GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

LOCATION BASED TOURISM RECOMMENDED SYSTEM

A Project Report of Capstone Project - 2

Submitted by

MOHIT KUMAR PANI
(1613101417)

in partial fulfillment for the award of the

degree of

BACHELOR OF TECHNOLOGY

IN
COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Dr. D.Rajesh Kumar

APRIL/MAY-2020

GALGOTIAS UNIVERSITY

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"SMART TOURISM RECOMMENDED SYSTEM"** is the bonafide work of "**MOHIT KUMAR PANI (1613101417)"** who carried out the project work under my supervision.

SIGNATURE OF HEAD	**SIGNATURE OF SUPERVISER**
Dr. MUNISH SHABARWAL, PhD (Management), PhD (CS)	**Dr D Rajesh Kumar B.E., M.Tech., PhD.**
Professor & Dean	**Assistant Professor**
School of Computer Science & Engineering	**School of Computer Science & Engineering**

TABLE OF CONTENTS

ABSTRACT

The recent past showed a greater interest in recommender techniques. Now-adays there are many travel packages existing from different websites to almost all the places over the world. A customer finds it very difficult to search for the best package as he/she has to browse multiple websites, contact many travel agents and etc. which is a tedious process and is time consuming. There should be a system where the user should find the best package on the Internet with a single click. Initially, we will evaluate the particular characteristics of the current traveling packages and we mine the data on the tourists rating and the intrinsic features i.e., locations, travel seasons etc. Based on the data collected after mining, we will generate a list for personalized travel package recommendations. Furthermore, we will extract the data based on the tourist's relation with the area and season.

INTRODUCTION

Tourism can be considered as most favorite pass time when people get free time. Several travel organizations are available on the web. The people or the tourist select their own Travel Package according to their personal interest. The travel companies concentrate on the interest associated with tourist making sure to increase their particular market value and supply enormous package deals. So that they can make their Travel Package more effective. Now-a-days Recommender system is becoming very famous and people are getting attracted to it, as it is helping them to choose the best package in a short time. Recommender systems are categorized into

1. Content based system: With this, item recommendation is analyzed then it retrieves the information and filters this for research. For example, if the tourist goes to hill stations more often, then database contains “hill station” as recommendation .

2. Collaborative filtering systems: It rely on the similar factors of user or items. Preferences of different users for same item are recommended by system .

There are many challenges in designing and executing Personalized Travel Package Recommendation System. The following shows some of the challenges:

1. The data for Travel is very less and scattered. For an example, recommendation for a movie may cost more to travel than the movie price.

2. Usually Travel package are location based so they are pertained to space or time to reach destination. For an example, the package contains locations which are geographically near and also vary season wise .

3. The older recommendation method is dependent upon rating and the travel data may not consist of this sort of rating .

1.2 Related Work:

1.2.1 Collaborative Filtering for Orkut Communities: Discovery of User Latent Behavior

"Recommender systems can be classified into two categories:

Content - based filtering and collaborative filtering. Content-based filtering analyzes the association between user problems and the descriptions of items. To recommend new items to a user, the content-based filtering approach matches the new items descriptions to those items known to be of interest to the user. On the other hand, the collaborative filtering (CF) approach does not need content information to make recommendations. Users of social networking services can connect with each other by forming communities for online interaction. Yet as the number of communities hosted by such websites grows over time, 5 users have even greater need for effective community recommendations in order to meet more users. We investigate two algorithms from very different domains and evaluate their effectiveness for personalized community recommendation. The first algorithm is association rule mining (ARM) , which discovers associations between sets of communities that are shared across many users. The second algorithm is Latent Dirichlet allocation (LDA), which models user-community co-occurrences using latent aspects. In comparing LDA with ARM, we are interested in discovering whether modeling lowrank latent structure is more effective

for recommendations than directly mining rules from the observed data. Our empirical comparisons using the top ‘k’ recommendations metric show that LDA performs consistently better than ARM for the community recommendation task when recommending a list of 4 or more communities”.

1.2.2 Equip Tourists with Knowledge Mined from Travelogues:

“With the prosperity of tourism and Web 2.0 technologies, more and more people have willingness to share their travel experiences on the Web (e.g., weblogs, forums, or Web 2.0 communities). These so-called travelogues contain rich information, particularly including location-representative knowledge such as attractions which is Golden Gate Bridge, styles for e.g., beach, history, and activities (diving, surfing). The location representative information in travelogues can greatly facilitate other tourist trip planning, if it can be correctly extracted and summarized. However since most travelogues are unstructured and contain much noise, it is difficult for common users to utilize such knowledge effectively. In this paper, to mine location-representative knowledge from a 6 large collection of travelogues, we adopt a probabilistic topic model, named as Location Topic model. This model has the advantages of distinguishing between two kinds of topics, i.e., local topics which characterize locations and global topics which represent other common themes shared by various locations, and representing locations in the local topic space to encode both location-representative knowledge and similarities between locations.. Based on a large collection of travelogues, the Adopted framework is evaluated using both objective and subjective evaluation methods and shows promising results.”

