CSE 3021: SOCIAL AND INFORMATION NETWORKS

Restaurant Recommendation System using Python

PROJECT REPORT

Team Members:

15BCE0675 SHIVAM GUPTA
15BCE0584 AYUSH KULSHRESTHA
15BCE0662 YASH G GUPTA

VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
Vellore-632 014, Tamil Nadu, India.
www.vit.ac.in

ABSTRACT

A recommendation system is a subclass of information filtering system that seeks to predict the preference a user would give to a product. Such systems are increasingly popular in recent years, and utilized in variety of areas of market. For the recommendation system to work various criteria like previous browsing history of user, purchases, cost preferences etc. are considered.

For the development of our project-Hotel Recommendation system, we have used data from two sites- Foursquare and Yelp.

Foursquare is a local search and discovery system which provides personalized recommendation of place.

Yelp is crowd-sourced local business site where users submit a review of services/products using one to five-star rating system.

For the working of recommendation system, data is extracted from Yelp and Foursquare during run time using web crawlers.

Also, we have the datasets of the user-venue based details of Foursquare, which we have acquired from archive.org.

LITERATURE SURVEY

G. Adomavicius gives an outline of the field of recommender frameworks and depicts the present age of suggestion techniques that are typically arranged into the accompanying three primary classes: content based, shared, and mixed suggestion approaches. This paper additionally depicts different constraints of current suggestion strategies and talks about conceivable expansions that can move forward proposal capacities and make recommender frameworks material to a significantly more extensive scope of uses. These expansions incorporate among others a change of comprehension of clients and things joining of the logical data into the proposal procedure.

J. J. Lewandowsky gives a view on LARS an area - aware recommender framework that utilizations area - based appraisals to deliver suggestions. Customary recommender frameworks don't consider spatial

properties of clients nor things LARS then again underpins scientific categorization of three novel classes of area - based evaluations, in particular, spatial appraisals for non-spatial things, non-spatial evaluations for spatial things, and spatial evaluations for spatial things. LARS misuses client rating areas through client dividing, a method that impacts suggestions with evaluations spatially near questioning clients in a way that boosts framework adaptability while not relinquishing suggestion quality. LARS can apply these systems independently, or in show contingent upon the sort of area based rating accessible. Trial prove utilizing vast scale genuine world information from both the foursquare area.

J. S. Breese proposed Collaborative sifting or recommender frameworks which utilize a database about client inclinations to predict extra points or items another user may like. In this paper creators portray a few calculations intended for a problem, including strategies in view of connection coefficients, vector based likeness counts, and factual Bayesian strategies. Creators look at the prescient precision of the different techniques in an arrangement of delegate issue areas. Creators utilize two essential classes of assessment measurements. The first portrays precision over an arrangement of individual predicts as far as normal total deviation. The second gauges the utility of a positioned rundown of recommended things. This metric uses a gauge of the likelihood that a user will see a recommendation in a list.

MODULES

Module-1

- **Data** – Data is extracted from two sites: Foursquare and Yelp. Data from Yelp and Foursquare is collected in run- time.
- **Web scrapping**- It is used for extracting data from front-end of a website. It is form of copying html source-code of page from which data is extracted from web. For scraping, BeautifulSoup library is used.
- **Recommendation**- The user receives recommendation from two web-sites. Recommendation considers location and price criteria.

- **GUI**- System is developed using Python and for the development of interface Tkinter library is used.

Module-2

- **Data –** Data related to user-venue of foursquare is acquired from archive.org website. Data sets include: Sociograph, Ratings, Check-In data, and location based information of users and venues.
- **Recommendation-** In this module, the recommendation is done on the basis of where the friends of a user like to go.
- **Graphs-** Graphs are plotted based on the datasets provided. Ex- User-User friend graph, User-Venue Rating graph,etc.
- **GeoPlotting-** Since the data sets are provided with the location of users and venues, thus their locations are plotted on map using Basemap library, and markers are shown on google maps.

METHODOLOGY

Module-1

Input

- A user is required to enter the name of a city, in which it wants to search for hotels/restaurants. User can also enter current location, so the algorithm searches hotels nearby its location.
- User can also enter the price budget, which filters the hotels based on: cheap, costly, costlier, and costliest.

