Assignment 7 — **Hussam Hallak**

CS532, Web Science, Spring 2017 — CS Master's Student

Old Dominion University, Computer Science Dept — Prof: Dr. Nelson

Question 1:

The goal of this project is to use the basic recommendation principles we have learned for user-collected data. You will modify the code 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?Top162—On Golden Pond (1981)—01-Jan-1981——http://us.imdb.com/M/title-exact?On163—Return of the Pink Panther, The (1974)—01-Jan-1974——http://us.imdb.com/M/title-exact?Return

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.

Questions (10 points).

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".

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

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).

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?

Answer:

1. In order to find 3 users who are closest to me in terms of age, 36 years old, and gender, male, and occupation, programmer. I wrote a small python script and named it “findclosestusers.py”. I kept refining the results until I got the 3 closest users. The script takes a text file as input from the command line and saves the results to a text file named “closest3users.txt”.

Listing 1: The content of findclosestusers.py

```
import sys

if len(sys.argv) < 2:
   print "Usage: python findclosestusers.py <input_file>"
   print "e.g: python findclosestusers.py u.user"
   exit()

users = []
fh_output = open('closest3users.txt', 'w')

with open(sys.argv[1]) as fp_input:
   for line in fp_input:
        users.append(line.split('|'))

for user in users:
   if (user[1] < '37' and user[1] > '34' and user[2] == 'M' and user[3] == '
      programmer'):
      fh_output.write(str(user[0])+'|')
      fh_output.write(str(user[1])+'|')
      fh_output.write(str(user[2])+'|')
      fh_output.write(str(user[3])+'|')
      fh_output.write(str(user[4]))

fp_input.close()
fh_output.close()
```

After running the script “findclosestusers.py” and passing the file “u.user”, I got the following users:

Listing 2: The content of closest3users.txt

```
182|36|M|programmer|33884
215|35|M|programmer|63033
633|35|M|programmer|55414
```

In order to find the top 3 and the bottom 3 favorite films for each of the 3 users, I wrote a python script, "findtopbottom343.py". The program takes 3 command line arguments:

a. The file "closest3users.txt": Gives users' ids.

b. The file "u.data": Gives items' ids and ratings that belong to the 3 users.

c. The file "u.item" Gives the data for the 6 films' we need.

The output is saved to a file named "topbottom343.txt"