1.3 Existing System:

1. Travel data is generally less than the data for other items, such as movies for recommendation, the charges for a travel are considerably more costly than watching a movie.
2. Almost every travel package contains numerous landscapes with lots of people's interest and attractions and thus has intrinsic complex spatial-temporal relationships. For an example, a travel package includes landscapes which are geographically aligned together.
3. The existing recommender systems usually rely on data which are gathered and analysed based on the ratings given by the user, but it is not conveniently available for travel data.

Disadvantages:

1. Travel data are much fewer and sparser than traditional items.
2. The traditional items for recommendation usually have a long period of stable value, while the values of travel packages can easily depreciate over time.
3. The real world travel recommendation systems are usually very complicated.
4. Every travel package consists of many landscapes (places of interest and attractions), and thus has intrinsic complex spatial temporal relationship.

1.4 Adopted Solution:

The problem of unique features to distinguish personalized travel package recommendations from traditional recommender systems remains pretty open. There are many technical and domain problems designing and implementing the effective recommender system for personalized travel recommendation system . This project will help tourist to suggest the best Travel Package among all the package deals on the web. In this, a customer will select a travel package for a particular place based on the recommendations provided by the previous customers who had experience with the package. This makes easy for the user to choose the best package deal.

Advantages:

1. The Travel Packages will be presented based on the interest of the tourist.
2. By using tourist, area and season as our inputs we can represent our travel data in the best form.
3. By using this recommendation approach the flaws of the existing system will be eliminated as it performs much better than traditional techniques.
4. The algorithm ‘Weighted Average Entropy’ will help the tourist to find the best package in the particular area based on season and theme .

NARRATIVE

2.1 Problem Statement:

There are several packages available for a travel system on the web . In order to select the best package to certain destination, there is no efficient recommendation system available. To overcome this problem, we are coming up with Travel Package Recommendation System where you can select the best package.

2.2 Motivation:

A tourist has to select a package based on season and location. For example, if a tourist wants to visit ‘Landscapes’ in ‘Winter’ season, then there will an option of choosing place and season. Therefore, through this a tourist can customize their package accordingly. This feature is implemented by using Tourist, Area and Season Model which can effectively capture the unique characteristics of travel data and also captures the relationships among the tourists which implements the better performance of travel package recommendation. This approach is much better than the traditional techniques . The goal of the personalized travel package recommendation represents the Travel Packages and interest of the tourists .

PROPOSED SYSTEM

3.1 Proposed system

The starting point of collaborative filtering is:

Suppose that users with similar interests should favorite to the same items as each other. So, as long as the maintenance of a database on the user's preference, the neighbor users with similar interests can be calculated by analyzing the stored preference, and then it can be recommended to the user based on the neighbor users' interest. On the basis of collaborative filtering principle, the recommendation process of tourist attractions can be divided into three steps.

i. The representation of user (tourist) information. The visiting history of attractions by tourist need to be analyzed and modeled.

ii. The generation of neighbor users (tourists). The similarity of tourists can be computed according to the visiting history data and the collaborative filtering algorithm presented by us. A neighbor tourist list can be calculated on the basis of known similarities.

iii. The generation of attraction recommendations. Top- N attractions will recommend to the tourist according to the visiting history of his neighbors.

SYSTEM ARCHITECTURE

The primary objective of our framework is to give a customized arrangement to a given vacationer. This subsequent arrangement needs to mirror the inclinations of the vacationer as per his/her profile. Additionally, to manufacture this plan, the term of the exercises to play out, the opening hours of the spots to visit and the topographical separations between places needs likewise to be thought of. Hence, tackling this issue requires the utilization of an arranging framework equipped for managing durative activities to speak to the term of visits; worldly requirements to communicate the opening times of spots and delicate objectives for the client inclinations.

SYSTEM IMPLEMENTATION

Firstly, we have to extract the data sets based on content and collaboration. We have to match the cosines and the ratings in content filtering and also locate the API. In collaborative filtering, we have to find the correlation between the users and have to find the k-nearest neighbour. The dataset in the collaborative used is stored in an excel file “data-collaborative” it contains userid, itemid and the rating and the rating date. Content data is stored in the excel file “datacontent” it has itemid, title, category and the prating. The datasets are extracted from web scraping technique to extract a lot of information from sites whereby the information is removed and spared to a neighbourhood record and python framework is used to extract data efficiently .Datasets are preprocessed using NumPy and pandas.

