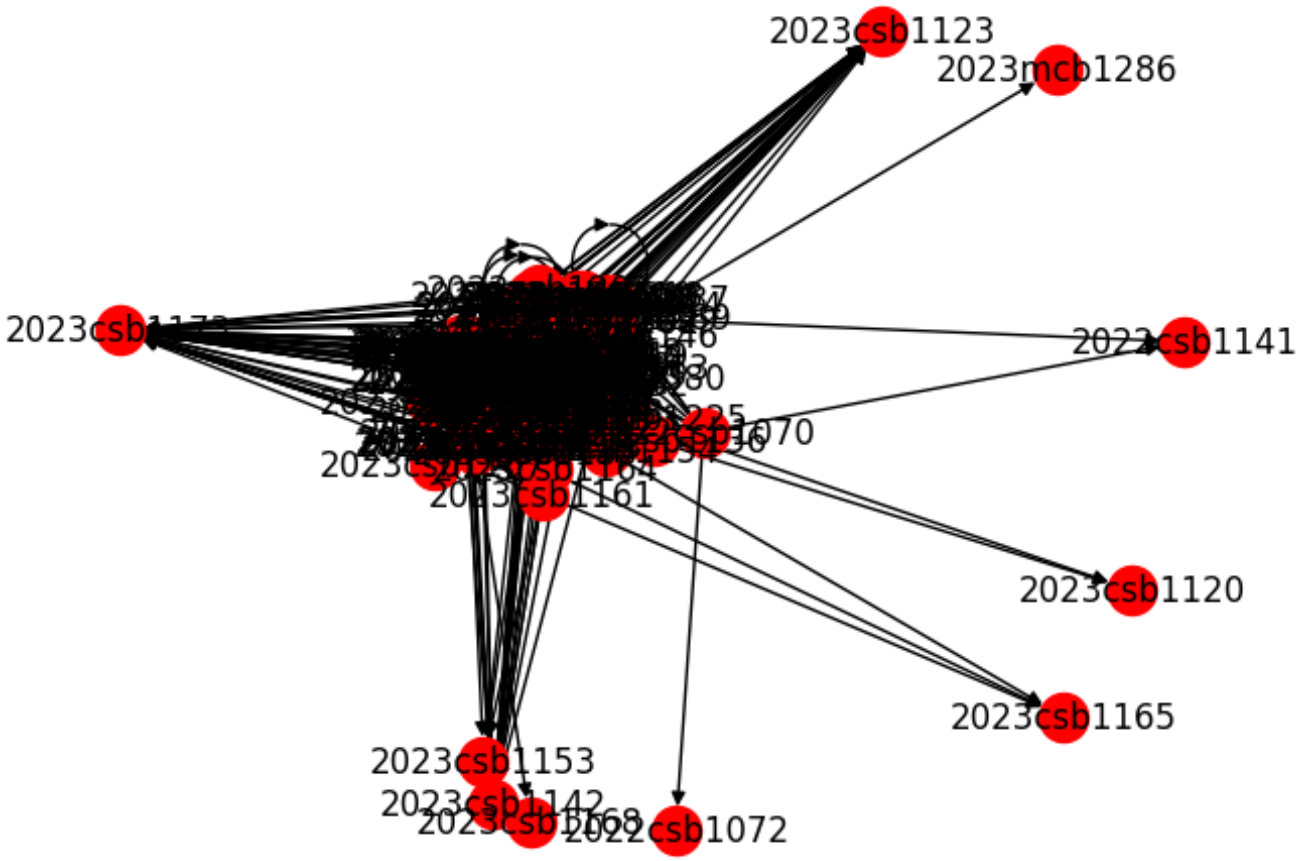

Figure 1: Image for the question.

3.3 Preparing matrix:

```
for i in range(143):
for j in range(143):
    if(matrix[i][j] == matrix[j][i] and matrix[i][j] == 0 and i!=j):
        matrix[i][j] = -1
        matrix[j][i] = -1
```

This changes the value of node to -1 if two nodes A and B never meed during the process of the activity.