Listing 3: The content of findtopbottom343.py

```
import sys
from operator import itemgetter

if len(sys.argv) < 4:
       print "Usage: python findtopbottom343.py <closest_3_users> <ratings_file>
           <films_file>"
       print "e.g: python findtopbottom343.py closest3users.txt u.data u.item"
       exit()

users = []
ratings = []
films = []
fh_output = open('topbottom343.txt', 'w')

with open(sys.argv[1]) as users_input:
       for line in users_input:
              users.append(line.split('|'))

with open(sys.argv[2]) as ratings_input:
       for line in ratings_input:
              ratings.append(line.split('\t'))

with open(sys.argv[3]) as films_input:
       for line in films_input:
              films.append(line.split('|'))
first_user_ratings = []
second_user_ratings = []
third_user_ratings = []

i = 0
for user in users:
   i = i+1
   for rating in ratings:
      if (user[0] == rating[0]):
        if (i==1):
           first_user_ratings.append(rating)
        if (i==2):
           second_user_ratings.append(rating)
        if (i==3):
           third_user_ratings.append(rating)
```

```
first_user_ratings = sorted(first_user_ratings , key=itemgetter(2))
second_user_ratings = sorted(second_user_ratings , key=itemgetter(2))
third_user_ratings = sorted(third_user_ratings , key=itemgetter(2))

######## First user top and bottom films #########

first_user_bottom = []
first_user_top = []

for i in range(0,3):
   first_user_bottom.append(first_user_ratings[i])

for i in range(len(first_user_ratings)-3, len(first_user_ratings)):
   first_user_top.append(first_user_ratings[i])

film_first_bottom = []
for rat in first_user_bottom:
   film_first_bottom.append(rat[1])

film_first_top = []
for rat in first_user_top:
        film_first_top.append(rat[1])

fh_output.write('First User id: ' + first_user_ratings[0][0])
fh_output.write('\n******************\n')

fh_output.write('Top 3 films:')
fh_output.write('\n--------------\n')

for film in film_first_top:
   for movie in films:
      if (film == movie[0]):
        fh_output.write(movie[1] + '\n')

fh_output.write('--------------------------------------------\n')
fh_output.write('Bottom 3 films:')
fh_output.write('\n-----------------\n')

for film in film_first_bottom:
        for movie in films:
                if (film == movie[0]):
                        fh_output.write(movie[1] + '\n')

fh_output.write('*****************************************************\n')

######## Second user top and bottom films #########

second_user_bottom = []
second_user_top = []

for i in range(0,3):
        second_user_bottom.append(second_user_ratings[i])

for i in range(len(second_user_ratings)-3, len(second_user_ratings)):
        second_user_top.append(second_user_ratings[i])
```

```
film_second_bottom = []
for rat in second_user_bottom:
        film_second_bottom.append(rat[1])

film_second_top = []
for rat in second_user_top:
        film_second_top.append(rat[1])

fh_output.write('Second User id: ' + second_user_ratings[0][0])
fh_output.write('\n******************\n')

fh_output.write('Top 3 films:')
fh_output.write('\n--------------\n')

for film in film_second_top:
        for movie in films:
                if (film == movie[0]):
                        fh_output.write(movie[1] + '\n')

fh_output.write('--------------------------------------------\n')
fh_output.write('Bottom 3 films:')
fh_output.write('\n-----------------\n')

for film in film_second_bottom:
        for movie in films:
                if (film == movie[0]):
                        fh_output.write(movie[1] + '\n')

fh_output.write('*****************************************************\n')


######## Third user top and bottom films #########

third_user_bottom = []
third_user_top = []

for i in range(0,3):
        third_user_bottom.append(third_user_ratings[i])

for i in range(len(third_user_ratings)-3, len(third_user_ratings)):
        third_user_top.append(third_user_ratings[i])

film_third_bottom = []
for rat in third_user_bottom:
        film_third_bottom.append(rat[1])

film_third_top = []
for rat in third_user_top:
        film_third_top.append(rat[1])

fh_output.write('Third User id: ' + third_user_ratings[0][0])
fh_output.write('\n******************\n')

fh_output.write('Top 3 films:')
fh_output.write('\n--------------\n')
```

```
for film in film_third_top:
        for movie in films:
                if (film == movie[0]):
                        fh_output.write(movie[1] + '\n')

fh_output.write('--------------------------------------------\n')
fh_output.write('Bottom 3 films:')
fh_output.write('\n------------------\n')

for film in film_third_bottom:
        for movie in films:
                if (film == movie[0]):
                        fh_output.write(movie[1] + '\n')

fh_output.write('*****************************************************\n')
```

After running the script, I got the following output:

Listing 4: Running findtopbottom343.py

```
root@ima-app:/var/www/Hussam/A7# python findtopbottom343.py closest3users.txt u.
    data u.item
root@ima-app:/var/www/Hussam/A7# cat topbottom343.txt
First User id: 182
******************
Top 3 films:
--------------
Star Wars (1977)
Hunchback of Notre Dame, The (1996)
12 Angry Men (1957)
--------------------------------------------
Bottom 3 films:
-----------------
Emma (1996)
That Thing You Do! (1996)
Donnie Brasco (1997)
*****************************************************
Second User id: 215
******************
Top 3 films:
--------------
Silence of the Lambs, The (1991)
Wizard of Oz, The (1939)
Searching for Bobby Fischer (1993)
--------------------------------------------
Bottom 3 films:
-----------------
Unbearable Lightness of Being, The (1988)
Seven (Se7en) (1995)
Star Trek V: The Final Frontier (1989)
*****************************************************
Third User id: 633
******************
Top 3 films:
--------------
Die Hard 2 (1990)
```

```
Basic Instinct (1992)
Fugitive, The (1993)
--------------------------------------------
Bottom 3 films:
-----------------
Ghost and the Darkness, The (1996)
Feeling Minnesota (1996)
Scream (1996)
*****************************************************
root@ima-app:/var/www/Hussam/A7#
```

I mostly identify with user 182. Moving forward I will call 182 “my substitute”.

2. In order to find which 5 users are most correlated to the my substitute, 182, I utilized the code in the file “recommendations.py” from PCI book, Programming Collective Intelligence. I should have used the code for part one also, but I was trying to do everything from scratch. I modified the function “topMatches” and used it to find the 5 users that are most and least correlated to the my substitute, 182. I simply added an argument “m” to the function. if m = 1 or True, the function is not changed at all. If m = 0 or False is passed to the function, the function does not reverse the list after sorting which gives the least 5 correlated users. Below is the function “topMatches” after it has been modified.