SOURCE CODE

(Content)

```
import pandas as pd
import numpy as np
import re, math
from collections import Counter
from googlemaps import convert
from googlemaps import Client
from googlemaps.convert import as_list

WORD = re.compile(r'\w+')

#applying cosine similarity for finding similarities between user interests and places
def get_cosine(vec1, vec2):
    intersection = set(vec1.keys()) & set(vec2.keys())
    numerator = sum([vec1[x] * vec2[x] for x in intersection])
    sum1 = sum([vec1[x]**2 for x in vec1.keys()])
    sum2 = sum([vec2[x]**2 for x in vec2.keys()])
    denominator = math.sqrt(sum1) * math.sqrt(sum2)
    if not denominator:
      return 0.0
    else:
```

```
        return float(numerator) / denominator


def text_to_vector(text):
     words = WORD.findall(text)
     return Counter(words)


#remove spaces from the category column of dataset
def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
        else:
            return ''


#calulating weighted rating of places




metadata = pd.read_csv('data_content.csv', low_memory=False)
#print(metadata.head())
print("Select your preferred category:\n1.wildlife \n2.heritage \n3.pilgirmage\n4.park\n5.museum")
text1 = input("Enter User Interests: ")   #user preference
```

```
vector1 = text_to_vector(text1)
C = metadata['p_rating'].mean()
m = metadata['count'].quantile(0.75)

def weighted_rating(x, m=m, C=C):
    v = x['count']
    R = x['p_rating']
    # Calculation based on the Bayesian Rating Formula
    return (v/(v+m) * R) + (m/(m+v) * C)

metadata['category'] = metadata['category'].apply(clean_data)
metadata['score'] = metadata.apply(weighted_rating, axis=1)
#print(metadata.head())
cos=[]
for i in list(metadata['category']):
    #print(type(i))
    text2 = i
    vector2 = text_to_vector(text2)
    cosine = get_cosine(vector1, vector2)
    cos.append(cosine)
metadata['cosine']=cos
x=metadata['cosine']>0.0
rec=pd.DataFrame(metadata[x])
rec=rec.sort_values('score',ascending=False)
```

```
src=input("Enter your location: ")
dest=list(rec['title'])
#print(type(dest))
    def distance_matrix(client,origins, destinations,
                        mode=None, language=None, avoid=None, units=None,
                        departure_time=None, arrival_time=None, transit_mode=None,
                        transit_routing_preference=None, traffic_model=None, region=None):
    params = {
        "origins": convert.location_list(origins),
        "destinations": convert.location_list(destinations)
    }

    if mode:
        # NOTE(broady): the mode parameter is not validated by the Maps API
        # server. Check here to prevent silent failures.
        if mode not in ["driving", "walking", "bicycling", "transit"]:
            raise ValueError("Invalid travel mode.")
        params["mode"] = mode

    if language:
        params["language"] = language

    if avoid:
        if avoid not in ["tolls", "highways", "ferries"]:
```

```
        raise ValueError("Invalid route restriction.")
    params["avoid"] = avoid

if units:
    params["units"] = units

if departure_time:
    params["departure_time"] = convert.time(departure_time)

if arrival_time:
    params["arrival_time"] = convert.time(arrival_time)

if departure_time and arrival_time:
    raise ValueError("Should not specify both departure_time and"
                     "arrival_time.")

if transit_mode:
    params["transit_mode"] = convert.join_list("|", transit_mode)

if transit_routing_preference:
    params["transit_routing_preference"] = transit_routing_preference

if traffic_model:
    params["traffic_model"] = traffic_model
```

```
    if region:
        params["region"] = region
    #print(client._request("/maps/api/distancematrix/json", params))
    return client._request("/maps/api/distancematrix/json", params)

client = Client(key='AIzaSyA8wq3R8WASxgUqTvWCh5blEmGzU8njVZ0')
dist=[]
dur=[]
for d in dest:
    d=d+",Jaipur"
    #print(d)
    output=distance_matrix(client,src,d)
    #print(output)
    a1=(output['rows'][0]['elements'][0]['distance']['text'])
    a2=(output['rows'][0]['elements'][0]['duration']['text'])
    dist.append(a1)
    dur.append(a2)

rec['distance']=dist
rec['duration']=dur

final=pd.DataFrame(rec,index=None,columns=['title','category','score','distance','duration'])
print(final)
```

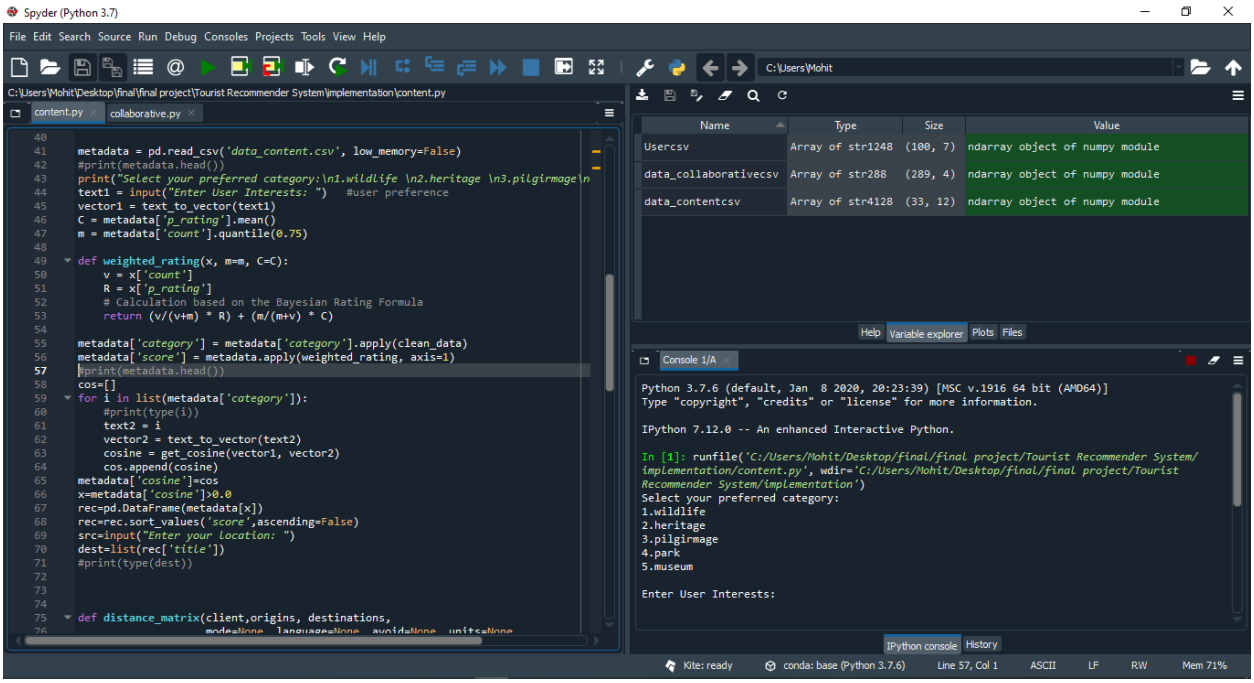
(Collaborative)