Algorithm

- As the user clicks search button, the program sends the URL, which includes name of city and price range (if entered).
- "Urllib" library stores the HTML of the page after retrieving resource and pass it to the "BeautifulSoup" library which is used for scraping (crawling) the data.
- Data is then extracted from the HTML page, with the help of tag searching and filtering. Data which is to be extracted contains: Name of the hotel ,and the overall rating given by users.
- This process is applied to both Foursquare and Yelp websites.
- Data is then represented in GUI which is developed using "Tkinter" library of python.

Module-2

Input

- A user is required to enter the user ID, which is provided by the foursquare.

Algorithm

- User IDs of friends of the user are extracted, and their liked venues are taken.
- The locations of the venues and user are stored.
- Distance of each user-venue pair is calculated on the basis of latitude and longitude.
- Then, venues are filtered according to the distance.
- User location and venues are plotted on geomap using "basemap" library and on google maps by URL request.
- Thus, venues which are visited by the friends of user and have high ratings, and are also near to the user are recommended.

IMPLEMENTATION

Module-1

Code:

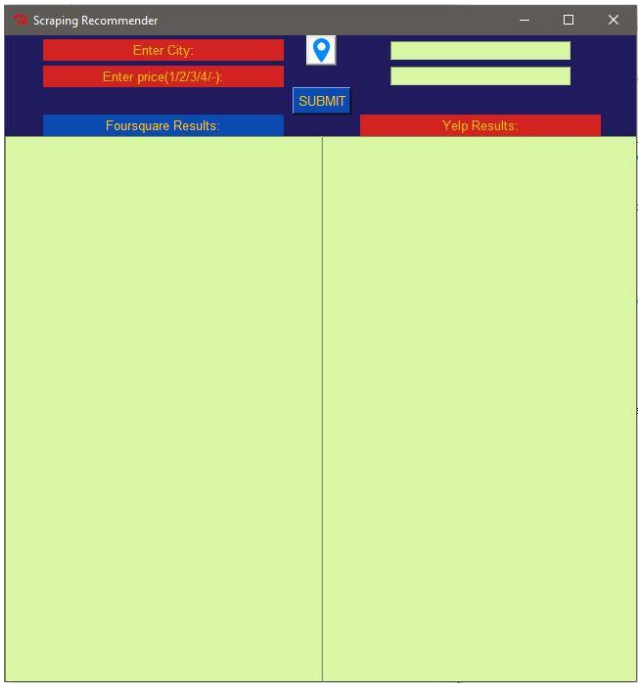
```
from bs4 import BeautifulSoup
import urllib as ur
import Tkinter as tk
from tkMessageBox import *
import requests
import json
from PIL import ImageTk
def foursquare(city,price):

    if(price!="-"):

response=ur.urlopen('https://foursquare.com/explore?mode=url&near='+cit
y+'&price='+price+'&q=food')
    else:
```

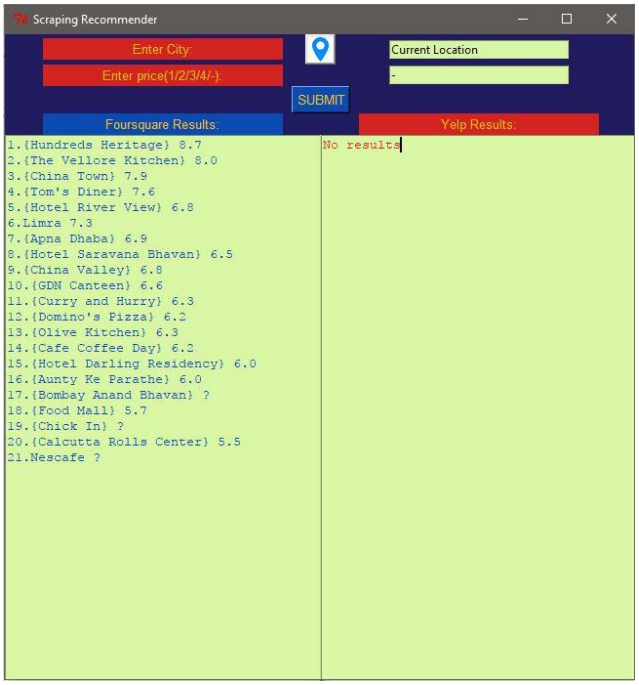
```
response=ur.urlopen('https://foursquare.com/explore?mode=url&near='+cit
y+'&price=&q=food')



    html=response.read()

    soup=BeautifulSoup(html,'html.parser')

    name=[]
    rate=[]
    add=[]

    y="""<li class="card singleRecommendation hasPhoto"""
    for link in soup.find_all('li'):
        if (y in str(link)):
            name.append(link.find('div',{"class":"venueName"}))
            if(link.find('div',{"class":"venueScore positive"})!=None):
                rate.append(link.find('div',{"class":"venueScore
positive"}))
            elif(link.find('div',{"class":"venueScore
neutral"})!=None):
                rate.append(link.find('div',{"class":"venueScore
neutral"}))
            else:
                rate.append(link.find('div',{"class":"venueScore
unknown"}))

    rate1=[]

    for x in rate:
        if(x!=None):
            rate1.append(x.string.encode("utf-8"))
        else:
            rate1.append("?")
    name1=[]
    for x in name:
        name1.append(x.a.string.encode("utf-8"))


    name_rate=[]
    for i in range(0,len(name1)):
        name_rate.append((name1[i],rate1[i]))

    return name_rate
root=tk.Tk()
root.geometry("645x660")
root.configure(bg='#211C5F')
root.title("Scraping Recommender")

def locate():
    send_url = 'http://freegeoip.net/json'
    r = requests.get(send_url)
```

