Mini Project : Semester I

Course : MCA - Data Science & Informatics

Project Name : Social Network Graph of Email Communications

Group Number 19:
Rishabh Bhardwaj - 2446002
Mohammed Irfan - 2446046
Kuldeep Lohani - 2446086

Submitted to :
Dr. Akshay Deepak
21 November 2024

INDEX

1. INTRODUCTION

In recent years, the emergence of digital communication has turned the organizational and personal communications upside down. Email, one of the most prevalent modes of communication, offers glimpses into the dynamics of interpersonal relations and information circulation. It is important to note, however, that knowing how emails are exchanged within an organization or between parties assisting a firm is one of the fundamental components in the process involving distinguishing the networks and their actors and determining the connections concealed within them.

The Social Network Graph of Email Communications project aims to analyze the structure of email exchanges within the Enron corporation using graph theory. In this case, individuals are referred to as nodes and communication lines between nodes are the edges in a social network graph which can be used to describe the interactions of the individuals or groups within the structure.

The Enron Email Dataset serves as a rich source of data, containing over 500,000 email messages exchanged between employees of the Enron corporation before its collapse in 2001. This dataset offers an excellent opportunity to construct a social network and analyze it from various perspectives such as communication frequency, centrality, and network density.

As a result of this project, we hope to make use of some aspects of graph theory and the study of networks to understand email communication in organizations more clearly. As part of this effort, we will pinpoint the important communicators, analyze how two or more people are related to each other, and provide an explanation of the social network that should result from the emails.

2. PROBLEM STATEMENT

1. Construct a social network graph from a dataset of email communications.
2. Use the "Enron Email Dataset" available here: Enron Email Dataset. Link: https://www.kaggle.com/datasets/wcukierski/enron-email-dataset
3. Write a Python program that reads the email data to identify sender-recipient pairs.
4. Use the networkx library to build a social network graph where nodes represent individuals, and edges represent communication between them.
5. Visualize the network using matplotlib. Adjust node sizes based on the number of emails sent/received.
6. Highlight nodes with the highest centrality scores (most influential individuals) in the network.

3. METHODOLOGY

The approach to building the social network graph of email communications is carried out in the following steps:

3.1 Data Collection

The dataset used for this project is the Enron Email Dataset, which contains email communication records. The dataset was downloaded from the Kaggle repository and includes various details such as the sender, recipient, subject, and body of each email.

3.2 Data Preprocessing

- The raw dataset contains unnecessary columns and some missing values that need to be cleaned before processing.
- The file column is dropped as it is not relevant to the analysis.
- The message column, which contains the actual email content, is parsed to extract important fields like the sender (From) and recipient (To) using the Python email module.
- Empty values (NaN) are dropped, and the index is reset to ensure the data is consistent for further analysis.

3.3 Data Transformation

- The “To” field (recipient) can contain multiple recipients separated by commas. This is split into individual email addresses and then "exploded" into separate rows for each recipient using the explode() function.
- Both the From and To fields are stripped of any leading or trailing whitespace to avoid any inconsistencies in email addresses.
- Self-references (where the sender and recipient are the same) are removed since they do not contribute to the network structure.

- Duplicate entries are removed to ensure each communication pair is represented only once.

3.4 Identifying Core Users

The core users of the email network are identified as the top 10 senders and top 10 receivers based on the frequency of their email communications. These core users form the central nodes in the social network graph.

3.5 Visualizing the Network Graph

A network graph is constructed using the networkx library.

- Each unique user is represented as a node, and an edge is created between two nodes (representing a sender-recipient pair) if both the sender and recipient are in the core users list.
- The edges are weighted equally since the focus is on identifying influential individuals, not the volume of communication.

3.6 Degree Centrality Calculation

To find the most influential individuals, the degree centrality of each node in the graph is calculated. Degree centrality measures the number of direct connections (edges) a node has. A higher degree centrality indicates a more influential person in the network.

3.7 Visualization

- The network graph is visualized using matplotlib and networkx.
- The size of each node is proportional to its degree centrality score, so more influential nodes (with higher centrality) will appear larger in the graph.
- The most influential nodes are highlighted with a different color (brown) compared to the others (blue) to make them stand out.
- The network layout is circular to represent the interconnectedness of the individuals.