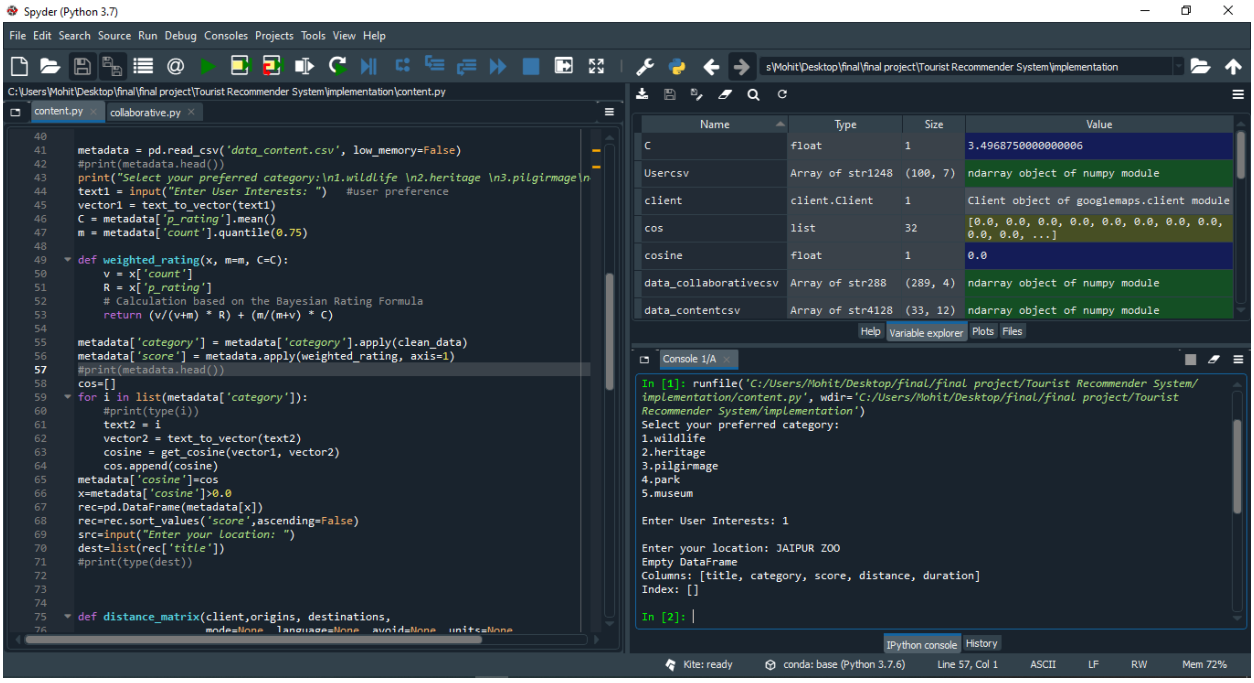
```
import numpy as np
import pandas as pd
import scipy.sparse
from scipy.spatial.distance import correlation

data=pd.read_csv('data_collaborative.csv')
placeInfo=pd.read_csv('data_content.csv')

data=pd.merge(data,placeInfo,left_on='itemId',right_on="itemId")
userIds=data.userId
userIds2=data[['userId']]

data.loc[0:10,['userId']]
data=pd.DataFrame.sort_values(data,['userId','itemId'],ascending=[0,1])



def favoritePlace(activeUser,N):
    topPlace=pd.DataFrame.sort_values(
        data[data.userId==activeUser],['rating'],ascending=[0])[:N]
    return list(topPlace.title)

userItemRatingMatrix=pd.pivot_table(data, values='rating',
                                    index=['userId'], columns=['itemId'])



def similarity(user1,user2):
    try:
```