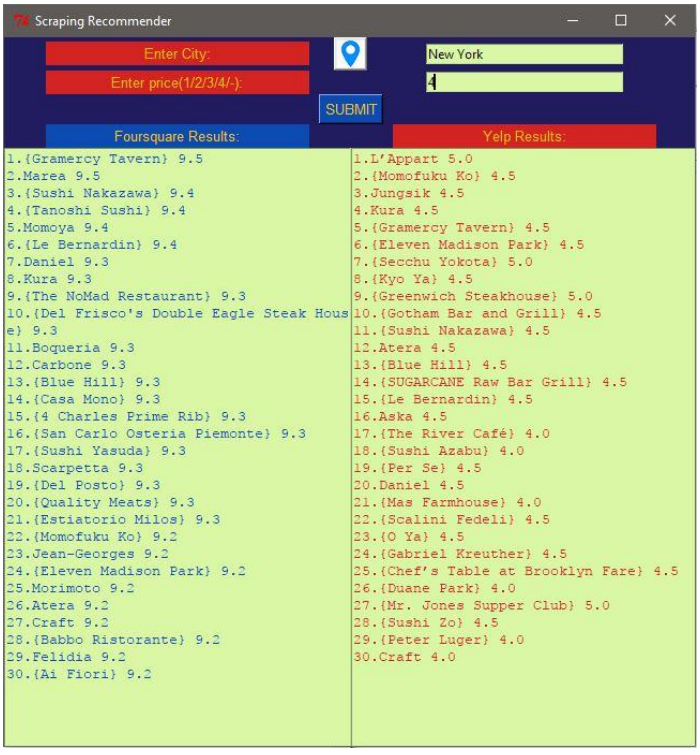
```
    j = json.loads(r.text)
    lat = j['latitude']
    lon = j['longitude']
    e1.delete(0,tk.END)
    city=str(lat)+" "+str(lon)
    e1.insert(tk.END,"Current Location")
    return city
def get():
    city=e1.get()
    if(city=="Current Location"):
        city=locate()
    price=e2.get()
    if city=="":
        showerror(title="Empty Field",message="Please Enter a City!")
        return
    l=["1","2","3","4","-"]
    if price not in l:
        showerror(title="Please Wait",message="Please Enter Correct
Price Range!")
        return

    res1=foursquare(city,price)
    res2=yelp(city,price)

    counter=0
    T1.delete('1.0',tk.END)
    if(len(res1)==0):
        T1.insert(tk.END,"No results")
    for i in res1:
        counter+=1
        T1.insert(tk.END,counter)
        T1.insert(tk.END,".")
        T1.insert(tk.END,i)
        T1.insert(tk.END,"\n")

    #T1.config(state=tk.DISABLED)

    T2.delete('1.0',tk.END)
    if(len(res2)==0):
        T2.insert(tk.END,"No results")
    counter=0
    for i in res2:
        counter+=1
        T2.insert(tk.END,counter)
        T2.insert(tk.END,".")
        T2.insert(tk.END,i)
        T2.insert(tk.END,"\n")

    #T2.config(state=tk.DISABLED)
    for x in res1:
        for y in res2:
            if(x[0].lower()==y[0].lower()):
                print((x[0],x[1]))
```

```
w1=tk.Label(root,width=30,text="Enter
City:",font=("Arial",10),bg="#d32323",fg="#FFC300")
w1.grid(row=0,column=0)

e1=tk.Entry(root,width=30,bg="#DAF7A6")
e1.grid(row =0, column =1)


mi=ImageTk.PhotoImage(file="2.png")


b=tk.Button(root,text="Loc",command=locate)
b.grid(row=0 ,columnspan=2)
b.config(image=mi)

l2=tk.Label(root,width=30,text="Enter price(1/2/3/4/-
):",font=("Arial",10),bg="#d32323",fg="#FFC300")
l2.grid(row=1,column=0)

e2=tk.Entry(root,width=30,bg="#DAF7A6")
e2.grid(row =1, column =1)

b1=tk.Button(root,text="SUBMIT",font=("Arial",10),command=get,bg="#0c4c
b2",fg="#FFC300")
b1.grid(rowspan=1,columnspan=2)

l3=tk.Label(root,width=30,text="Foursquare
Results:",bg="#0c4cb2",font=("Arial",10),fg="#FFC300")
l3.grid(row=4,column=0)

T1=tk.Text(root,height=35,width=40,bg="#DAF7A6",fg="#0c4cb2")
T1.grid(row=5,column=0)


l4=tk.Label(root,width=30,text="Yelp
Results:",bg="#d32323",font=("Arial",10),fg="#FFC300")
l4.grid(row=4,column=1)

T2=tk.Text(root,height=35,width=40,bg="#DAF7A6",fg="#d32323")
T2.grid(row=5,column=1)

root.mainloop()
```

