PROJECT REPORT

16 May 2023

—

Recommender System

—

Sir Zain UL Hassan

Project Title:

RESTAURANT RECOMMENDATION SYSTEM

Group Members:

- Ali Jodat (20K-0155)
- Basil Ali Khan (20K-0477)
- Muhammad Umer (20K-0225)

Introduction:

In today's digital age, the abundance of online restaurant information can overwhelm users when it comes to choosing a suitable place to dine. To address this challenge, we have developed a restaurant recommendation system using the K-Nearest Neighbors (KNN) algorithm and correlation similarities between users. The system aims to provide personalized recommendations based on user likings enhancing the dining experience for individuals.

Problem Statement:

The goal of this project is to create a restaurant recommendation system that suggests relevant dining options based on likings, restaurants similarities and popularity. The system takes into account the ratings in dataset provided by users to calculate similarities between them. By simply calculating most visited and rated restaurants percentages, KNN algorithm and correlation similarities, the system can recommend restaurants that align with a user's preferences.

Methodology:

Project involves following steps:

a. **Data Collection:** Restaurant data is gather from various sources such as online review platforms, social media, and user feedback. The collected data includes attributes like restaurant name, location, cuisine, ratings, and user reviews.
b. **Data Preprocessing:** The collected data is cleaned and transformed to ensure consistency and remove any irrelevant information. This involves tasks such as removing duplicates, handling missing values, and standardizing the data format.
c. **Popular Restaurants:** Calculated sum, mean and percentages to recommend most visited and rated restaurants (You may have visited web/apps and there are recommendations of popular restaurants)
d. **KNN Algorithm**: The KNN algorithm is apply to find the K nearest neighbors for a given user based on similarity scores. The neighbors' ratings are then use to make predictions for restaurants that the user has not yet rated.
e. **Similarity Correlation:** Similarity between users is calculate using the correlation coefficient. This measure captures the correlation between the rating patterns of two users. Higher correlation indicates similar preferences.
f. **Restaurant Recommendation:** Based on:
    i. Popularity (Most Visited and rated)
    ii. K Nearest Neighbor (N number of Similar Restaurants)
    iii. Correlation (Similar Users)

Programming Language and Environment:

- ✓ Python Programming Language
- ✓ Google Colaboratory

Implementation:

The recommendation system implemented using a programming language such as Python. The following code snippets shows implementation:

- ✓ **Datasets**
  - geoplace2.csv: Containing location of restaurants and place ID

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	placeID	latitude	longitude	the_geom_	name	address	city	state	country	fax	zip	alcohol	smoking_a	dress_cod	accessibil	price	url	Rambienc	franchise	area	other_services	
2	134999	18.9154	-99.1849	01010000	Kiku Cuer	Revolucio	Cuernava	Morelos	Mexico	?	?	No_Alcoh	none	informal	no_access	medium	kikucuern	familiar	f	closed	none	
3	132825	22.1474	-100.983	01010000	puesto de	esquina s	s.l.p.	s.l.p.	mexico	?	78280	No_Alcoh	none	informal	completel	low	?	familiar	f	open	none	
4	135106	22.1497	-100.976	01010000	El Rinci¿½	Universid	San Luis P	San Luis P	Mexico	?	78000	Wine-Beer	only at ba	informal	partially	medium	?	familiar	f	open	none	
5	132667	23.7527	-99.1634	01010000	little pizz	calle emil	victoria	tamaulipa	?	?	?	No_Alcoh	none	informal	completel	low	?	familiar	t	closed	none	
6	132613	23.7529	-99.1651	01010000	carnitas_r	lic. Emilio	victoria	Tamaulipa	Mexico	?	?	No_Alcoh	permitted	informal	completel	medium	?	familiar	t	closed	none	
7	135040	22.1356	-100.97	01010000	Restauran	Camino a	San Luis P	SLP	Mexico	?	74000	Wine-Beer	none	informal	no_access	high	?	familiar	f	closed	none	