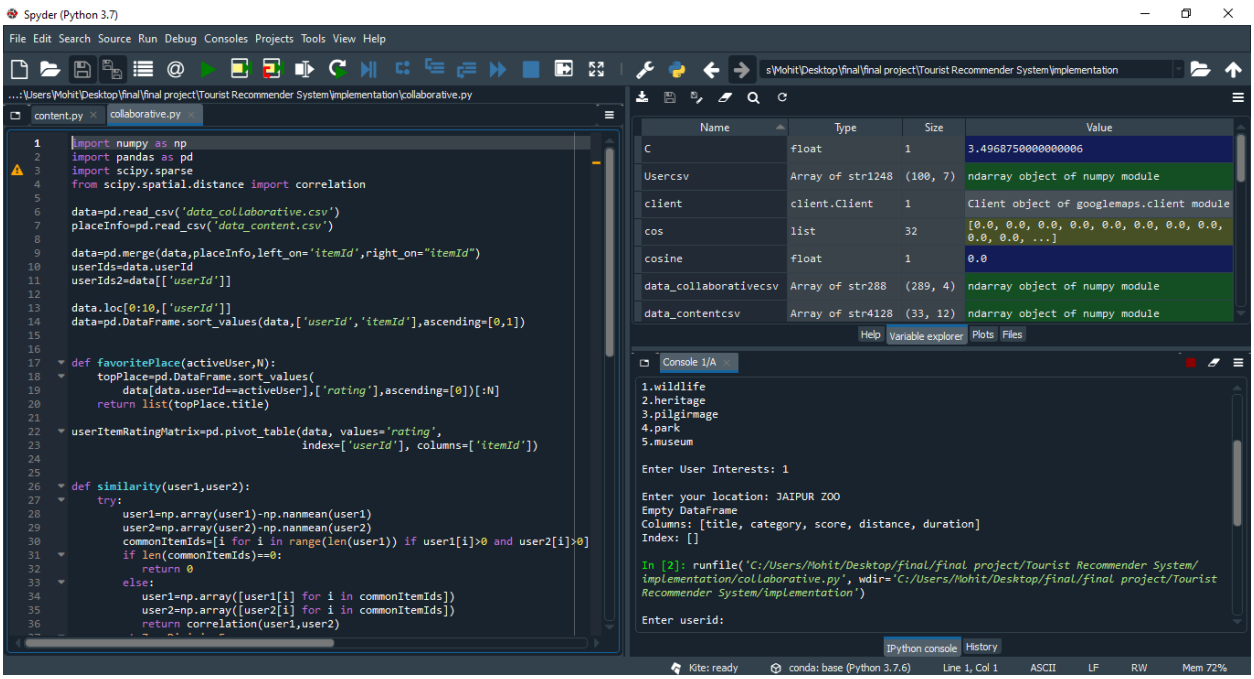
```
        user1=np.array(user1)-np.nanmean(user1)
        user2=np.array(user2)-np.nanmean(user2)
        commonItemIds=[i for i in range(len(user1)) if user1[i]>0 and user2[i]>0]
        if len(commonItemIds)==0:
          return 0
        else:
          user1=np.array([user1[i] for i in commonItemIds])
          user2=np.array([user2[i] for i in commonItemIds])
          return correlation(user1,user2)
    except ZeroDivisionError:
        print("You can't divide by zero!")



def nearestNeighbourRatings(activeUser,K):
    try:
        similarityMatrix=pd.DataFrame(index=userItemRatingMatrix.index,columns=['Similarity'])
        for i in userItemRatingMatrix.index:

similarityMatrix.loc[i]=similarity(userItemRatingMatrix.loc[activeUser],userItemRatingMatrix.l
oc[i])
        similarityMatrix=pd.DataFrame.sort_values(similarityMatrix,['Similarity'],ascending=[0])
        nearestNeighbours=similarityMatrix[:K]
        neighbourItemRatings=userItemRatingMatrix.loc[nearestNeighbours.index]
        predictItemRating=pd.DataFrame(index=userItemRatingMatrix.columns,
columns=['Rating'])
        for i in userItemRatingMatrix.columns:
            predictedRating=np.nanmean(userItemRatingMatrix.loc[activeUser])
            for j in neighbourItemRatings.index:
                if userItemRatingMatrix.loc[j,i]>0:
```

