Web Science: Assignment #6

Alexander Nwala

Mohd. Nauman Siddique

Sunday, March 31, 2019

Contents

About the data set used in the assignment

The goal of this project is to use the basic recommendation principles we have learned for user-collected data. You will modify the code (https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py) given to you which performs movie recommendations from the MovieLense data sets.
The MovieLense data sets were collected by the GroupLens Research Project at the University of Minnesota during the seven-month period from September 19th, 1997 through April 22nd, 1998. We are using the "100k dataset"; available for download from: http://grouplens.org/datasets/movielens/100k/
There are three files which we will use:

1. u.data: 100,000 ratings by 943 users on 1,682 movies. Each user has rated at least 20 movies. Users and items are numbered consecutively from 1. The data is randomly ordered. This is a tab separated list of

   user id — item id — rating — timestamp

   The time stamps are unix seconds since 1/1/1970 UTC.

   Example:

   196 242 3 881250949

   186 302 3 891717742

   22 377 1 878887116

   244 51 2 880606923

   166 346 1 886397596

   298 474 4 884182806

   115 265 2 881171488

2. u.item: Information about the 1,682 movies. This is a tab separated list of

   movie id — movie title — release date — video release date — IMDb URL — unknown — Action — Adventure — Animation —Children's — Comedy — Crime — Documentary — Drama — Fantasy — Film-Noir — Horror — Musical — Mystery — Romance — Sci-Fi — Thriller — War — Western —

   The last 19 fields are the genres, a 1 indicates the movie is of that genre, a 0 indicates it is not; movies can be in several genres at once. The movie ids are the ones used in the u.data data set.

   Example:

   161—Top Gun (1986)—01-Jan-1986——http://us.imdb.com/M/title-exact?Top%20Gun%20(1986)—0—1—0

   162—On Golden Pond (1981)—01-Jan-1981——http://us.imdb.com/M/title-exact?On%20Golden%20Pond%20(1981)—0—0—0—0—0—0—0—0—1—0—0—0—0—0—0—0—0—0—0

   163—Return of the Pink Panther, The (1974)—01-Jan-1974——http://us.imdb.com/M/title-exact?Return%20of%20the%20Pink%20Panther,%20The%20(1974)—0—0—0—0—0—1—0—0—0—0—0—0—0—0—0—0—0—0—0

3. u.user: Demographic information about the users. This is a tab separated list of:

   user id — age — gender — occupation — zip code

   The user ids are the ones used in the u.data data set.

   Example:

   1—24—M—technician—85711

   2—53—F—other—94043

3—23—M—writer—32067

4—24—M—technician—43537

5—33—F—other—15213

The code for reading from the u.data and u.item files and creating recommendations is described in the book Programming Collective Intelligence. Feel free to modify the PCI code to answer the following questions.

Problem 1

Find 3 users who are closest to you in terms of age, gender, and occupation. For each of those 3 users:

1. what are their top 3 favorite films?
2. bottom 3 least favorite films?

Based on the movie values in those 6 tables (3 users X (favorite + least)), choose a user that you feel is most like you. Feel free to note any outliers (e.g., "I mostly identify with user 123, except I did not like "Ghost" at all").
This user is the "substitute you".

SOLUTION

The solution to the problem is as below:

1. **Loading Movie Lens Dataset**: I read the u.data, u.user and u.item table from the Movie Lens data set and loaded them into separate json files.
2. **Find closest users** : We find the top three users that manage my category. We then find the highest rated and least rated movies for each of the three users.
3. **Most matching user**: The user 774 matches my choice of movies from all the three users. Figure 1 shows the results for similar users anf their least favourite and most favourite movies.

```
import json
import csv


def read_users_table():
    path_data_set = "/home/msiddique/WSDL_Work/WebScience/Assignment6/ml-100k/"
    file_open = open(path_data_set + "u.user", "r")
    list_json = []
    reader = csv.reader(file_open, delimiter="|")
    for row in reader:
        insert_row = {'user': row[0],'age': row[1],'gender': row[2],'occupation':
            row[3],'zip': row[4]}
        list_json.append(insert_row)
    file_json = open("Users.json", "w")
    json.dump(list_json, file_json)
    file_open.close()
    file_json.close()
```

```
def read_items_table():
    path_data_set = "/home/msiddique/WSDL_Work/WebScience/Assignment6/ml-100k/"
    # ratings = pd.read_table(path_dataset + 'u.data',sep='::',names=['user','
        movie','rating','time'])
    file_open = open(path_data_set + "u.item", "r", encoding = "ISO-8859-1")
    list_json = []
    reader = csv.reader(file_open, delimiter="|")
    for row in reader:
        print(row)
        insert_row = {"movie id": row[0], "movie title": row[1], "release date":
            row[2], "video release date": row[3],
                      "IMDb URL": row[4], "unknown": row[5], "Action": row[6], "
                          Adventure": row[7], "Animation":row[8],
                      "Children's": row[9], "Comedy": row[10], "Crime": row[11], "
                          Documentary": row[12],
                      "Drama": row[13], "Fantasy": row[14], "Film-Noir": row[15],
                          "Horror": row[16], "Musical": row[17],
                      "Mystery": row[18], "Romance": row[19], "Sci-Fi": row[20], "
                          Thriller": row[21], "War": row[22],
                      "Western": row[23]}
        list_json.append(insert_row)
    file_json = open("Items.json", "w")
    json.dump(list_json, file_json)
    file_open.close()
    file_json.close()


def read_users_data_table():
    path_data_set = "/home/msiddique/WSDL_Work/WebScience/Assignment6/ml-100k/"
    file_open = open(path_data_set + "u.data", "r")
    list_json = []
    reader = csv.reader(file_open, delimiter="\t")
    for row in reader:
        insert_row = {"user_id": row[0], "item_id": row[1], "rating": row[2], "
            timestamp": row[3]}
        list_json.append(insert_row)
    file_json = open("Data.json", "w")
    json.dump(list_json, file_json)
    file_open.close()
    file_json.close()


# read_users_data_table()
# read_items_table()
read_users_table()
```

```
import json


def find_closest_user(age, occupation):
    file_json = open("Users.json", "r")
    list_json = json.load(file_json)
    list_gender = []
```

```
    for row in list_json:
        if row["gender"] == "M":
            list_gender.append(row)
    list_my_age = []
    for row in list_gender:
        if age == int(row["age"]):
            list_my_age.append(row)
    list_occupation = []
    for row in list_my_age:
        if row["occupation"] == occupation:
            list_occupation.append(row)
    if len(list_occupation) < 3:
        condition_break = False
        for row in list_my_age:
            for inner_row in list_occupation:
                if row["occupation"] != inner_row["occupation"]:
                    list_occupation.append(row)
                    condition_break = True
                    break
            if condition_break:
                break
        if len(list_occupation) < 3:
            condition_break = False
            for row in list_gender:
                for inner_row in list_occupation:
                    if row["user"] != inner_row["user"]:
                        list_occupation.append(row)
                        condition_break = True
                        break
                if condition_break:
                    break
    file_json.close()
    return list_occupation


def find_list_of_movie_ratings(age, occupation):
    list_users = find_closest_user(age, occupation)
    list_user_id = []
    for row in list_users:
        list_user_id.append(row["user"])
    file_json = open("Data.json", "r")
    list_json = json.load(file_json)
    list_favorites = []
    list_least_favourite = []
    for user in list_user_id:
        list_item_id = []
        list_ratings = []
        for row in list_json:
            if row["user_id"] == user:
                list_item_id.append(row["item_id"])
            list_ratings.append(int(row["rating"]))
        list_ratings, list_item_id = zip(*sorted(zip(list_ratings, list_item_id)))
        list_favorites.append(list_item_id[-1])
```

Problem 1 continued on next page...

```
            list_least_favourite.append(list_item_id[0])
        file_json.close()
        file_json = open("Items.json", "r")
        list_json = json.load(file_json)
        list_least_favourite_movies = []
        list_favorites_movies = []
        for item_id in list_favorites:
            for row in list_json:
                if row["movie id"] == item_id:
                    list_favorites_movies.append(row["movie title"])
        for item_id in list_least_favourite:
            for row in list_json:
                if row["movie id"] == item_id:
                    list_least_favourite_movies.append(row["movie title"])

        for i in range(0, len(list_user_id)):
            print("For user id:" + list_user_id[i])
            print("Favorite:" + list_favorites_movies[i])
            print("Favorite movie:" + list_favorites[i])
            print("Least favorite:" + list_least_favourite[i])
            print("Least favorite movie:" + list_least_favourite_movies[i])


find_list_of_movie_ratings(30, 'student')
```

```
/home/msiddique/WSDL_Work/Politowoops_Analysis/venv/bin/python /home/msiddique/WSDL_Work/WebScience/Assignment6/FindClosestUser.py
For user id:774
Favorite:Turbulence (1997)
Favorite movie:986
Least favorite:174
Least favorite movie:Raiders of the Lost Ark (1981)
For user id:869
Favorite:One Fine Day (1996)
Favorite movie:815
Least favorite:118
Least favorite movie:Twister (1996)
For user id:17
Favorite:City of Lost Children, The (1995)
Favorite movie:919
Least favorite:1
Least favorite movie:Toy Story (1995)

Process finished with exit code 0
```