3.4 Matrix Prediction:

```
def find_link(i, j):
  try:
    A = matrix[:i-1, :j-1]
    B = matrix[i, :j-1]
    C = np.transpose(A)
    x = np.linalg.lstsq(C, B, rcond=None)[0]
    x = x.reshape(-1, 1)
    M = matrix[:i-1, j]
    M = M.reshape(-1, 1)
    final = np.dot(M.T, x)
    # print(final)
    if(final > 0):
      return 1
    else:
      return 0
  except np.linalg.LinAlgError as e:
    return 0
```

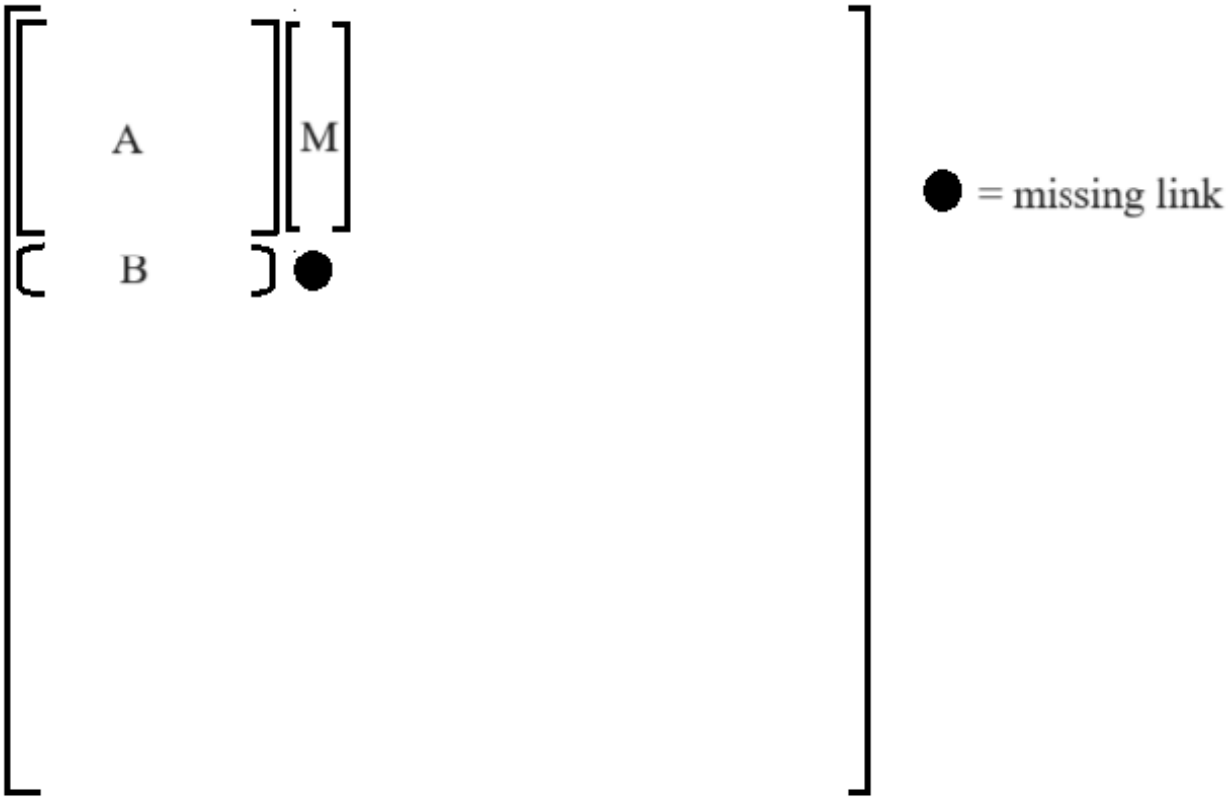

Figure 2: Image for the question.

Now to predict the missing links between A and B. We first prepare our matrix A and B. Consider the above picture. Matrix A and Matrix B are the matrices in the equation $Ax = B$. The above can be calculated using inbuilt functions in python.

The same linear combination can be applied on the matrix M to achieve the missing link which we were trying to figure out. The method does not return exact 0 or 1. It in turn returns a number. If this number is greater than 0 it can be taken to be 1, else a 0.

4 Result :

For the sake of it all the missing links cannot be listed in this listed file. Some of them are:

2023csb1162 −− > 2023mcb1298
2023csb1162 −− > 2023csb1091
2023csb1162 −− > 2023csb1156
2023csb1162 −− > 2023csb1143
2023csb1162 −− > 2023mcb1301
2023csb1162 −− > 2023csb1134
2023csb1162 −− > 2023csb1138
2023csb1162 −− > 2023csb1128
2023csb1162 −− > 2023mcb1285
2023csb1162 −− > 2023csb1146
2023csb1162 −− > 2023mcb1299
2023csb1162 −− > 2023csb1133
2023csb1162 −− > 2023csb1110
2023csb1162 −− > 2023csb1129
2023csb1162 −− > 2023csb1099
2023csb1162 −− > 2023csb1130
2023csb1162 −− > 2023csb1139
2023csb1162 −− > 2023csb1172
2023csb1162 −− > 2023mcb1292
2023csb1162 −− > 2020mcb1225
2023csb1162 −− > 2023csb1094

There are many other links which have been recommeded by the algorithm.