CS 432/532 Web Science: Assignment #6

Alexander C. Nwala

Bethany DeMerchant

31 March 2019

Contents

Problem 1:

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".

Problem 1– Methods:

There are exactly three 23-year-old female students: users 49, 159, and 477. Their favorite and least favorite movies are as follows[1]:

49 (214 total rankings)	
Monty Python and the Holy Grail (1974)	5
Willy Wonka and the Chocolate Factory (1971)	5
Nightmare Before Christmas, The (1993)	5
Oliver & Company (1988)	1
Top Gun (1986)	1
Independence Day (ID4) (1996)	1

159 (106 total rankings)	
Godfather, The (1972)	5
Grease (1978)	5
Mr. Holland's Opus (1995)	5
English Patient, The (1996)	1
All Dogs Go to Heaven 2 (1996)	1
Love Jones (1997)	1

477 (35 total rankings)	
Grease (1978)	5
While You Were Sleeping (1995)	5
Sleepless in Seattle (1993)	5
So I Married an Axe Murderer (1993)	4
Pretty Woman (1990)	4
Mr. Holland's Opus (1995)	4

[1] Each user had given both their highest and lowest rating to more than three movies. These samples were selected in no particular order.

Problem 1– Results:

I do not particularly enjoy movies and had some difficulty choosing a user to serve as a substitute for myself. First, I eliminated user 49 because they gave *The Aristocats* (my favorite movie on the list) a rating of 2. I selected user 477 as the "substitute me" because they have rated few movies and gave them all high ratings. This suggests to me that they have not seen many movies and do not watch movies they do not think they will enjoy. In general, user 477's ratings seem to be mostly romantic comedies, which I find entertaining if sometimes a bit shallow. I also enjoyed both *Grease* and *While You Were Sleeping*, though I have never seen *Sleepless in Seattle*.

Problem 2:

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

Problem 2—Methods:

Correlation between users was calculated using the `topMatches()` function in `recommendations.py`[2]. The `topMatches()` function calculates a similarity score between a given user and each other user based on the ratings given by each user. As provided, the `topMatches()` function returns only the top 5 results. To access the negative correlations, the code limiting the returned data was removed, causing the function to return all correlation data. The highest 5 values and lowest 5 values were then selected from the list of correlated users.

Problem 2—Results:

The five users most correlated[3] to user 477 were user 857 (correlation 1.0), user 895 (correlation 1.0), user 875 (correlation 1.0), user 867 (correlation 1.0), and user 828 (correlation 1.0).

The five users least correlated to user 477 were user 173 (correlation -1.0), user 122 (correlation -1.0), user 100 (correlation -1.0), user 71 (correlation -1.0), and user 48 (correlation -1.0).

Problem 3:

Compute ratings for all the films that the 'substitute you' has 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).

Problem 3—Methods:

Ratings for movies that user 477 has not seen were calculated using the `getRecommendations()` function in `recommendations.py`. The `getRecommendations()` function calculates a similarity score between the given user and each other user, and then weighs the rating a user gave a movie using the similarity score. Ratings by users who are not similar to the given user (that is, a similarity score of zero or lower) are ignored. The ratings are then combined to provide a weighted average rating of the movie according to users similar to the given user. As provided, the `getRecommendations()` function calculates an average rating for

[2] https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py
[3] Ordering of users with the same correlation score is from highest user ID to lowest as a result of sorting the list.

all movies. The function was modified to recommend only movies with five or more reviews in order to prevent movies with a single high rating from dominating the results[4].

Problem 3—Results:

The top five recommended and bottom five non-recommended movies for user 477 are as follows:

Top 5 Recommendations	
To Live (Huozhe) (1994)	4.81
Wallace & Gromit: The Best of Aardman Animation (1996)	4.66
Schindler's List (1993)	4.60
Sum of Us, The (1994)	4.56
Wrong Trousers, The (1993)	4.56

Bottom 5 Recommendations	
McHale's Navy (1997)	1.46
Stupids, The (1996)	1.46
Jingle All the Way (1996)	1.43
Big Bully (1996)	1.31
Children of the Corn: The Gathering (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?