3.8 Analysis of Centrality

After the visualization, the most influential individuals are identified by checking the nodes with the highest degree centrality values. If multiple nodes share the highest value, all are printed as the most influential individuals. By following this methodology, the project effectively constructs a social network graph, identifies core users, and highlights the most influential individuals based on their connectivity in the network.

4. RELEVANT CODE

4.1 Importing the required libraries

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

4.2 Reading the data

```
# Reading the dataset
dataframe=pd.read_csv("emails.csv")
```

```
dataframe.head()
```

```
dataframe.shape
```

```
# Dropping the 'file' column
df=dataframe.drop('file',axis=1)
```

4.3 Extracting 'From' and 'To' Fields from Email Messages

```
# Function to extract fields from the message
def get_field(field, messages):
    column = []
    for message in messages:
        e = email.message_from_string(message)
        column.append(e.get(field))
    return column
```

```
# Extracting 'From' and 'To' fields from email
messages
df['From']=get_field('From',df['message'])
df['To']=get_field('To',df['message'])
```

```
# Dropping the 'message' column
edf=df.drop('message',axis=1)
```

```
edf.head()
```

4.4 Data Cleaning and Preprocessing

```
edf.isna().sum()
```

```
# Dropping null values and resetting index
edf=edf.dropna()
edf = edf.reset_index(drop=True)
print(edf.head(50))
print(edf.shape)
```

```
# Splitting 'To' column by commas and exploding it
into individual rows
edf['To'] = edf['To'].apply(lambda x: x.split(','))
edf = edf.explode('To', ignore_index=True)
edf.shape
print(edf.head(50))
```

```
# Stripping whitespace from 'From' and 'To' columns
edf['From']=edf['From'].str.strip()
edf['To']=edf['To'].str.strip()
```

```
# Dropping self-references (same sender and receiver)
index=edf[(edf["From"]==edf["To"])].index
print(index)
edf.drop(index,inplace=True)
```

```
# Dropping duplicate entries
edf=edf.drop_duplicates()
```

4.5 Identifying Unique Senders and Receivers

```
# Getting unique senders and receivers
unique_senders = edf['From'].value_counts()
print(unique_senders.head(50))
```

```
unique_receivers = edf['To'].value_counts()
print(unique_receivers.head(50))
```

```
# Getting the top 10 senders and receivers
top_sen=unique_senders.head(10)
top10_senders = top_sen.index.tolist()
top_rec = unique_receivers.head(10)
top10_rec = top_rec.index.tolist()
```

```
# Creating the set of core users (top senders and
receivers)
core_users = set(top10_senders).union(set(top10_rec))
for index,i in enumerate(core_users, 1):
    print(f"{index}.", i)
```

4.6 Building the Social Network Graph

```
# Creating a network graph using NetworkX
G = nx.Graph()

# Adding edges for the core users
for index, row in edf.iterrows():
    sender = row['From']
    recipient = row['To']

    if sender in core_users:
        if recipient in core_users:
            if not G.has_edge(sender, recipient):
                G.add_edge(sender, recipient,
```

4.7 Calculating Degree Centrality and Plotting the Network

```
# Calculating degree centrality for each node
degree_centrality = nx.degree_centrality(G)

# Plotting the network graph
plt.figure(figsize=(12, 12))

node_sizes = [500 * degree_centrality[node] for node
in G.nodes()]

most_central_nodes = sorted(degree_centrality.items(),
key=lambda x: x[1], reverse=True)
top_centrality_value = most_central_nodes[0][1]
influential_nodes=[]
for node, centrality_value in most_central_nodes:
     if centrality_value==top_centrality_value:
          influential_nodes.append(node)

n_colors = ['brown' if node in influential_nodes else
'blue' for node in G.nodes()]

# Circular layout for the graph
pos = nx.circular_layout(G)
nx.draw_networkx_nodes(G, pos, node_size=node_sizes,
node_color=n_colors, alpha=0.7)
nx.draw_networkx_edges(G, pos, width=1.0, alpha=0.5,
edge_color='black')
nx.draw_networkx_labels(G, pos, font_size=9,
font_family="sans-serif")

# Display the graph
plt.title("Social Network Graph of Enron")
```

4.8 Printing the Most Influential Person

```
# Print the most influential person

if len(influential_nodes)==1:
    most_influential_node = max(degree_centrality,
key=degree_centrality.get)
    print(f"The most influential person is:
{most_influential_node}")
else:
    for i in influential_nodes:
         print(i)
```

