Assignment Seven

Jacob Berlin

CS432 – Spring 2016

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 ll"). This user is the "substitute you".

For questions 1, 2, 3, and 4 I used and referenced ‘recommendations.py’.

To start this assignment I downloaded the grouping of the 100k movie reviews off of http://grouplens.org/datasets/movielens/100k/ and sorted all of the data into their respective folder under ‘./Data/MovieLens’.

```
import argparse
import logging
import sys
#NOTE: THIS PROGRAM WAS CREATED BY KEVIN CLEMMONS. ALL CREDIT IS GIVEN TO HIM
from data_extractor.data_set import Data_Set

logging.basicConfig(level=logging.DEBUG,format='%(asctime)s %(name)-12s %(levelnam

defaultLogger = logging.getLogger('default')


def get_prefs(data):
    '''Create a dictionary of people and the movies that they have rated.
    '''
    # This code taken from page 26 in collective intelligence book

    movies = {}
    for movie in data.movie_list:
        movie_id = movie['movie_id']
        movie_title = movie['movie_title']
        movies[movie_id] = movie_title

    prefs = {}

    for dataPoint in data.rating_list:
        user_id = int(dataPoint['user_id'])
        movieId = dataPoint['item_id']
        rating = dataPoint['rating']
        prefs.setdefault(user_id,{})
        prefs[user_id][movies[movieId]] = float(rating)
    return prefs

# Returns the best matches for person from the prefs dictionary.
# Number of results and similarity function are optional params.
#def topMatches(prefs,person,n=5,similarity=sim_pearson):
#   scores=[(similarity(prefs,person,other),other)
                    #for other in prefs if other!=person]
#   scores.sort()
#   scores.reverse()
#   return scores[0:n]

def printResults(similarUsers):
    seperator = '--------------------------------------------------------'
    heading1 =  '------------------------Top-3---------------------------'
    heading2 =  '---------------------Bottom-Three-----------------------'
    heading3 =  '-----------------------Summary--------------------------'
    heading0 =  '---------------------Similar Users----------------------'
    print(heading0)
    for key in similarUsers.keys():

        print("User: {0}:".format(key))
```

This problem required us to take the files ‘u.user’, ‘u.data’, and ‘u.item’ which housed the data for 943 different users, 1700 different movies, and 100,000 individual ratings to link together three users most like ourselves based on the ratings that each had given six different movies, three of their favorite and three of their least. Given the preliminary information that I am a 21 year old male programmer who enjoys mainly science fiction or horror movies, my search was narrowed immensely. After modifying a program created by my classmate Kevin Clemmons named ‘substitute_you.py’ (partially shown above) which took in the parameters of the user’s age, gender, and occupation to generate the users who were most like myself, I came up with the data needed.

```
jacob_000@JakesLaptop ~/Desktop/CS432/assignmentSeven
$ python subsitute_you.py -a 21 -g m -o programmer
--------------------Similar Users--------------------
User: 603:
-----------------------Top-3-------------------------
1)Return of the Jedi (1983).......................5.0
2)Day the Earth Stood Still, The (1951)...........5.0
3)Twelve Monkeys (1995)...........................5.0
--------------------Bottom-Three---------------------
1)Heat (1995).....................................1.0
2)Platoon (1986)..................................1.0
3)Island of Dr. Moreau, The (1996)................2.0
-----------------------------------------------------
-----------------------------------------------------
User: 868:
-----------------------Top-3-------------------------
1)Star Trek: The Wrath of Khan (1982).............5.0
2)Wrong Trousers, The (1993)......................5.0
3)Taxi Driver (1976)..............................5.0
--------------------Bottom-Three---------------------
1)Shaggy Dog, The (1959)..........................1.0
2)Mission: Impossible (1996)......................1.0
3)Cool Runnings (1993)............................1.0
-----------------------------------------------------
-----------------------------------------------------
User: 671:
-----------------------Top-3-------------------------
1)Terminator 2: Judgment Day (1991)...............5.0
2)Executive Decision (1996).......................5.0
3)Desperado (1995)................................5.0
--------------------Bottom-Three---------------------
1)Pulp Fiction (1994).............................1.0
2)Tough and Deadly (1995).........................1.0
3)Romper Stomper (1992)...........................1.0
-----------------------------------------------------
-----------------------------------------------------
----------------------Summary------------------------
Number of Users...................................3
(venv)
```

(5.0 for top rated, 1.0 for bottom rated)

<u>User 603:</u>
Top: Return of the Jedi (1983), The Day the Earth Stood Still (1951), Twelve Monkeys (1995)
Bottom: Heat (1995), Platoon (1986), The Island of Dr. Moreau (1996)
<u>User 868:</u>
Top: Star Trek: The Wrath of Khan (1982), The Wrong Trousers (1993), Taxi Driver (1976)
Bottom: The Shaggy Dog (1959), Mission: Impossible (1996), Cool Runnings (1993)
<u>User 671:</u>
Top: Terminator 2: Judgement Day (1991), Executive Decision (1996), Desperado (1995)
Bottom: Pulp Fiction (1994), Tough and Deadly (1995), Rough Stomper (1992)

