PSG COLLEGE OF TECHNOLOGY, COIMBATORE 641004

Department of Computer Science and Engineering

19ZO02 - SOCIAL AND ECONOMIC NETWORK ANALYSIS

PROJECT REPORT

Topic : Merging multi modal customer service tickets

Team members

19z209 - Deepika S
19z212 - G Solaisneha
19z222 - Kaniska Varsini P D
19z231 - Pavithra L
19z233 - Pranitha V S
19z252 - Swetha G

Table of Contents

Problem Statement

Solving customer issues is now a part of any organization offering its products or services. Customers reach out to companies by opening a ticket with their email or phone number and describing their issue. They contact the customer service via multiple channels. Oftentimes, customers reach out to customer service with the same issue, but under different email ids, phone numbers and accounts. Merging such tickets makes it efficient to address them correctly and in least amount of time. Our problem statement is to merge relevant tickets of a user to efficiently handle customer issues.

Analysis

1. Merging tickets with the same issues to avoid addressing duplicate issues with the help of orderID, email, phoneNo. (A user may have more than one email,phoneNo and orders)

2. Analysis on how many times a user has contacted customer service for issue and find users who have raised higher number of issues

3. Analysis on the number of unique orders a user has made to find the super users and prioritize resolving their issues.

Dataset Description

Contacts is the dataset used. It contains 5,00,000 instances, with the following five features:

- Id - uniquely identifies a ticket
- Email - email id with which the ticket was initiated
- Phone - phone number with which the ticket was initiated
- Contacts - Number of times user reached out in the particular ticket
- OrderId - uniquely identifies an transaction

Dataset link : https://www.kaggle.com/datasets/eugenevoon/multichannel-contacts-problem

Tools used

Gephi:

Gephi is a visualization application developed in the Java language. It is mainly used for visualizing, manipulating, and exploring networks. The user interacts with the representation, manipulates the structures, shapes, and colors to reveal hidden patterns.

Python Packages used:

1. Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

2. Networkx

NetworkX is a package for the Python programming language that's used to create, manipulate, and study the structure, dynamics, and functions of complex graph networks

Challenges faced:

- As the dataset is huge(50 Mb file with 5 lakh instances) , it was difficult to handle the data
- Importing the dataset and running visualization algorithms on Gephi was difficult given the size of the dataset
- Extracting the required nodes from the constructed graph was challenging

Contributions:

NAME	ROLLNO	CONTRIBUTION
Deepika S	19z209	Graph creation
G Solai Sneha	19z212	Analysis on how many times a user has contacted customer service for issue and find users who have raised higher number of issues
Kaniska Varshini PD	19z222	Node extraction from connected components
Pavithra L	19z231	Merging tickets with the same issues to avoid addressing duplicate issues with the help of orderID, email, phoneNo
Pranitha V S	19z233	Dataset and Gephi visualization

Swetha G	19z252	Analysis on the number of unique orders a user has made to find the super users and prioritize resolving their issues.

Annexure I: Code

```
import pandas as pd
import networkx as nx

#Reading data

df=pd.read_json('/contacts.json')
df.Email=df.Email.apply(lambda x: "Email_"+x if x !=" else ")
df.Phone=df.Phone.apply(lambda x: "Phone_"+x if x !=" else ")
df.OrderId=df.OrderId.apply(lambda x: "OrderId_"+x if x !=" else ")

#Create graph G with edges id -> email, id -> phone, id ->orderId

nodes=[]
for _,Id,_,_,Contacts,_ in df.itertuples():
    nodes.append((Id,{"Contacts": Contacts}))
G=nx.Graph()
G.add_nodes_from(nodes)
G.add_edges_from(df[df.Email != ''][['Id', 'Email']].to_records(index=False))
G.add_edges_from(df[df.Phone != ''][['Id', 'Phone']].to_records(index=False))
G.add_edges_from(df[df.OrderId != ''][['Id', 'OrderId']].to_records(index=False))
print(G.number_of_nodes(), G.number_of_edges())

#Finding connected components in Graph G
conn_comp=list(nx.connected_components(G))

#Analysis

users=[]
contact_sum=[]
unique_orders=[]
# for each connected component we appended the ticket_id within
for each_connected_component in conn_comp:
    id_list=[]
```

```
    order_list=[]
    for each_node in each_connected_component:
        # check if the node is a number, append to the id_list
        if str(each_node).isnumeric():
            id_list.append(each_node)
#check if the node is an OrderId, append to the order_list
        if str(each_node).startswith('OrderId_'):
            order_list.append(each_node)
    # sum of contacts that a node has made
    sum_of_contacts=0
    for order_id in id_list:
        sum_of_contacts += G.nodes[order_id]['Contacts']
    output_str1='-'.join([str(each_node) for each_node in sorted(id_list)])
    output_str2=','.join([str(each_node) for each_node in sorted(order_list)])
    for id in id_list:
        users.append([id, output_str1])
        contact_sum.append(sum_of_contacts)
    for order_id in id_list:
        unique_orders.append([order_id,output_str2,len(order_list)])

#Output 1 - Merging tickets belonging to the same user

output_final1=pd.DataFrame(users)
output_final1=output_final1.rename(columns={0:'id', 1:'ticket_trace'})
output_final1.sort_values('id',inplace=True)
print(output_final1)

#Output 2 - Number of times a user has contacted customer service

output_final2=pd.DataFrame(contact_sum)
output_final2=output_final2.rename(columns={0:'No. of contacts made'})
output_final2.sort_values('No. of contacts made',ascending=False,inplace=True)
print(output_final2)

#Output 3 - Number of unique orders a user has made

output_final3=pd.DataFrame(unique_orders)
output_final3=output_final3.rename(columns={0: 'id', 1:'Unique Orders', 2:'No. of Orders'})
output_final3.sort_values('No. of Orders',ascending=False,inplace=True)
print(output_final3)
```

