OLD DOMINION UNIVERSITY

Assignment 6

David Bayard

April 1, 2019

QUESTION 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?

Solution:

Three users were selected from the u.user file, based on their age, gender, and occupation. For each of these users, the top three rated movies and bottom three rated movies were extracted from the u.data and u.item files. In order for this to be accomplished, the loadMovieLens method was used to create a dictionary for the corresponding files, and the top three and bottom three entries for the dictionary were extracted, based on the user id, as shown below:

```
pref = recommendations.loadMovieLens()

# Get sorted list of user ratings
userRatings1 = (sorted(pref['368'].items(), key =
            lambda kv:(kv[1], kv[0])))
userRatings2 = (sorted(pref['81'].items(), key =
            lambda kv:(kv[1], kv[0])))
userRatings3 = (sorted(pref['135'].items(), key =
            lambda kv:(kv[1], kv[0])))

# Get top 5 for each user
userRatings1.reverse()
userRatings2.reverse()
userRatings3.reverse()
```

Listing 1: Loading user data

In the code above, each user has their data extracted and stored inside a dictionary. These dictionaries are then sorted, which gives direct access to their list of rated movies, and bottom rated movies through a simple call of .reverse().

Out of the three users, I relate most to user 368, except for films Jaws and Terminator 2, which I would have ranked higher. Although, except for a few outliers, user 368 shares my overall interest in movies.

QUESTION 2

Which 5 users are most correlated to the substitute you? Which 5 users are least correlated

Solution:

The functionality to determine the correlation of users is implemented in the recommendations.py file, which was provided by the Programming Collective Intelligence book. In this function, the Pearson score is used to compute the correlation between two users, as shown below:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Where:
x = User1's ratings of all iters rated by both users
y = User2's ratings of all items rated by both users
$\bar{x}$ = Mean of all User1's ratings
$\bar{y}$ = Mean of all User2's ratings

In the formula above, the variable r will represent the correlation between user one and user two, and this formula also accounts for inflation by normalizing the values between 0 and 1. This allows each user to be compared by reviewing their ratings per movie.

```
def topMatches(
    prefs,
    person,
    n=5,
    similarity=sim_pearson,
):
    '''
    Returns the best matches for person from the prefs dictionary.
    Number of results and similarity function are optional params.
    '''

    scores = [(similarity(prefs, person, other), other) for other in prefs
              if other != person]
    scores.sort()
    scores.reverse()
    return scores[0:n]
```

Listing 2: Getting User Objects

In the above code, the pearson score is calculated for the provided user, comparing that user to every other user in the list. This will produce an r, or correlation score which measures the similarity between the selected user and each of the users in the list. In order to get the lowest correlation users, the list is reversed by commenting the scores.reverse() function.

QUESTION 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

Solution:

The same process is followed as in question two, except the correlation will be measured per movie, instead of per user. The getRecommendations() method will be utilized in this instance, which will take the items, and score each by producing a weighted score. Then the ratings of all the other critics are taken and multiplied based on how similar they are to the fake me by the score they gave each movie. As in question two, the list returned by the getRecommendations() method may be reversed in order to get the movies with the lowest scores, thus being the least appealing for the fake me.

```
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
```

Listing 3: Function from Programming Collective Intelligence

QUESTION 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.

Solution:

In this instace, each film needs a correlation value, this means the key/value pair used to generate the earlier correlation values for each user must be transformed. This will allow a comparison per each movie, instead of comparing each user. The Pearson formula is used with this new transformed dictionary, and a list of movies with r values between zero and one will be generated. The method to perform this operation is called calculateSimilarItems(), and this method returns a list of movies with the highest correlation values, which may be reversed to access the lowest correlation movies.

```
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(itemPrefs, item, n=n, similarity=sim_distance)
        result[item] = scores
    return result
```

Listing 4: Function from Programming Collective Intelligence

As seen in the code above, the key/value pair is switched, allowing a direct comparison by rating, instead of by user. This code uses the Pearson correlation score to determine whether two movies have any correlation.

Top Correlation Values for Jurassic Park	
Name of Movie	Correlation
You So Crazy (1994)	1.0
Witness (1985)	1.0
Wings of Courage (1995)	1.0
Wife, The (1995)	1.0
Two Much (1996)	1.0

Bottom Correlation Values for Jurassic Park	
Name of Movie	Correlation
A Chef in Love (1996)	0
American Strays (1996)	0
August (1996)	0
Ayn Rand: A Sense of Life (1997)	0
B*A*P*S (1997)	0

Top Correlation Values for Children of the Corn	
Name of Movie	Correlation
Wooden Man's Bride, The (Wu Kui) (1994)	1.0
Winter Guest, The (1997)	1.0
Wild Reeds (1994)	1.0
White Balloon, The (1995)	1.0
When the Cats Away (Chacun cherche son chat) (1996)	1.0

Bottom Correlation Values for Children of the Corn	
Name of Movie	Correlation
'Til There Was You (1997)	0
8 Heads in a Duffel Bag (1997) (1996)	0
8 Seconds (1994) (1996)	0
Addicted to Love (1997)	0
Affair to Remember, An (1957)	0

My favorite film selected is Jurassic Park, and the least favorite film is Children of the Corn. I agree with most of the results from the list of recommended movies. I have seen Wings of Courage, and Two Much, and found them to be excellent movies. I read the summaries for the other three movies and it seems like I would enjoy them as well.

I also agree with the films that were selected as having low correlation values. They do not quite fit into the same genra as Children of the corn, but the summaries did not peak my interest, leading me to believe that I would most likely not enjoy these movies. With regards to the top correlation values for Children of the Corn, I am in complete agreement. I would not consider watching Children of the Corn, nor the rest of the movies that have a high correlation with it. Last of all, I have not seen any of the movies on the table of lowest correlations for Children of the Corn, but the movies do, in fact, have major differences from Children of Corn, making the low correlation reasonable.

Overall, I found the Pearson Correlation score to be mostly accurate. There are movies such as "You So Crazy" that I probably wouldn't want to watch, but in general most of the correlation scores match my expectations.