CS532 Web Science: Assignment 7

Finished on March 31, 2016

Dr. Michael L. Nelson

Naina Sai Tipparti
ntippart@cs.odu.edu

Contents

Listings

List of Figures

List of Tables

Problem 1

Question

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/

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.

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 ll").

This user is the "substitute you".

1.1 Answer

Each question was answered by using some combination of the existing functions from `recommendations.py` and some functions that were added. All of the tables provided were created using the `tabulate` function (with some minor edits), which is shown in Listing 1.

Question 1 was solved using the code in Listing 2, which utilizes the added `flatten` and `get_top` functions, which are found in Listing 3 and 4.

```
def tabulate(tuples, caption, label, colnames, output):
    output.write('\\begin{table}[h!]\n')
    output.write('\\centering\n')
    opts = '| ' + ' | '.join(['l' for i in xrange(len(tuples[0]))]) + ' |'
    output.write('\\begin{{tabular}}{{{0}}}\n'.format(opts))
    output.write('\\hline\n')
    header = ' & '.join(['{}' for i in xrange(len(tuples[0]))]).format(*colnames)
    output.write(header + ' \\\\\n\\hline\n')
    for item in tuples:
        temp = ' & '.join(['{}' for i in xrange(len(item))])
        output.write(temp.format(*item) + ' \\\\\n')
    output.write('\\hline\n\\end{tabular}\n')
    output.write('\\caption{{{0}}}\n'.format(caption))
    output.write('\\label{{tab:{0}}}\n'.format(label))
    output.write('\\end{table}\n\n')
```

Listing 1: tabulate function

```
        getuser = {}
        user_filter = lambda x: x['gender'] == 'M' and x['job'] == 'student' and int(x['age'
            ]) == 23
        ratings = []
        for mid, movie in movies.iteritems():
            for user, user_ratings in prefs.iteritems():
                if user_filter(users[user]) and user_ratings.has_key(movies[mid]):
                    getuser.setdefault(int(user),{})
                    getuser[int(user)][movies[mid]]=float(user_ratings[movies[mid]])
                    ratings.append(user_ratings[movies[mid]])
        sorted_getuser = {}
        movie_sort = {}
        for user, user_movie in getuser.items():
            user_movie_sort = sorted(user_movie.items(), key=itemgetter(1), reverse=False)
            for title, rating in user_movie_sort[:3]:
                movie_sort.setdefault(title,rating)
            # print movie_sort
            sorted_getuser.setdefault(user, user_movie_sort[:3])
        sorted_avg_all = sorted(sorted_getuser.items(), key=itemgetter(0), reverse=False)
        # print sorted_avg_all
        top_raters = get_top(sorted_avg_all, key=lambda x,i: x[i][1][0])
        top_raters = [flatten(rater) for rater in sorted_avg_all]
        tabulate(top_raters, 'Users', 'user', ('User', 'Rating', 'Movie'), outfile)
        print "done with 1"
```

Listing 2: Question 1 code

```
def get_top(sorted_list, key=lambda x, i: x[i][1], n=5):
    top = key(sorted_list, 0)
    top_items = []
    i = 0
    while i < n or key(sorted_list, i) == top:
        top_items.append(sorted_list[i])
        if i < n and key(sorted_list, i) != top:
            top = key(sorted_list, i)
        i += 1
    return top_items
```

Listing 3: get_top functions

```
def flatten(tup, f=lambda x: (x[0], x[1][0][1], x[1][0][0])):
    return f(tup)
```

Listing 4: flatten function

1.1.1 3-Users Closest to Me in Terms of Age, Gender, and Occupation

Id	Age	Gender	Occupation
33	23	M	student
37	23	M	student
66	23	M	student
135	23	M	student
391	23	M	student
408	23	M	student
706	23	M	student
838	23	M	student

Table 1: All-Users Closest Match

1.1.2 3-Users Closest to Me Top 3 Favorite Films

User	Rating	Movie
33	5.0	Titanic (1997)
	4.0	Game, The (1997)
	4.0	Air Force One (1997)
37	5.0	Pulp Fiction (1994)
	5.0	Raiders of the Lost Ark (1981)
	5.0	Terminator, The (1984)
66	5.0	Return of the Jedi (1983)
	5.0	Air Force One (1997)
	5.0	Ransom (1996)
135	5.0	Silence of the Lambs, The (1991)
	4.0	Rear Window (1954)
	4.0	Liar Liar (1997)
391	5.0	Rear Window (1954)
	5.0	Magnificent Seven, The (1954)
	5.0	Blues Brothers, The (1980)
408	5.0	Liar Liar (1997)
	5.0	Lost Highway (1997)
	5.0	Everyone Says I Love You (1996)
706	5.0	Phenomenon (1996)
	5.0	Edge, The (1997)
	5.0	Star Wars (1977)
838	5.0	Bringing Up Baby (1938)
	5.0	Toy Story (1995)
	5.0	City of Lost Children, The (1995)

Table 2: Top 3 Favorite Films

1.1.3 3-Users Closest to Me Least 3 Favorite Films

User	Rating	Movie
33	3.0	Liar Liar (1997)
	3.0	Devil's Advocate, The (1997)
	3.0	Soul Food (1997)
37	1.0	Jurassic Park (1993)
	2.0	Twister (1996)
	2.0	Arrival, The (1996)
66	1.0	English Patient, The (1996)
	1.0	Muppet Treasure Island (1996)
	1.0	Excess Baggage (1997)
135	1.0	Tales from the Hood (1995)
	2.0	Jaws 2 (1978)
	2.0	Star Trek III: The Search for Spock (1984)
391	1.0	Mimic (1997)
	2.0	Star Trek: The Wrath of Khan (1982)
	2.0	Courage Under Fire (1996)
408	1.0	Mouse Hunt (1997)
	2.0	U Turn (1997)
	2.0	Conspiracy Theory (1997)
706	1.0	Game, The (1997)
	1.0	Fargo (1996)
	1.0	Crash (1996)
838	2.0	Mars Attacks! (1996)
	2.0	Independence Day (ID4) (1996)
	2.0	Air Force One (1997)

Table 3: Least 3 Favorite Films

1.1.4 Substitute Me

Selecting substitute me was a difficult choice. I couldn't relate to user 33, 37, 66, 391, 706, 838, despite the fact that I would rank "Titanic (1997)" similarly with a 5.

I am greatly removed from user 33, 37, 66, 391, 706, 838 all the most maximum ranked movies by 33, 37, 66, 391, 706, 838 would be in my minimum ranked movies determination.

I am not that nearby either with user 135, 408, the most elevated ranked movies will be in my 4 rate selection. So, User 135 was selected as substitute of me.

User	Rating	Movie
135	5.0	Silence of the Lambs, The (1991)
	4.0	Rear Window (1954)
	4.0	Liar Liar (1997)
	1.0	Tales from the Hood (1995)
	2.0	Jaws 2 (1978)
	2.0	Star Trek III: The Search for Spock (1984)

Table 4: User 135 was selected as substitute me

References