Problem 4—Methods:

Correlation between films was found using the `calculateSimilarItems()` function in `recommendations.py`. The `calculateSimilarItems()` function calls the `transformPrefs()` function to create a dictionary of items that have reviews instead of a dictionary of users who have reviewed items. It then calls the same `topMatches()` function used in Problem 2, passing the newly created item dictionary instead of the standard user dictionary. The resulting list of scores is used to create a dictionary of similarity scores between items. As provided, the `similarity=sim_`distance keyword argument is used to specify that the `topMatches()` function should calculate similarity scores using distance. This was not producing any scores below zero, so the argument was removed, and the default Pearson correlation coefficient was used instead.

My favorite movie on the list is *The Aristocats (1970)*, which I would give a rating of 5. My least favorite (that I have actually seen) is *What's Eating Gilbert Grape (1993)*, which I would give a rating of 2. The titles were used as keys to the dictionary created by `calculateSimilarItems()`. The top 5 values and bottom 5 values for each key were then pulled as the most correlated and least correlated movies.

[4] This was an actual problem. Before adding the limit, 'Santa With Muscles' was continually the top recommendation because its sole reviewer rated it '5' and had a high correlation with user 477. 'Santa With Muscles' is considered one of the worst movies ever made (https://en.wikipedia.org/wiki/Santa_with_Muscles).

Problem 4—Results:

The most- and least-correlated movies to *The Aristocats* and *What's Eating Gilbert Grape* are as follows:

Most like *What's Eating Gilbert Grape*	
Eve's Bayou (1997)	1.00
Believers, The (1987)	1.00
Even Cowgirls Get the Blues (1993)	1.00
Aparajito (1956)	1.00
Wedding Gift, The (1994)	1.0

Most like *The Aristocats*	
Fools Rush In (1997)	1.00
Associate, The (1996)	1.00
Women, The (1939)	1.0
Wings of the Dove, The (1997)	1.0
Wild Bill (1995)	1.0

Least like *What's Eating Gilbert Grape*	
Blues Brothers 2000 (1998)	-1.0
Beat the Devil (1954)	-1.0
Above the Rim (1994)	-1.0
Love and a .45 (1994)	-1.00
Gay Divorcee, The (1934)	-1.00

Least like *The Aristocats*	
Deconstructing Harry (1997)	-1.0
Calendar Girl (1993)	-1.0
Blown Away (1994)	-1.0
When the Cats Away (1996)[5]	-1.0
Before and After (1996)	-1.0

I have never heard of a single one of these movies, let alone seen them, and was left to judge them based on their Wikipedia pages.

Most of the movies suggested as like *What's Eating Gilbert Grape* are dramas with themes of family. Most of them also seem decently written but largely uninteresting (my chief complaint regarding *What's Eating Gilbert Grape).* This makes them appropriate recommendations. Most of the movies suggested as least like *What's Eating Gilbert Grape* are comedies or not meant to be taken seriously and are indeed not much like *What's Eating Gilbert Grape.*

While they have little else in common, none of the five movies suggested as least like *The Aristocats* seem entertaining or even particularly interesting. Interestingly, the same holds true for the five movies suggested as most like *The Aristocats*—none of their summaries make me interested in watching them.

Problem 4—Analysis:

Many of the movies reported as most similar to *The Aristocats* have very little in common with it. This is especially interesting when taken together with the observation that the comparison algorithm apparently considers 1995 Western *Wild Bill* (similarity 1.0) much more similar to *The Aristocats* than is *101 Dalmatians* (similarity 0.577). The Pearson correlation coefficient measures how closely related two terms are. When passed a dictionary of items, the `sim_pearson()` function provided by `recommendations.py` calculates the difference between ratings given by the same users to different films. The actual ratings are not considered, nor is the number of users who rated a given movie[6].

54 users rated *The Aristocats*. Of these 54, 24 also rated *101 Dalmatians* (Of the 17 users who rated *The Aristocats* at 3 or above, 14 also rated *101 Dalmatians* at 3 or above. Of the 54 users who rated *The Aristocats*, 2 also rated *Wild Bill*. One of these two users gave both movies a score of 3 and one gave both a score of 1.

Because the both users who rated both *Wild Bill* and *The Aristocats* rated it identically, the algorithm considers the two movies to be highly similar. Because the variation in ratings is higher, 101 Dalmatians is considered less similar despite the fact that a much greater number of users who liked The Aristocats also liked 101 Dalmatians ($14/17 \approx 82\%$) than they did *Wild Bill* ($1/17 \approx 6\%$).

[5] The original title, *Chacun cherche son chat*, is left out of the table to save space.

[6] This alone may make correlation a poor mechanism for recommending movies.