Web Science Assignment 6 Report

Dr. Nelson

James Pindell

3/20/2018

Content

General Info

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|0|0|0|0|0|0|0|0|0|0|1|0|0|0|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

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:

- what are their top 3 favorite films?

- 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 for this problem is outlined by the following steps:

1. **Modify Original Code:** In order to find 3 users who were similar to me in terms of age, gender, and occupation, the code for finding the distance-based similarity score, between myself and all the other users, had to be modified. Unfortunately, the code for that function is too long to show as a snippet, but you can find the code for that function, called userSimilarity, from within the file, called MovieLens.py, which is within the package.

   The 3 users whom I chose to be most similar to me have the User ID's of 245, 327, and 359.

2. **Identify Users' 3 Favorite and 3 Least Favorite Movies**: Once 3 users have been identified, The u.data file was manually searched through in order to find the 3 favorite and 3 least favorite movies for each user. You can find the u.data file in the ml-100k file folder, which is located within the package.

3. **Create Tables for Favorite and Least Favorite Movies**: After manually searching through the u.data file, tables that show the 3 user's favorite and least favorite movies were manually created. Unfortunately, the tables are too large to show them as snippets, but you can find these tables in the file, called the Problem 1 Similarity Tables.txt, from within the package.

4. **Substitute Me**: Once the tables were created, I chose User 245 to be the substitute me, however, I have not seen the movies “Grumpier Old Men (1995)” and “Homeward Bound II: Lost in San Francisco (1996)”.

Problem 2

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

Solution

The solution for this problem is outlined by the following steps:

1. **Use Provided Code to Find Pearson Correlation Coefficients**: In order to identify which 5 users were most correlated and which 5 users were least correlated, the original code was used to compute the Pearson Correlation Coefficients, that compares User 245 (substitute me) to all other users. Unfortunately, the code for that function is too long to show as a snippet, but you can find the code for that function, called userCorrelation, from within the file, called MovieLens.py, which is within the package.

2. **Create Tables for 5 Most Correlated and 5 Least Correlated Movies**: Once all of the Pearson Correlation Coefficients had been computed, tables that show the 5 most correlated users and 5 least correlated users were manually created. You can find these tables in the file, called the Problem 2 User Correlation Tables.txt, from within the package, but a snippet of the table is shown below:

```
Users with Positive Correlation to User 245
-------------------------------------------

User | Correlation
-------------------------
766  | 1.000000000000004
97   | 1.000000000000001
719  | 1.0000000000000004
93   | 1.0
71   | 1.0

Users with Negative Correlation to User 245
-------------------------------------------

User | Correlation
-------------------------
214  | -1.000000000000004
765  | -1.0000000000000007
657  | -1.0000000000000007
608  | -1.0000000000000004
567  | -1.0000000000000004
```

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

The solution for this problem is outlined by the following steps:

1. **Use Provided Code to Compute Ratings For Movies That Have Not Been Seen:** In order to compute ratings for all the films that the substitute me have not seen, the original code was used to compute said ratings, ordering them from the movie with the highest rating to the movie with the lowest rating. You can view the full code for this function, called getNotSeenRatings, from within the file, called MovieLens.py, from within the package, but a snippet of the code is shown below:

```
totals = {}
simSums = {}
for other in prefs:
# Don't compare me to myself
    if other == person:
        continue
    sim = similarity(prefs, person, other)
    # Ignore scores of zero or lower
    if sim <= 0:
        continue
    for item in prefs[other]:
        # Only score movies I haven't seen yet
        if item not in prefs[person] or prefs[person][item] == 0:
            # Similarity * Score
            totals.setdefault(item, 0)
            # The final score is calculated by multiplying each item by the
            #   similarity and adding these products together
            totals[item] += prefs[other][item] * sim
            # Sum of similarities
            simSums.setdefault(item, 0)
            simSums[item] += sim
# Create the normalized list
rankings = [(total / simSums[item], item) for (item, total) in totals.items()]
# Return the sorted list
rankings.sort()
rankings.reverse()
return rankings
```

2. **Create Tables for 5 Highest and 5 Lowest Recommended Movies**: Once all of recommended ratings have been computed, tables that show the 5 most recommended movies and the 5 least recommended movies were manually created. You can find these tables in the file, called the Problem 3 Rating Tables.txt, from within the package, but a snippet of the table is shown below:

```
User 245's Top 5 Recommended Movies
-----------------------------------

Movie Title                                   | Rating
--------------------------------------------------------------
Drunks (1995)                                 | 5.000000000000001
Whole Wide World, The (1996)                  | 5.0
Wedding Gift, The (1994)                      | 5.0
Two or Three Things I Know About Her (1966)   | 5.0
Thieves (Voleurs, Les) (1996)                 | 5.0

User 245's Bottom 5 Recommended Movies
--------------------------------------

Movie Title                                   | Rating
--------------------------------------------------------------
1-900 (1994)                                  | 1.0
3 Ninjas: High Noon At Mega Mountain (1998)   | 1.0
Amityville 1992: It's About Time (1992)       | 1.0
Amityville: A New Generation (1993)           | 1.0
Amityville: Dollhouse (1996)                  | 1.0
```

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

The solution for this problem is outlined by the following steps:

1. **Choose Favorite and Least Favorite Films**: My favorite film that I chose from the data is “Toy Story (1995)” and my least favorite film that I chose from the data (although I have never seen this film, but I know I hate horror movies) is “Nightmare on Elm Street, A (1984)” .

2. **Modify Original Code**: In order to find the 5 most correlated movies and the 5 least correlated movies for both my chosen favorite movie and my chosen least favorite movie, the code for finding the 5 most correlated movies and the 5 least correlated movies had to be modified. You can find the code for that function, called movieCorrelation, from within the file, called MovieLens.py, which is within the package, but a snippet of the code is shown below:

```
Returns the Pearson Correlation Coefficient for movies m1 and m2.
'''

# Get the list of mutually rated items
si = {}
for item in prefs[m1]:
    if item in prefs[m2]:
        si[item] = 1
# If they are no ratings in common, return 0
if len(si) == 0:
    return 0
# Sum calculations
n = len(si)
# Sums of all the preferences
sum1 = sum([prefs[m1][it] for it in si])
sum2 = sum([prefs[m2][it] for it in si])
# Sums of the squares
sum1Sq = sum([pow(prefs[m1][it], 2) for it in si])
sum2Sq = sum([pow(prefs[m2][it], 2) for it in si])
# Sum of the products
pSum = sum([prefs[m1][it] * prefs[m2][it] for it in si])
# Calculate r (Pearson score)
num = pSum - sum1 * sum2 / n
den = sqrt((sum1Sq - pow(sum1, 2) / n) * (sum2Sq - pow(sum2, 2) / n))
if den == 0:
    return 0
r = num / den
return r
```

3. **Create Tables for 5 Most Correlated and 5 Least Correlated Movies**: Once all of the Pearson Correlation Coefficients had been computed, tables that show the 5 most correlated movies and 5 least correlated movies were manually created. Unfortunately, the tables are too long to show as a snippet, but you can find these tables in the file, called the Problem 4 Movie Correlation Tables.txt, from within the package.

4. **Like / Dislike Resulting Films**: It is tough for me to say whether or not I agree with the results, because I have never seen any of the movies listed in any of the tables. However, if I must choose between liking and disliking the results, I will claim that I agree with them (assuming the fact that the calculation for the movie correlation is indeed correct).