OUTPUT:


Scraping Recommender
Enter City:
Enter price(1/2/3/4/-):
SUBMIT
Foursquare Results:
Yelp Results:


Scraping Recommender
Enter City:
Enter price(1/2/3/4/-):
Current Location
-
SUBMIT
Foursquare Results:
Yelp Results:
1.{Hundreds Heritage} 8.7
2.{The Vellore Kitchen} 8.0
3.{China Town} 7.9
4.{Tom's Diner} 7.6
5.{Hotel River View} 6.8
6.Limra 7.3
7.{Apna Dhaba} 6.9
8.{Hotel Saravana Bhavan} 6.5
9.{China Valley} 6.8
10.{GDN Canteen} 6.6
11.{Curry and Hurry} 6.3
12.{Domino's Pizza} 6.2
13.{Olive Kitchen} 6.3
14.{Cafe Coffee Day} 6.2
15.{Hotel Darling Residency} 6.0
16.{Aunty Ke Parathe} 6.0
17.{Bombay Anand Bhavan} ?
18.{Food Mall} 5.7
19.{Chick In} ?
20.{Calcutta Rolls Center} 5.5
21.Nescafe ?
No results


Scraping Recommender
Enter City:
Enter price(1/2/3/4/-):
New York
4
SUBMIT
Foursquare Results:
Yelp Results:
1.{Gramercy Tavern} 9.5
2.Marea 9.5
3.{Sushi Nakazawa} 9.4
4.{Tanoshi Sushi} 9.4
5.Momoya 9.4
6.{Le Bernardin} 9.4
7.Daniel 9.3
8.Kura 9.3
9.{The NoMad Restaurant} 9.3
10.{Del Frisco's Double Eagle Steak House} 9.3
11.Boqueria 9.3
12.Carbone 9.3
13.{Blue Hill} 9.3
14.{Casa Mono} 9.3
15.{4 Charles Prime Rib} 9.3
16.{San Carlo Osteria Piemonte} 9.3
17.{Sushi Yasuda} 9.3
18.Scarpetta 9.3
19.{Del Posto} 9.3
20.{Quality Meats} 9.3
21.{Estiatorio Milos} 9.3
22.{Momofuku Ko} 9.2
23.Jean-Georges 9.2
24.{Eleven Madison Park} 9.2
25.Morimoto 9.2
26.Atera 9.2
27.Craft 9.2
28.{Babbo Ristorante} 9.2
29.Felidia 9.2
30.{Ai Fiori} 9.2
1.L'Appart 5.0
2.{Momofuku Ko} 4.5
3.Jungsik 4.5
4.Kura 4.5
5.{Gramercy Tavern} 4.5
6.{Eleven Madison Park} 4.5
7.{Secchu Yokota} 5.0
8.{Kyo Ya} 4.5
9.{Greenwich Steakhouse} 5.0
10.{Gotham Bar and Grill} 4.5
11.{Sushi Nakazawa} 4.5
12.Atera 4.5
13.{Blue Hill} 4.5
14.{SUGARCANE Raw Bar Grill} 4.5
15.{Le Bernardin} 4.5
16.Aska 4.5
17.{The River Café} 4.0
18.{Sushi Azabu} 4.0
19.{Per Se} 4.5
20.Daniel 4.5
21.{Mas Farmhouse} 4.0
22.{Scalini Fedeli} 4.5
23.{O Ya} 4.5
24.{Gabriel Kreuther} 4.5
25.{Chef's Table at Brooklyn Fare} 4.5
26.{Duane Park} 4.0
27.{Mr. Jones Supper Club} 5.0
28.{Sushi Zo} 4.5
29.{Peter Luger} 4.0
30.Craft 4.0