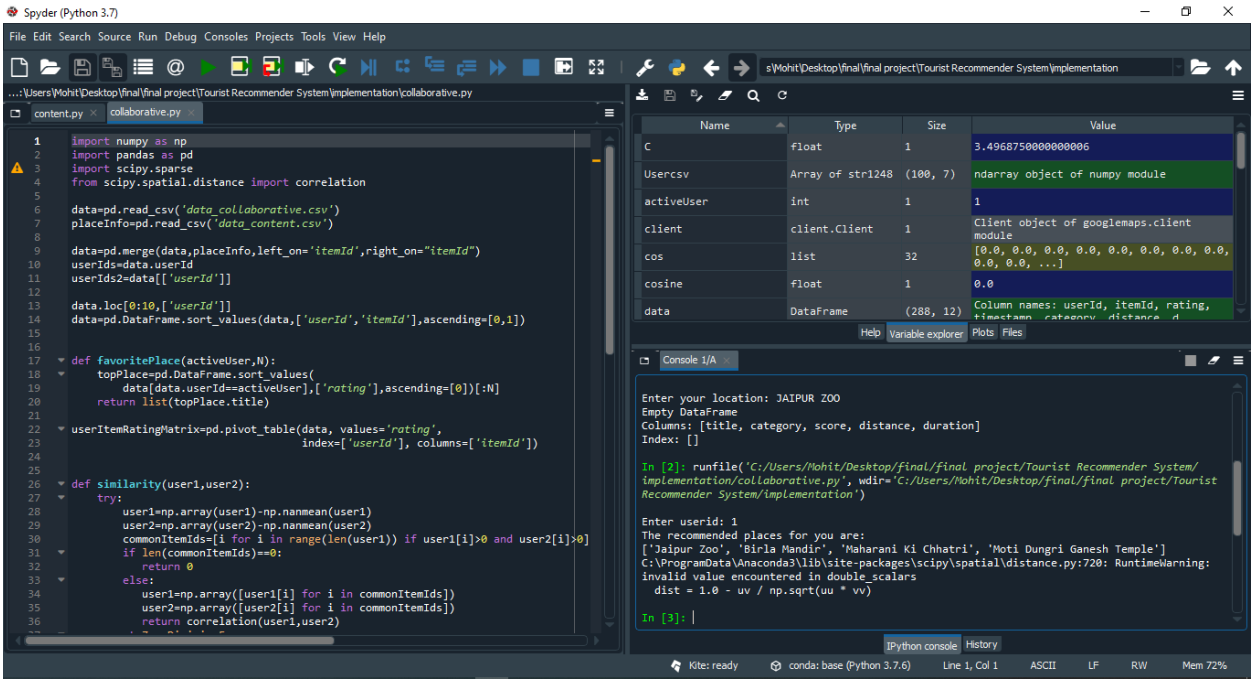
```
                predictedRating += (userItemRatingMatrix.loc[j,i]-
np.nanmean(userItemRatingMatrix.loc[j]))*nearestNeighbours.loc[j,'Similarity']
            predictItemRating.loc[i,'Rating']=predictedRating
    except ZeroDivisionError:
        print("You can't divide by zero!")
    return predictItemRating


def topNRecommendations(activeUser,N):
    try:
        predictItemRating=nearestNeighbourRatings(activeUser,10)
        placeAlreadyWatched=list(userItemRatingMatrix.loc[activeUser]
                        .loc[userItemRatingMatrix.loc[activeUser]>0].index)
        predictItemRating=predictItemRating.drop(placeAlreadyWatched)
        topRecommendations=pd.DataFrame.sort_values(predictItemRating,
                                ['Rating'],ascending=[0])[:N]

topRecommendationTitles=(placeInfo.loc[placeInfo.itemId.isin(topRecommendations.index)])
    except ZeroDivisionError:
        print("You can't divide by zero!")
    return list(topRecommendationTitles.title)


activeUser=int(input("Enter userid: "))
#print("The user's favorite places are: ")
#print(favoritePlace(activeUser,5))
print("The recommended places for you are: ")
print(topNRecommendations(activeUser,4))
```

OUTPUTS


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Mohit\Desktop\final\final project\Tourist Recommender System\implementation\content.py
content.py
collaborative.py
metadata = pd.read_csv('data_content.csv', low_memory=False)
#print(metadata.head())
print("Select your preferred category:\n1.wildlife \n2.heritage \n3.pilgirmage\n
text1 = input("Enter User Interests: ") #user preference
vector1 = text_to_vector(text1)
C = metadata['p_rating'].mean()
m = metadata['count'].quantile(0.75)
def weighted_rating(x, m=m, C=C):
v = x['count']
R = x['p_rating']
# Calculation based on the Bayesian Rating Formula
return (v/(v+m) * R) + (m/(m+v) * C)
metadata['category'] = metadata['category'].apply(clean_data)
metadata['score'] = metadata.apply(weighted_rating, axis=1)
#print(metadata.head())
cos=[]
for i in list(metadata['category']):
#print(type(i))
text2 = i
vector2 = text_to_vector(text2)
cosine = get_cosine(vector1, vector2)
cos.append(cosine)
metadata['cosine']=cos
x=metadata['cosine']>0.0
rec=pd.DataFrame(metadata[x])
rec=rec.sort_values('score',ascending=False)
src=input("Enter your location: ")
dest=list(rec['title'])
#print(type(dest))
def distance_matrix(client,origins, destinations,
C:\Users\Mohit
Name Type Size Value
Usercsv Array of str1248 (100, 7) ndarray object of numpy module
data_collaborativecsv Array of str288 (289, 4) ndarray object of numpy module
data_contentcsv Array of str4128 (33, 12) ndarray object of numpy module
Help Variable explorer Plots Files
Console 1/A
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 7.12.0 -- An enhanced Interactive Python.
In [1]: runfile('C:/Users/Mohit/Desktop/final/final project/Tourist Recommender System/implementation/content.py', wdir='C:/Users/Mohit/Desktop/final/final project/Tourist Recommender System/implementation')
Select your preferred category:
1.wildlife
2.heritage
3.pilgirmage
4.park
5.museum
Enter User Interests:
IPython console History
Kite: ready
conda: base (Python 3.7.6)
Line 57, Col 1
ASCII
LF
RW
Mem 71%