5. RELEVANT OUTPUTS

5.1 Data Inspection

```
# Reading the dataset
dataframe=pd.read_csv("emails.csv")
```

```
dataframe.head()
```

```
dataframe.shape
```

Output

```
                        file                                            message
0     allen-p/_sent_mail/1.  Message-ID: <18782981.1075855378110.JavaMail.e...
1    allen-p/_sent_mail/10.  Message-ID: <15464986.1075855378456.JavaMail.e...
2   allen-p/_sent_mail/100.  Message-ID: <24216240.1075855687451.JavaMail.e...
3  allen-p/_sent_mail/1000.  Message-ID: <13505866.1075863688222.JavaMail.e...
4  allen-p/_sent_mail/1001.  Message-ID: <30922949.1075863688243.JavaMail.e...

(517401, 2)
```

5.2 Checking for Missing Values

The command edf.isna().sum() is used to check if there are any missing values in the dataset.

Output

```
From          0
To        21847
dtype: int64
```

This output indicates that there are 21847 null values in "To" column which are then removed.

5.3 Exploded 'To' Column

Before exploding: Each row in the 'To' column may contain multiple email addresses (separated by commas).

Output

```
(495554, 2)
```

After exploding: Each email address gets its own row, which increases the total number of rows.

Output

```
(3130272, 2)
```

We can see that the rows have increased from 495554 to 3130272 due to the exploding function.

5.4 Dropping Self-References and Duplicates

```python
# Dropping self-references (same sender and receiver)
index=edf[(edf["From"]==edf["To"])].index
print(index)
edf.drop(index,inplace=True)
```

```python
# Dropping duplicate entries
edf=edf.drop_duplicates()
df.shape
```

Output

```
  df.shape
✓  0.3s
(517401, 3)
```

We can see that the number of rows have reduced now.

5.5 Core Users

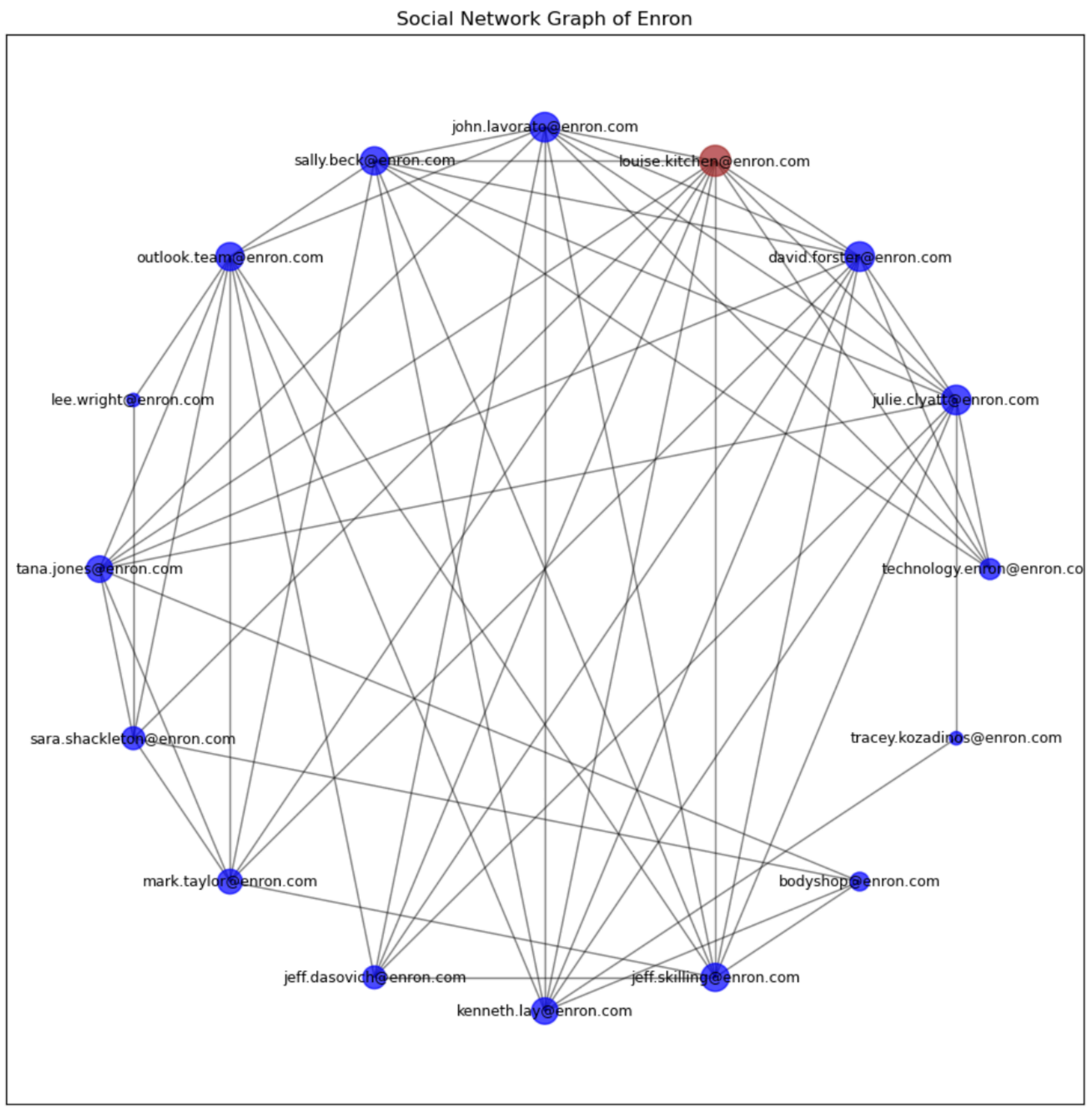
```
# Creating the set of core users (top senders and
receivers)
core_users = set(top10_senders).union(set(top10_rec))
for index,i in enumerate(core_users, 1):
    print(f"{index}.", i)
```