Module-2

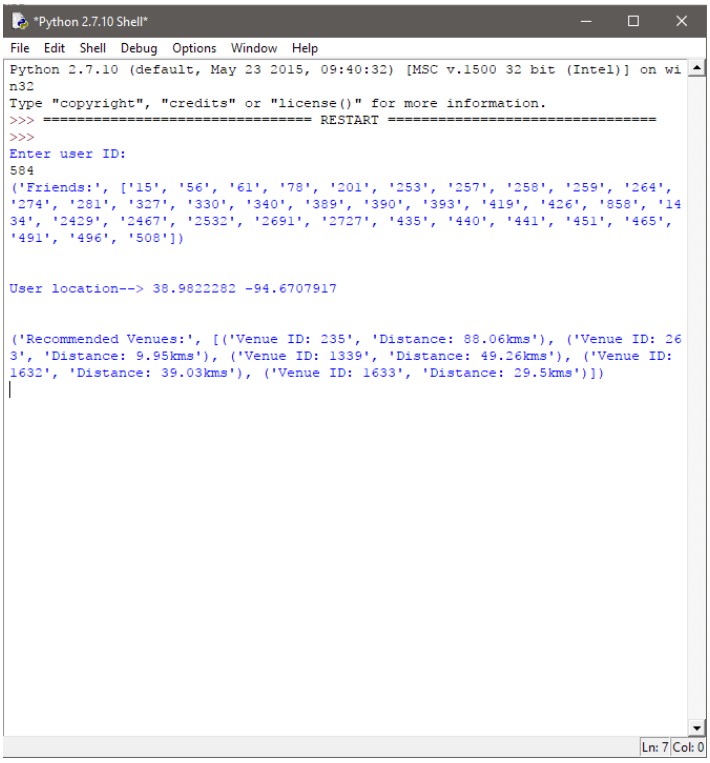
Code:

```
import csv
from math import sin,cos,sqrt,atan2,radians
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import warnings
import sys
import webbrowser as wb

warnings.filterwarnings("ignore")

R = 6373.0
print("Enter user ID:")
ID=raw_input()


with open('socialgraph.csv') as csvfile:
    readCSV=csv.reader(csvfile,delimiter=',')
    friends=[]
    for row in readCSV:
        #print(row)
        if(row[0]==ID):
            friends.append(row[1])

print("Friends:",friends)
print("\n")
with open('ratings.csv') as rate:
    readCSV2=csv.reader(rate,delimiter=',')
    venues=[]
    for row in readCSV2:
        if((row[2]=='5' or '4') and (row[0] in friends)):
            if(row[1] not in venues):
                venues.append(row[1])

with open('user_locate.csv') as user_loc:
    readCSV3=csv.reader(user_loc,delimiter=',')
    u_loc=[]
    for row in readCSV3:
```