From the program I received a choice of three different people from the 'u.users' file who were all 21 year old male programmers. Now, specifically because User 671 rated 'Pulp Fiction (1944)' as their least favorite movie of all time, I immediately crossed him off of my list of potential substitutes. User 868 has a strong lineup of favorite movies with 'Star Trek: The Wrath of Khan (1982)' being the most notable, however User 603's top rated movie, 'Return of the Jedi (1983),' is my favorite movie of all time…thus making this an easy choice. Although 'Twelve Monkeys (1995)' would not be close to one of my top movies, I generally agree with most of 603's choices, therefore User 603 will be my 'substitute me.'

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

To calculate which users are most and least correlated to my substitute me I used the sim_pearson correlation score calculator in ‘recommendations.py’.

```
def sim_pearson(prefs, p1, p2):
    '''
    Returns the Pearson correlation coefficient for p1 and p2.
    '''

    # Get the list of mutually rated items
    si = {}
    for item in prefs[p1]:
        if item in prefs[p2]:
            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[p1][it] for it in si])
    sum2 = sum([prefs[p2][it] for it in si])
    # Sums of the squares
    sum1Sq = sum([pow(prefs[p1][it], 2) for it in si])
    sum2Sq = sum([pow(prefs[p2][it], 2) for it in si])
    # Sum of the products
    pSum = sum([prefs[p1][it] * prefs[p2][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
```

```
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]

def bottomMatches(
    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]
```

With list of the users and the items each user has rated the sim_pearson is able to calculate a correlation score based off of who rated movies similarly to my user. The ‘topMatches’ and ‘bottomMatches’ functions were called to actually get the list of the greatest and least correlated users. A score of 1.0 signifies that the correlated user is extremely similar to substitute me, a score of -1.0 means that the user rates movies exactly the opposite of how substitute me would.

```
>>> recommendations.bottomMatches(pref, '603', n=5)
[(-1.0, '220'), (-1.0, '252'), (-1.0, '266'), (-1.0, '300'), (-1.0, '34')]
>>> recommendations.topMatches(pref, '603', n=5)
[(1.0000000000000016, '182'), (1.0000000000000004, '939'), (1.0, '920'), (1.0, '915'), (1.0, '873')]
```

Most correlation to 603:	Value:	u.user information:
User 182	1.0	182|36|M|programmer|33884
User 939	1.0	939|26|F|student|33319
User 920	1.0	920|30|F|artist|90008
User 915	1.0	915|50|M|entertainment|60614
User 873	1.0	873|48|F|administrator|33763
Least correlation to 603:	Value:	u.user information:
User 220	-1.0	220|30|M|librarian|78205
User 252	-1.0	252|42|M|engineer|07733
User 266	-1.0	266|62|F|administrator|78756
User 300	-1.0	300|26|F|programmer|55106
User 34	-1.0	34|38|F|administrator|42141

3. Compute ratings for all the films that the substitute you hasn't 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).

In the file 'recommendations.py' there is a function called 'getRecommendations' that will take the list of rankings for the movies that substitute me has made and generate ratings based off of those statistics on movies that substitute me has not seen depending on what other users like me would rate them (all based off of the correlation score).

```
def getRecommendations(prefs, person, similarity=sim_pearson):
    '''
    Gets recommendations for a person by using a weighted average
    of every other user's rankings
    '''

    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
```