Figure 1: Choice of most similar users to my profile

Problem 2

Which 5 users are most correlated to the substitute you? Which 5 users are least correlated (i.e., negative correlation)?

SOLUTION

For fiding correlated users, I created a json in the format shown of critics as shown in the sample recommendation code provided with the assignment. I used the json object to supply it to the function *sim_pearson* from the sample recommendation code with the sustituted me and the user ids to generate the correlation values. Figure 2 shows the top 5 least and most correlated users.

```
'''
```

```
/home/msiddique/WSDL_Work/Politowoops_Analysis/venv/bin/python /home/msiddique/WSDL_Work/WebScience/Assignment6/CorrelatedUsers.py
Top 5 correlated users
('909', '127', '578', '509', '510')
Last 5 correlated users
('309', '681', '799', '803', '857')
Top 5 recommendation
[(5.0, '814'), (5.0, '1599'), (5.0, '1536'), (5.0, '1467'), (5.0, '1293')]
Last 5 recommendation
[(1.0, '1325'), (1.0, '1321'), (1.0, '1320'), (1.0, '1309'), (1.0, '1308')]
Top 5 correlated movies for my Least favourite movie
('1442', '1558', '1095', '1173', '1249')
Last 5 correlated movies for my least favourite movie
('75', '1465', '113', '119', '1298')
Top 5 correlated movies for my favourite movie
('1508', '883', '1038', '1085', '1102')
Last 5 correlated movies for my favourite movie
('970', '981', '1475', '1015', '1223')

Process finished with exit code 0
```

Figure 2: Top 5 most and least correlated users for sustitute me

```
'''
Create a json file of users to mimic the critic dictionary from Recommendation.py
'''


def create_critics_json():
    file_json = open("Data.json", "r")
    list_json = json.load(file_json)
    critic ={}
    for row in list_json:
        if row["user_id"] in critic.keys():
            critic[row["user_id"]].update({row["item_id"]: int(row["rating"])})
        else:
            critic[row["user_id"]] = {row["item_id"]: int(row["rating"])}
    file_json.close()
    file_correlated_users = open("Critic.json", "w")
    json.dump(critic, file_correlated_users)
    file_correlated_users.close()


'''
Use Recommendation.py function to calculate correlation and get top 5 and last five
'''


def find_correlated_users(user_id):
    create_critics_json()
    file_json = open("Critic.json", "r")
    list_json = json.load(file_json)
    list_correlation = []
    list_user = []
    for user in list_json:
        list_user.append(user)
        list_correlation.append(sim_pearson(list_json, user_id, user))
    file_json.close()
    list_correlation, list_user = zip(*sorted(zip(list_correlation, list_user)))
    print("Top 5 correlated users")
    print(list_user[:5])
    print("Last 5 correlated users")
    print(list_user[-5:])
    get_recommended_list(user_id, list_json)
```

```
    choose_real_you(list_json)
```

Problem 3

Compute ratings for all the films that the substitute you have not seen. Provide a list of the top 5 recommendations for films that the substitute you should see. Provide a list of the bottom 5 recommendations (i.e., films the substitute you is almost certain to hate).