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
s\Mohit\Desktop\final\final project\Tourist Recommender System\implementation
C:\Users\Mohit\Desktop\final\final project\Tourist Recommender System\implementation\content.py
content.py
collaborative.py
metadata = pd.read_csv('data_content.csv', low_memory=False)
#print(metadata.head())
print("Select your preferred category:\n1.wildlife \n2.heritage \n3.pilgirmage\n
text1 = input("Enter User Interests: ") #user preference
vector1 = text_to_vector(text1)
C = metadata['p_rating'].mean()
m = metadata['count'].quantile(0.75)
def weighted_rating(x, m=m, C=C):
v = x['count']
R = x['p_rating']
# Calculation based on the Bayesian Rating Formula
return (v/(v+m) * R) + (m/(m+v) * C)
metadata['category'] = metadata['category'].apply(clean_data)
metadata['score'] = metadata.apply(weighted_rating, axis=1)
#print(metadata.head())
cos=[]
for i in list(metadata['category']):
#print(type(i))
text2 = i
vector2 = text_to_vector(text2)
cosine = get_cosine(vector1, vector2)
cos.append(cosine)
metadata['cosine']=cos
x=metadata['cosine']>0.0
rec=pd.DataFrame(metadata[x])
rec=rec.sort_values('score',ascending=False)
src=input("Enter your location: ")
dest=list(rec['title'])
#print(type(dest))
def distance_matrix(client,origins, destinations,
Name Type Size Value
C float 1 3.4968750000000006
Usercsv Array of str1248 (100, 7) ndarray object of numpy module
client client.Client 1 Client object of googlemaps.client module
cos list 32 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...]
cosine float 1 0.0
data_collaborativecsv Array of str288 (289, 4) ndarray object of numpy module
data_contentcsv Array of str4128 (33, 12) ndarray object of numpy module
Help Variable explorer Plots Files
Console 1/A
In [1]: runfile('C:/Users/Mohit/Desktop/final/final project/Tourist Recommender System/implementation/content.py', wdir='C:/Users/Mohit/Desktop/final/final project/Tourist Recommender System/implementation')
Select your preferred category:
1.wildlife
2.heritage
3.pilgirmage
4.park
5.museum
Enter User Interests: 1
Enter your location: JAIPUR ZOO
Empty DataFrame
Columns: [title, category, score, distance, duration]
Index: []
In [2]:
IPython console History
Kite: ready
conda: base (Python 3.7.6)
Line 57, Col 1
ASCII
LF
RW
Mem 72%


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
...:\Users\Mohit\Desktop\final\final project\Tourist Recommender System\implementation\collaborative.py
content.py
collaborative.py
import numpy as np
import pandas as pd
import scipy.sparse
from scipy.spatial.distance import correlation
data=pd.read_csv('data_collaborative.csv')
placeInfo=pd.read_csv('data_content.csv')
data=pd.merge(data,placeInfo,left_on='itemId',right_on="itemId")
userIds=data.userId
userIds2=data[['userId']]
data.loc[0:10,['userId']]
data=pd.DataFrame.sort_values(data,['userId','itemId'],ascending=[0,1])
def favoritePlace(activeUser,N):
topPlace=pd.DataFrame.sort_values(
data[data.userId==activeUser],['rating'],ascending=[0])[:N]
return list(topPlace.title)
userItemRatingMatrix=pd.pivot_table(data, values='rating',
index=['userId'], columns=['itemId'])
def similarity(user1,user2):
try:
user1=np.array(user1)-np.nanmean(user1)
user2=np.array(user2)-np.nanmean(user2)
commonItemIds=[i for i in range(len(user1)) if user1[i]>0 and user2[i]>0]
if len(commonItemIds)==0:
return 0
else:
user1=np.array([user1[i] for i in commonItemIds])
user2=np.array([user2[i] for i in commonItemIds])
return correlation(user1,user2)
s\Mohit\Desktop\final\final project\Tourist Recommender System\implementation
Name Type Size Value
C float 1 3.4968750000000006
Usercsv Array of str1248 (100, 7) ndarray object of numpy module
client client.Client 1 Client object of googlemaps.client module
cos list 32 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...]
cosine float 1 0.0
data_collaborativecsv Array of str288 (289, 4) ndarray object of numpy module
data_contentcsv Array of str4128 (33, 12) ndarray object of numpy module
Help Variable explorer Plots Files
Console 1/A
1.wildlife
2.heritage
3.pilgirmage
4.park
5.museum
Enter User Interests: 1
Enter your location: JAIPUR ZOO
Empty DataFrame
Columns: [title, category, score, distance, duration]
Index: []
In [2]: runfile('C:/Users/Mohit/Desktop/final/final project/Tourist Recommender System/implementation/collaborative.py', wdir='C:/Users/Mohit/Desktop/final/final project/Tourist Recommender System/implementation')
Enter userid:
IPython console History
Kite: ready
conda: base (Python 3.7.6)
Line 1, Col 1
ASCII
LF
RW
Mem 72%