Listing 5: topMatches

```python
def topMatches(m, prefs, person, n=5, similarity=sim_pearson):
    scores = [(similarity(prefs, person, other), other) for other in prefs
              if other != person]
    scores.sort()
    if m:
        scores.reverse()
    return scores[0:n]
```

I wrote a python script “findcorrelated.py” that takes a command line argument, user_id, as input and prints out the 5 users that are most and least correlated to the user_id passed to the script.

Listing 6: The content of findcorrelated.py

```python
import sys
import recommendations as rec

if len(sys.argv) < 2:
   print "Usage: python findcorrelated.py <user_id>"
   print "e.g: python findcorrelated.py 182"
   exit()

pref= rec.loadMovieLens()
correlated_users = rec.topMatches(1, pref, sys.argv[1])
noncorrelated_users = rec.topMatches(0, pref, sys.argv[1])
print "Five Most Correlated Users to My Substitute: " + sys.argv[1]
print "----------------------------------------------------------"
for user in correlated_users:
   print user[1]

print '++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++'
print "Five Least Correlated Users to My Substitute: " + sys.argv[1]
print "---------------------------------------------------------"
for user in noncorrelated_users:
   print user[1]
```

After running the script “findcorrelated.py” and passing my substitute’s id 182, I got the following users:

Listing 7: Running findcorrelated.py

```
root@ima-app:/var/www/Hussam/A7# python findcorrelated.py 182
Five Most Correlated Users to My Substitute: 182
---------------------------------------------------------
283
156
777
603
826
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Five Least Correlated Users to My Substitute: 182
---------------------------------------------------------
127
132
492
617
641
root@ima-app:/var/www/Hussam/A7#
```

3. In order to compute ratings for all the films that the my substitute has not seen and provide a list of the top/bottom 5 recommendations for films that my substitute you should/shouldn’t see. I used the function “getRecommendations” in “recommendations.py” to get top and bottom recommendations based on collaborative filtering.

I wrote a python script “findrecommended55.py” that takes user_id as a command line argument, in this case my substitute user 182, and prints out the top and bottom 5 recommended films.

Listing 8: The content of findrecommended55.py

```
import sys
import recommendations as rec

if len(sys.argv) < 2:
       print "Usage: python findrecommended.py <user_id>"
       print "e.g: python findrecommended.py 182"
       exit()

pref= rec.loadMovieLens()
ranks = rec.getRecommendations(pref, sys.argv[1])

print "Five Most recommended Movies for My Substitute: " + sys.argv[1]
print "---------------------------------------------------------"
for movie in ranks[:5]:
       print movie[1]

print '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++'
print "Five Least Recommended Movies for My Substitute: " + sys.argv[1]
print "----------------------------------------------------------"
for movie in ranks[-5:]:
       print movie[1]

print '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++'
```

After running the script "findrecommended55.py" and passing my substitute's id 182, I got the following output which has both lists of top and bottom 5 recommendations for my substitute, user 182:

Listing 9: Running findrecommended55.py

```
root@ima-app:/var/www/Hussam/A7# python findrecommended55.py 182
Five Most recommended Movies for My Substitute: 182
---------------------------------------------------------
They Made Me a Criminal (1939)
The Deadly Cure (1996)
Someone Else's America (1995)
Saint of Fort Washington, The (1993)
Prefontaine (1997)
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Five Least Recommended Movies for My Substitute: 182
---------------------------------------------------------
Amityville II: The Possession (1982)
Amityville Curse, The (1990)
Amityville 3-D (1983)
Amityville 1992: It's About Time (1992)
3 Ninjas: High Noon At Mega Mountain (1998)
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
root@ima-app:/var/www/Hussam/A7#
```

4. It was easy to find my favorite movie from the data, but so hard to find the least favorite movie because if I decide to watch a movie, I give it 5 minutes to get my attention. If it did not, I look for something else and forget the name, actors, ...etc and completely remove it from my memory. Anyways, I looked up one of the movies and read its story and it seems so boring, so I decided to choose it as my least favorite movie without seeing it.

In order to generate a list of the top 5 most correlated and bottom 5 least correlated films to the two films, I have chosen.

My absolute favorite film of all times is: Braveheart (1995)

My least favorite film is: Amityville II: The Possession (1982)

I wrote a python script, findmyrecommendations.py. It imports "recommendations.py" and uses the function "calculateSimilarItems" with a slight modification to compute a list of recommended/not-recommended movies for me to watch/not-watch. The modification is done by passing an extra argument in the function, m. This argument is, then, passed to the function "topMatches" called from "calculateSimilarItems". If m = 1 or True, the function returns a list of recommended movies to watch. Similarly, if m = 0 or False, the function returns a list of non-recommended movies to not watch. Below is the function "calculateSimilarItems" after it has been modified.