Annexure II - Snapshots of the Output

Output 1 : Merging tickets belonging to the same user

sena.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Output 1 - Merging tickets belonging to the same user

```
[19] output_final1=pd.DataFrame(users)
     output_final1=output_final1.rename(columns={0:'id', 1:'ticket_trace'})
     output_final1.sort_values('id',inplace=True)
     print(output_final1)
```

```
                id                                ticket_trace
0                0                                           0
1                1  1-2458-98519-115061-140081-165605-476346
8                2                     2-159312-322639-348955
12               3                                           3
13               4                                           4
...            ...                                         ...
499996      499995                                      499995
499997      499996                                      499996
499998      499997                                      499997
236554      499998                                 121111-499998
499999      499999                                      499999

[500000 rows x 2 columns]
```

Output 2 - Number of times a user has contacted customer service

sena.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Output 2 - Number of times a user has contacted customer service

[20]
```
output_final2=pd.DataFrame(contact_sum)
output_final2=output_final2.rename(columns={0:'No. of contacts made'})
output_final2.sort_values('No. of contacts made',ascending=False,inplace=True)
print(output_final2)
```

```
        No. of contacts made
60351                     99
9063                      99
60339                     99
9056                      99
9057                      99
...                      ...
304023                     0
304016                     0
454397                     0
304011                     0
499999                     0

[500000 rows x 1 columns]
```

Output 3 - Number of unique orders a user has made

sena.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Output 3 - Number of unique orders a user has made

```python
output_final3=pd.DataFrame(unique_orders)
output_final3=output_final3.rename(columns={0: 'id', 1:'Unique Orders', 2:'No. of Orders'})
output_final3.sort_values('No. of Orders',ascending=False,inplace=True)
print(output_final3)
```

```
             id                                      Unique Orders  \
9072     304811  OrderId_BNiqIFerGaPGDfHYQtEMDctQz,OrderId_BZfV...
8524     102740  OrderId_ATdeYmjfHGpRkxOEiUQEYIEHz,OrderId_DFKI...
8519      18722  OrderId_ATdeYmjfHGpRkxOEiUQEYIEHz,OrderId_DFKI...
8520     443180  OrderId_ATdeYmjfHGpRkxOEiUQEYIEHz,OrderId_DFKI...
8521      12596  OrderId_ATdeYmjfHGpRkxOEiUQEYIEHz,OrderId_DFKI...
...         ...                                               ...
148343    61814
350579   235657
350580   235658
350581   289537
0             0

         No. of Orders
9072                22
8524                22
8519                22
8520                22
8521                22
...                ...
148343               0
350579               0
350580               0
350581               0
0                    0

[500000 rows x 3 columns]
```

References

1. Dataset link : https://www.kaggle.com/datasets/eugenevoon/multichannel-contacts-problem
2. Problem statement: https://www.kaggle.com/competitions/scl-2021-da
3. https://pandas.pydata.org/docs/
4. https://networkx.org/documentation/stable/reference/index.html
5. https://gephi.org/users/
6. https://www.geeksforgeeks.org/networkx-python-software-package-study-complex-networks/
7. https://towardsdatascience.com/social-network-analysis-from-theory-to-applications-with-python-d12e9a34c2c7
8. https://www.cl.cam.ac.uk/teaching/1415/L109/l109-tutorial_2015.pdf
9. https://www.askpython.com/python/examples/network-graphs-from-pandas-dataframe
10. https://www.python-graph-gallery.com/320-basic-network-from-pandas-data-frame/