SOLUTION

I used the *get_recommendation* function from the recommendation sample code with the json data created from the previous problem of finding correlation users with substitute me user id to get the top 5 most and meast favourite movies. Figure 3 shows the resultsfor recommendation results.

```
/home/msiddique/WSDL_Work/Politowoops_Analysis/venv/bin/python /home/msiddique/WSDL_Work/WebScience/Assignment6/CorrelatedUsers.py
Top 5 correlated users
('909', '127', '578', '509', '510')
Last 5 correlated users
('309', '681', '799', '803', '857')
Top 5 recommendation
[(5.0, '814'), (5.0, '1599'), (5.0, '1536'), (5.0, '1467'), (5.0, '1293')]
Last 5 recommendation
[(1.0, '1325'), (1.0, '1321'), (1.0, '1320'), (1.0, '1309'), (1.0, '1308')]
Top 5 correlated movies for my Least favourite movie
('1442', '1558', '1095', '1173', '1249')
Last 5 correlated movies for my least favourite movie
('75', '1465', '113', '119', '1298')
Top 5 correlated movies for my favourite movie
('1508', '883', '1038', '1085', '1102')
Last 5 correlated movies for my favourite movie
('970', '981', '1475', '1015', '1223')

Process finished with exit code 0
```

Figure 3: Top 5 favorite and least favorite recommendation for sustitute me user

```
'''
Function to get recommendation list for user
'''


def get_recommended_list(user_id, list_json):
    list_recommendation = getRecommendations(list_json, user_id)
    print("Top 5 recommendation")
    print(list_recommendation[:5])
    print("Last 5 recommendation")
    print(list_recommendation[-5:])
```

Problem 4

Choose your (the real you, not the substitute you) favorite and least favorite film from the data. For each film, generate a list of the top 5 most correlated and bottom 5 least correlated films. Based on your knowledge of the resulting films, do you agree with the results? In other words, do you personally like / dislike the resulting films?

SOLUTION

I added my favourite movie as movie id 161 (Top Gun) and least favorite movie as 162 (On Golden Pond).

I added these enties into the json that I created in the correlation users case and transformed the json to be now listed by movie id rather than user id by using the function *transformPrefs* from the sample recommendation code and ran the *sim_pearson* function to get my list of most and least correlated top 5 movies for my favourite and least favourite movies. Figure 4 shows the movie correlation results.
The results for the correlation values did not make a lot of sense to me, as I had not watched many of the movies listed in the dataset. But looking at their genres and synopsis from IMDB, I realised the correlation results were very similar to my choices.

```
/home/msiddique/WSDL_Work/Politowoops_Analysis/venv/bin/python /home/msiddique/WSDL_Work/WebScience/Assignment6/CorrelatedUsers.py
Top 5 correlated users
('909', '127', '578', '509', '510')
Last 5 correlated users
('309', '681', '799', '803', '857')
Top 5 recommendation
[(5.0, '814'), (5.0, '1599'), (5.0, '1536'), (5.0, '1467'), (5.0, '1293')]
Last 5 recommendation
[(1.0, '1325'), (1.0, '1321'), (1.0, '1320'), (1.0, '1309'), (1.0, '1308')]
Top 5 correlated movies for my Least favourite movie
('1442', '1558', '1095', '1173', '1249')
Last 5 correlated movies for my least favourite movie
('75', '1465', '113', '119', '1298')
Top 5 correlated movies for my favourite movie
('1508', '883', '1038', '1085', '1102')
Last 5 correlated movies for my favourite movie
('970', '981', '1475', '1015', '1223')

Process finished with exit code 0
```

Figure 4: Movie Correlation for top 5 most and least correlated movies

```
def choose_real_you(list_json):
    list_json.update({"Me":{"161": 5, "162": 1}})
    list_transformed = transformPrefs(list_json)
    list_correlation_161 = []
    list_correlation_162 = []
    list_movies_161 = []
    list_movies_162 = []
    for movies in list_transformed:
        list_movies_161.append(movies)
        list_movies_162.append(movies)
        list_correlation_161.append(sim_pearson(list_transformed, "161", movies))
        list_correlation_162.append(sim_pearson(list_transformed, "162", movies))
    list_correlation_161, list_movies_161 = zip(*sorted(zip(list_correlation_161,
        list_movies_161)))
    list_correlation_162, list_movies_162 = zip(*sorted(zip(list_correlation_162,
        list_movies_162)))
    print("Top 5 correlated movies for my Least favourite movie")
    print(list_movies_162[:5])
    print("Last 5 correlated movies for my least favourite movie")
    print(list_movies_162[-5:])
    print("Top 5 correlated movies for my favourite movie")
    print(list_movies_161[:5])
    print("Last 5 correlated movies for my favourite movie")
    print(list_movies_161[-5:])
```