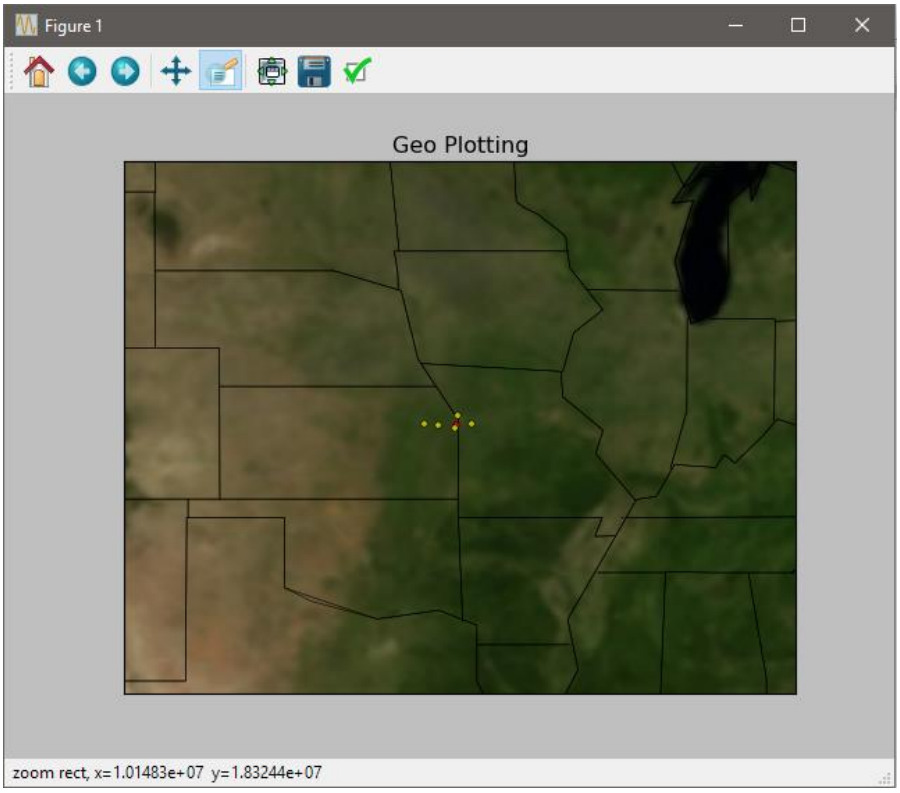
```
        if(row[0]==ID):
            u_loc.append(row[1])
            u_loc.append(row[2])

if len(u_loc)==0:
    print("Data insufficient or wrong ID entered!")
    sys.exit()


with open('venue_locate.csv') as venue_loc:
    readCSV4=csv.reader(venue_loc,delimiter=',')
    v_loc=[]
    for row in readCSV4:
        if(row[0] in venues):
            v_loc.append((row[1],row[2]))

d=[]
dist=[]
for i in v_loc:
    lat1 = radians(float(u_loc[0]))
    lon1 = radians(float(u_loc[1]))
    lat2 = radians(float(i[0]))
    lon2 = radians(float(i[1]))

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    #print(dlon,dlat)

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = round(R * c,2)
    if(distance<100.0):
        d.append((i[0],i[1]))
        dist.append(distance)

with open('venue_locate.csv') as venue_loc:
    readCSV5=csv.reader(venue_loc,delimiter=',')
    v=[]
    for row in readCSV5:
        if (row[1],row[2]) in d:
            v.append(row[0])
```

```
##print(len(venues))
print "User location-->",u_loc[0],u_loc[1]
print "\n"
w=[]
for i in range(0,len(d)):
    w.append(("Venue ID: "+str(v[i]),"Distance: "+str(dist[i])+"kms"))

if(len(d)==0):
    print("Sorry,can't recommend any venues near you!")
else:
    print("Recommended Venues:",w)


wb.open('https://www.google.co.in/maps/dir//'+u_loc[0]+','+u_loc[1]+'/@'+u
_loc[0]+','+u_loc[1]+',17z/data=!4m8!1m7!3m6!1s0x0:0x0!2zNDDCsDAwJzU0LjAiT
iAxMDXCsDE2JzE0LjAiVw!3b1!8m2!3d'+u_loc[0]+'!4d-'+u_loc[1])
    for i in d:

wb.open('https://www.google.co.in/maps/dir//'+i[0]+','+i[1]+'/@'+i[0]+','+
i[1]+',17z/data=!4m8!1m7!3m6!1s0x0:0x0!2zNDDCsDAwJzU0LjAiTiAxMDXCsDE2JzE0L
jAiVw!3b1!8m2!3d'+i[0]+'!4d-'+i[1])



    m = Basemap(projection='mill',llcrnrlat=-90,urcrnrlat=90,\
                llcrnrlon=-180,urcrnrlon=180,resolution='c')
    m.drawcoastlines()
    m.drawcountries()
    m.drawstates()
    #m.fillcontinents(color='green', lake_color='blue')
    m.drawmapboundary(fill_color='#FFFFFF')
    m.bluemarble(scale=0.5)

    ul1=u_loc[0]
    ul2=u_loc[1]
    x,y = m(float(u_loc[1]),float(u_loc[0]))
    m.plot(x,y,'ro',markersize=4)
    for i in range(0,len(d)):
            x,y=m(float(d[i][1]),float(d[i][0]))
            m.plot(x,y,'yo',markersize=4)

    plt.title('Geo Plotting')
    plt.show()
```