  - ratings_final.csv: Containing ratings of restaurant done by users

	A	B	C	D	E	F
1	userID	placeID	rating	food_ratir	service_rating	
2	U1077	135085	2	2	2	
3	U1077	135038	2	2	1	
4	U1077	132825	2	2	2	
5	U1077	135060	1	2	2	
6	U1068	135104	1	1	2	

- ✓ **Libraries**

```
# import libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import csv
import io
```

- ✓ **Preprocessing**

```
df1=pd.read_csv("geoplaces2.csv")
df2=pd.read_csv("rating_final.csv")
ratings=pd.merge(df1,df2) # merge two df's
ratings=ratings[['placeID','name','userID','rating']] # take needed columns
ratings['userID'] = ratings['userID'].str[1:]
ratings.dropna(inplace=True)
ratings.head() # show new dataframe
```

✓ **Popularity Based Recommendation**

```
[ ]  # function to calculate popularity stats
     def popularity_based_rec(df, group_col, rating_col):
         # group by title and get size, sum and mean values
         grouped = df.groupby(group_col).agg({rating_col: [np.size, np.sum, np.mean]})
         # most popular mean value on top
         popular = grouped.sort_values((rating_col, "sum"), ascending=False)
         return popular
```

```
[ ]  # call function and show top 5 restaurants
     popularity_stats = popularity_based_rec(ratings, "name", "rating")
     popularity_stats.head(10) # show top 5 restaurants
```

✓ **KNN Algorithm**

```
# function that finds the distance of an item from another item - SIMILARITY
def ComputeDistance(a, b):
    # Find the common ratings(by common user) for both item
    common_ratings = [rating for rating in a['ratings'] if rating in b['ratings']]

    # If there are no common ratings, the distance is infinity
    if len(common_ratings) == 0:
        return float('inf')

    # If the lists of ratings are not the same length, return infinity
    if len(a['ratings']) != len(b['ratings']):
        return float('inf')

    # Calculate the sum of the squared differences between the ratings
    sum_squared_differences = sum([(a['ratings'][i] - b['ratings'][i]) ** 2 for i in range(len(common_ratings))])

    # Return the square root of the sum of squared differences, which is the distance between the two items
    return sum_squared_differences ** 0.5
```

```
[ ]  # function to get K-Nearest Neighbors
     def getNeighbors(itemID, K):
         # Get the item object for the given item ID
         target_item = itemDict[itemID]
```

✓ **Correlation Similarity**

```
[ ]  #show user with most correlation with user 1001
     user = userratings[1001]
     corr_users = userratings.corrwith(user).sort_values(ascending=False).to_frame('corr').dropna()
     corr_users
```

```
[ ]  #Prediction function
     def predict(user_pred, correlated):

       common_indexes = list(user_pred.index.intersection(correlated.index))
       top_indexes = user_pred.loc[common_indexes].nlargest(2).index.tolist()

       if top_indexes==0:
         print('PREDICTION NOT POSSIBLE DUE TO LACK OF DATA')
         return 0

       Rating = 0
       Rating_numerator = 0
       Rating_denominator = 0
       #print(top_indexes)

       for similar_user in top_indexes:
         Rating_numerator = Rating_numerator + (user_pred.loc[similar_user]*correlated.loc[similar_user])
         Rating_denominator = Rating_denominator + abs(correlated.loc[similar_user])

       Rating = Rating_numerator/Rating_denominator
       return Rating
```

Conclusion:

In this project, we have developed a restaurant recommendation system using the KNN algorithm and correlation similarities between users. The system leverages user ratings to suggest personalized restaurant recommendations. By employing this system, users can easily discover new dining options that align with their preferences and enhance their overall dining experience. Further enhancements can be made by incorporating additional features like location-based recommendations, user feedback analysis, and real-time data updates.

Future Directions:

To build more friendly graphical interfaces, the next goal for a further project is to improve the performance of System and build an application for this and recommend based on geographical places. Few suggestions that might improve your user based recommender system. One is to normalize utility matrix before taking the dot product. This will help to account for any scale differences between users (some users may tend to rate items higher or lower than others do). Might also use a technique called singular value decomposition (SVD) to decompose your utility matrix into three matrices. This can help to reduce the noise in your data and improve the accuracy of your predictions