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
...:\Users\Mohit\Desktop\final\final project\Tourist Recommender System\implementation\collaborative.py
content.py
collaborative.py
import numpy as np
import pandas as pd
import scipy.sparse
from scipy.spatial.distance import correlation
data=pd.read_csv('data_collaborative.csv')
placeInfo=pd.read_csv('data_content.csv')
data=pd.merge(data,placeInfo,left_on='itemId',right_on="itemId")
userIds=data.userId
userIds2=data[['userId']]
data.loc[0:10,['userId']]
data=pd.DataFrame.sort_values(data,['userId','itemId'],ascending=[0,1])
def favoritePlace(activeUser,N):
topPlace=pd.DataFrame.sort_values(
data[data.userId==activeUser],['rating'],ascending=[0])[:N]
return list(topPlace.title)
userItemRatingMatrix=pd.pivot_table(data, values='rating',
index=['userId'], columns=['itemId'])
def similarity(user1,user2):
try:
user1=np.array(user1)-np.nanmean(user1)
user2=np.array(user2)-np.nanmean(user2)
commonItemIds=[i for i in range(len(user1)) if user1[i]>0 and user2[i]>0]
if len(commonItemIds)==0:
return 0
else:
user1=np.array([user1[i] for i in commonItemIds])
user2=np.array([user2[i] for i in commonItemIds])
return correlation(user1,user2)
s\Mohit\Desktop\final\final project\Tourist Recommender System\implementation
Name Type Size Value
C float 1 3.4968750000000006
Usercsv Array of str1248 (100, 7) ndarray object of numpy module
activeUser int 1 1
client client.Client 1 Client object of googlemaps.client module
cos list 32 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...]
cosine float 1 0.0
data DataFrame (288, 12) Column names: userId, itemId, rating,
Help Variable explorer Plots Files
Console 1/A
Enter your location: JAIPUR ZOO
Empty DataFrame
Columns: [title, category, score, distance, duration]
Index: []
In [2]: runfile('C:/Users/Mohit/Desktop/final/final project/Tourist Recommender System/implementation/collaborative.py', wdir='C:/Users/Mohit/Desktop/final/final project/Tourist Recommender System/implementation')
Enter userid: 1
The recommended places for you are:
['Jaipur Zoo', 'Birla Mandir', 'Maharani Ki Chhatri', 'Moti Dungri Ganesh Temple']
C:\ProgramData\Anaconda3\lib\site-packages\scipy\spatial\distance.py:720: RuntimeWarning:
invalid value encountered in double_scalars
dist = 1.0 - uv / np.sqrt(uu * vv)
In [3]:
IPython console History
Kite: ready
conda: base (Python 3.7.6)
Line 1, Col 1
ASCII
LF
RW
Mem 72%

CONCLUSION

This project will help to suggest the best Travel package among all the package deals on the web. In this, a customer will select a travel package for a particular place based on the recommendations provided by the previous customers who had experience with the package. This makes easy for the user to choose the best package deal. The user can select the best package in short amount of time (instead of navigating to other websites). Finally, the goal of the project is to make an efficient system which is effective in terms of cost and money.

Future Work

1. It can be made as a mobile app for platforms Android and IOS.

2. It can be used to solve other similar problems such as flight deals, best university and so on. 3. Festival as an input can be added.

4. Best Hotels in the recommended area can also be included.

REFRENCES

[1] D. Bogdanov, M. Haro, F. Fuhrmann, A. Xambo, E. Gomez and P. Herrera, "A content-based system for music recommendation and visualization of user preferences working on semantic notions," 2013.

[2] Q. Liu, E. Chen, H. Xiong, Y. Ge and Z. Li, "A Cocktail approach for Travel Package Recommendation System," 2014.

[3] G. Fischer, O. Omotoso, G. Chen and J. Evans, "Availability estimation for facilities in extreme geographical locations," 2012.

[4] X. Ye, "Dealing with Unfair Ratings," 2013.

[5] S. Solanki and J. Patel, "A Survey on Association Rule Mining," 2015.

[6] S. Bhutada, V. Balaram and V. Bulusu, "Latent Dirichlet Allocation based multilevel classification," 2014.

[7] X.-l. Zheng, C.-C. Chen, J.-L. Hung and W. He, "A Hybrid Trust-based Recommender System for Online Communities of Practice," 2015.

[8] C. Grun, H. Werthner, B. Proll and W. Retschitzegger, "Assisting Tourists on the Move- An Evaluation of Mobile Tourist Guides," 2012.

[9] Y. Ge, Z. Li and E. Chen, "Personalized Travel Package Recommendation," 2011.

[10] "Tourism Existing System," [Online]. Available: www.orbitz.com/vacations/.