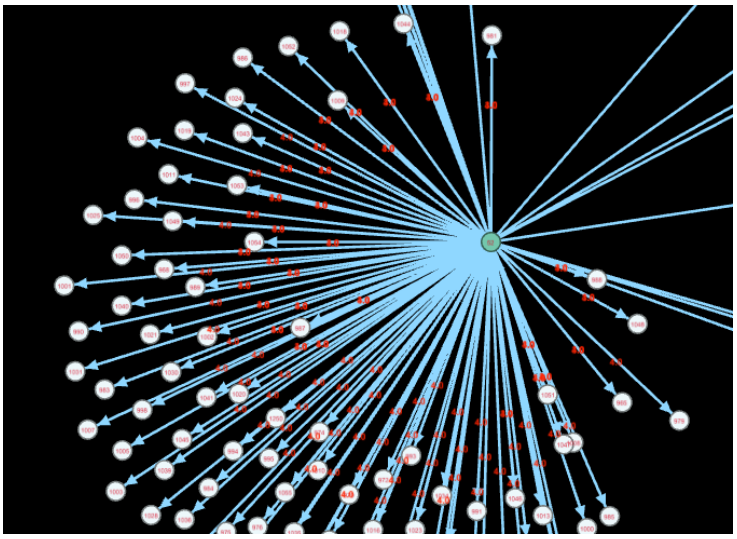
Output:



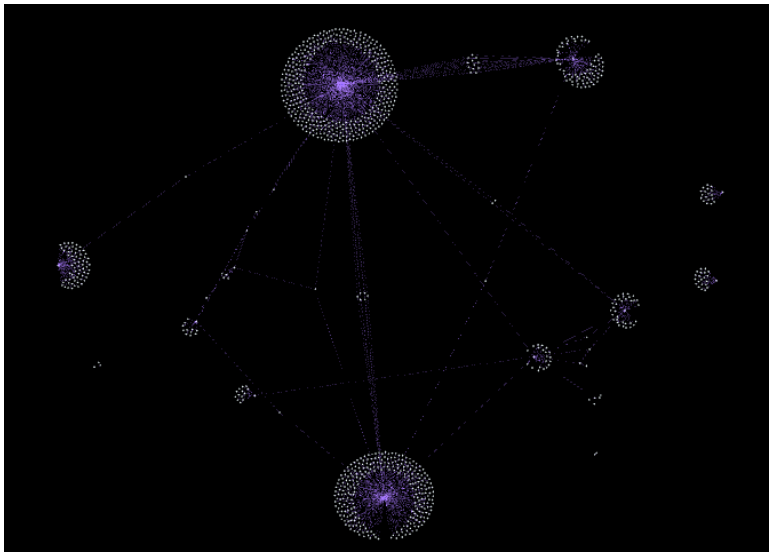
Graphs:



Sociograph (User-User)



Weighted Rating graph
(User-Venue)



CONCLUSION

Module-1

1. The recommendation is based on two sites. Data obtained in run time using web crawlers is more efficient than local data due to dynamic nature of World Wide Web. We gather up to date data using web crawling.
2. Disadvantages of web scraping is internet traffic, storage constraints, bandwidth constraints etc.
3. Since there are not many websites, which provide their source codes, thus we had difficulty in comparing multiple websites. Therefore, Foursquare and Yelp are the only two websites, which provide their codes for scraping.
4. Yelp is not available in India, so it would not show any results for any city in India, which can be another disadvantage.

Module-2

1. The recommendation is based on the dataset obtained from archieve.org, which are stored locally.
2. This recommender recommends venues based on the choices on your friends, assuming you and your friend having similar tastes.
3. Since the recommendation is based on the friends' choices it would not be logical always, that recommendation is good.
4. Scope for this algorithm is to make the recommendation based on the overall ratings not only friends' rating.
5. This system is also not dynamic in nature; it is based on the data stored locally, which is not changing with time.

REFERENCES

- https://archive.org/details/201309_foursquare_dataset_umn
- https://foursquare.com/
- https://yelp.com/
- https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- https://docs.python.org/2/library/tkinter.html