Listing 10: topMatches

```
def calculateSimilarItems(m, prefs, n=10):
    result = {}
    # Invert the preference matrix to be item-centric
    itemPrefs = transformPrefs(prefs)
    c = 0
    for item in itemPrefs:
        # Status updates for large datasets
        c += 1
        if c % 100 == 0:
            print '%d / %d' % (c, len(itemPrefs))
        # Find the most similar items to this one
        scores = topMatches(m, itemPrefs, item, n=n, similarity=sim_distance)
        result[item] = scores
    return result
```

Below is the code in “findmyrecommendations.py”.

Listing 11: The content of findmyrecommendations.py

```
import sys
import recommendations as rec

pref= rec.loadMovieLens()

top_movie = 'Braveheart (1995)'
bottom_movie = 'Amityville II: The Possession (1982)'

top_movies = rec.calculateSimilarItems(1, pref, 5)
bottom_movies = rec.calculateSimilarItems(0, pref, 5)

print 'Best Recommended movies to watch based on ' + top_movie + ':'
print '----------------------------------------------------------------'
for movie in top_movies[top_movie]:
    print movie[1]
print '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++'
print 'Least Recommended movies to watch based on ' + top_movie + ':'
print '----------------------------------------------------------------'
for movie in bottom_movies[top_movie]:
    print movie[1]
print '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++'
print 'Best Recommended movies to watch based on ' + bottom_movie + ':'
print '----------------------------------------------------------------'
for movie in top_movies[bottom_movie]:
    print movie[1]
print '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++'
print 'Least Recommended movies to watch based on ' + bottom_movie + ':'
print '----------------------------------------------------------------'
for movie in bottom_movies[bottom_movie]:
    print movie[1]
print '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++'
```

After running the script “findmyrecommendations.py”, I got the list of top and bottom 5 recommendations for myself:

Listing 12: Running findmyrecommendations.py

```
root@ima-app:/var/www/Hussam/A7# python findmyrecommendations.py
```

```
100 / 1664
200 / 1664
300 / 1664
400 / 1664
500 / 1664
600 / 1664
700 / 1664
800 / 1664
900 / 1664
1000 / 1664
1100 / 1664
1200 / 1664
1300 / 1664
1400 / 1664
1500 / 1664
1600 / 1664
100 / 1664
200 / 1664
300 / 1664
400 / 1664
500 / 1664
600 / 1664
700 / 1664
800 / 1664
900 / 1664
1000 / 1664
1100 / 1664
1200 / 1664
1300 / 1664
1400 / 1664
1500 / 1664
1600 / 1664
Best Recommended movies to watch based on Braveheart (1995):
-----------------------------------------------------------------
Wings of Courage (1995)
Wife, The (1995)
The Deadly Cure (1996)
Sexual Life of the Belgians, The (1994)
Of Love and Shadows (1994)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Least Recommended movies to watch based on Braveheart (1995):
-----------------------------------------------------------------
Aiqing wansui (1994)
American Strays (1996)
August (1996)
B. Monkey (1998)
Ballad of Narayama, The (Narayama Bushiko) (1958)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Best Recommended movies to watch based on Amityville II: The Possession (1982):
-----------------------------------------------------------------
Yankee Zulu (1994)
Women, The (1939)
Woman in Question, The (1950)
Withnail and I (1987)
Winter Guest, The (1997)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Least Recommended movies to watch based on Amityville II: The Possession (1982):
---------------------------------------------------------------
'Til There Was You (1997)
8 Heads in a Duffel Bag (1997)
A Chef in Love (1996)
Above the Rim (1994)
Adventures of Pinocchio, The (1996)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
root@ima-app:/var/www/Hussam/A7#
```

I honestly watch less than 5 movies a year. I have not watched the majority of these movies, but after reading about them, I mostly agree with both lists of least recommended movies, and mostly disagree with movies in the list of recommended movies.

Included Files:

closest3users.txt, findclosestusers.py, findcorrelated.py, findmyrecommendations.py, findrecommended55.py, findtopbottom343.py, recommendations.py, recommendations.pyc, topbottom343.txt, u.data, u.item, u.user

Folder: data has u.data, u.item, u.user, and a sub-folder: movielens that has u.data, u.item, u.user.