Output

```
1. outlook.team@enron.com
2. technology.enron@enron.com
3. louise.kitchen@enron.com
4. klay@enron.com
5. sara.shackleton@enron.com
6. john.lavorato@enron.com
7. tana.jones@enron.com
8. tracey.kozadinos@enron.com
9. david.forster@enron.com
10. kenneth.lay@enron.com
11. sally.beck@enron.com
12. lee.wright@enron.com
13. jeff.dasovich@enron.com
14. bodyshop@enron.com
15. julie.clyatt@enron.com
16. mark.taylor@enron.com
17. jeff.skilling@enron.com
```

The list is 17 in number instead of 20 (Top 10 senders+Top 10 recievers) because there are some common users that are there in both lists.

5.6 Social Network Graph Visualization



The highlighted node is the most influential individual, i.e, louis.kitchen@enron.com

6. CONCLUSION

In this project, we constructed a social network graph based on the Enron email dataset to visualize the communication patterns between individuals. By leveraging Python's powerful libraries such as Pandas, NetworkX, and Matplotlib, we successfully processed and analyzed the email data to uncover insights about the most influential individuals in the network.

The methodology involved reading the dataset, extracting key fields (such as sender and receiver), and cleaning the data by handling missing values and removing duplicate or self-referential emails. We then used NetworkX to create a graph where nodes represented individuals, and edges represented the communication between them. To highlight the central figures in the network, we calculated degree centrality and adjusted node sizes accordingly.
The resulting visualization not only gave us a clear picture of the communication flow within the Enron organization but also allowed us to identify the most influential individuals in the network based on their centrality. This analysis demonstrated how social network graphs can be used to understand relationships and power dynamics in a dataset of this nature.

Overall, the project achieved its goal of constructing a social network graph that offers meaningful insights into the structure of email communications. Further work could include exploring other centrality measures, like betweenness or closeness, to gain a more comprehensive understanding of the network's key players. Additionally, analyzing different time periods within the dataset could provide insights into how the communication patterns evolved over time.

7. REFERENCES

1. Kaggle. (n.d.). Enron Email Dataset. Retrieved from https://www.kaggle.com/datasets/wcukierski/enron-email-dataset.

2. NetworkX Documentation. (n.d.). NetworkX: Network Analysis in Python. Retrieved from https://networkx.github.io/.

3. Matplotlib Documentation. (n.d.). Matplotlib: Plotting with Python. Retrieved from https://matplotlib.org/stable/users/index.html.

4. Pandas Documentation. (n.d.). Pandas: Python Data Analysis Library. Retrieved from https://pandas.pydata.org/.

5. Python Software Foundation. (n.d.). Python Documentation. Retrieved from https://docs.python.org/3/.

6. Waskiewicz, M. (2018). Data Science with Python: Exploring the Enron Email Dataset. Retrieved from https://www.datacamp.com/community/tutorials/enron-email-dataset-python.