I ran the function and received the following output.

```
>>> recommendations.getRecommendations(pref, '603')
[(5.0, 'Wrong Trousers, The (1993)'), (5.0, 'When We Were Kings (1996)'), (5.0, 'Toy Story (1995)'), (
5.0, 'Tombstone (1993)'), (5.0, 'Time to Kill, A (1996)'), (5.0, "Sophie's Choice (1982)"), (5.0, 'Shi
ning, The (1980)'), (5.0, "Schindler's List (1993)"), (5.0, 'Ran (1985)'), (5.0, 'Notorious (1946)'),
(5.0, 'North by Northwest (1959)'), (5.0, 'Nikita (La Femme Nikita) (1990)'), (5.0, 'Monty Python and
the Holy Grail (1974)'), (5.0, 'Lawrence of Arabia (1962)'), (5.0, 'Jaws (1975)'), (5.0, 'Immortal Bel
oved (1994)'), (5.0, 'Godfather, The (1972)'), (5.0, 'Glory (1989)'), (5.0, 'Father of the Bride (1950
)'), (5.0, 'Contact (1997)'), (5.0, 'Close Shave, A (1995)'), (5.0, 'Charade (1963)'), (5.0, 'Bridge o
n the River Kwai, The (1957)'), (5.0, 'Babe (1995)'), (5.0, 'African Queen, The (1951)'), (4.0, 'Willy
 Wonka and the Chocolate Factory (1971)'), (4.0, "William Shakespeare's Romeo and Juliet (1996)"), (4.
0, 'Waiting for Guffman (1996)'), (4.0, 'Vertigo (1958)'), (4.0, 'Truth About Cats & Dogs, The (1996)'
), (4.0, 'To Catch a Thief (1955)'), (4.0, 'This Is Spinal Tap (1984)'), (4.0, 'Terminator, The (1984)
'), (4.0, 'Terminator 2: Judgment Day (1991)'), (4.0, 'Stand by Me (1986)'), (4.0, 'Sleepless in Seatt
le (1993)'), (4.0, 'Secret of Roan Inish, The (1994)'), (4.0, 'Sabrina (1995)'), (4.0, 'Room with a Vi
ew, A (1986)'), (4.0, 'Raising Arizona (1987)'), (4.0, 'Quiz Show (1994)'), (4.0, 'Psycho (1960)'), (4
.0, 'Professional, The (1994)'), (4.0, 'Primary Colors (1998)'), (4.0, 'Peacemaker, The (1997)'), (4.0
, "One Flew Over the Cuckoo's Nest (1975)"), (4.0, 'Night of the Living Dead (1968)'), (4.0, 'My Left
Foot (1989)'), (4.0, 'Mr. Smith Goes to Washington (1939)'), (4.0, 'Mission: Impossible (1996)'), (4.0
, 'Men in Black (1997)'), (4.0, 'Magnificent Seven, The (1954)'), (4.0, 'M*A*S*H (1970)'), (4.0, 'Kill
ing Fields, The (1984)'), (4.0, 'Kids in the Hall: Brain Candy (1996)'), (4.0, 'Jurassic Park (1993)')
, (4.0, "Jackie Chan's First Strike (1996)"), (4.0, 'Independence Day (ID4) (1996)'), (4.0, 'Hunt for
Red October, The (1990)'), (4.0, 'Highlander (1986)'), (4.0, 'Henry V (1989)'), (4.0, 'Heavy Metal (19
81)'), (4.0, 'Hamlet (1996)'), (4.0, 'Groundhog Day (1993)'), (4.0, 'Grosse Pointe Blank (1997)'), (4.
0, 'Graduate, The (1967)'), (4.0, 'GoodFellas (1990)'), (4.0, 'Good, The Bad and The Ugly, The (1966)'
), (4.0, 'Good Will Hunting (1997)'), (4.0, 'Godfather: Part II, The (1974)'), (4.0, 'Gattaca (1997)')
, (4.0, 'Fugitive, The (1993)'), (4.0, 'Four Weddings and a Funeral (1994)'), (4.0, 'Fly Away Home (19
96)'), (4.0, 'Fish Called Wanda, A (1988)'), (4.0, 'Evil Dead II (1987)'), (4.0, 'Emma (1996)'), (4.0,
 'Die Hard (1988)'), (4.0, 'Dead Poets Society (1989)'), (4.0, 'Dave (1993)'), (4.0, 'Crimson Tide (19
95)'), (4.0, 'Courage Under Fire (1996)'), (4.0, 'Cool Hand Luke (1967)'), (4.0, 'Cook the Thief His W
ife & Her Lover, The (1989)'), (4.0, 'Conspiracy Theory (1997)'), (4.0, 'Casablanca (1942)'), (4.0, 'B
razil (1985)'), (4.0, 'Big Night (1996)'), (4.0, 'Ben-Hur (1959)'), (4.0, 'Back to the Future (1985)')
, (4.0, 'Apollo 13 (1995)'), (4.0, 'American in Paris, An (1951)'), (4.0, 'American Werewolf in London
, An (1981)'), (4.0, 'Aladdin (1992)'), (4.0, 'Abyss, The (1989)'), (4.0, '2001: A Space Odyssey (1968
)'), (3.0, 'While You Were Sleeping (1995)'), (3.0, 'Victor/Victoria (1982)'), (3.0, 'Unforgiven (1992
)'), (3.0, 'Twister (1996)'), (3.0, 'Top Gun (1986)'), (3.0, 'Secret Garden, The (1993)'), (3.0, 'Rumb
le in the Bronx (1995)'), (3.0, 'Rock, The (1996)'), (3.0, 'River Wild, The (1994)'), (3.0, 'Right Stu
ff, The (1983)'), (3.0, 'Replacement Killers, The (1998)'), (3.0, 'Patton (1970)'), (3.0, 'Outbreak (1
995)'), (3.0, 'My Fair Lady (1964)'), (3.0, "Mr. Holland's Opus (1995)"), (3.0, "Monty Python's Life o
f Brian (1979)"), (3.0, 'Mimic (1997)'), (3.0, 'Metro (1997)'), (3.0, 'Mars Attacks! (1996)'), (3.0, '
Long Kiss Goodnight, The (1996)'), (3.0, 'Last of the Mohicans, The (1992)'), (3.0, 'Kull the Conquero
r (1997)'), (3.0, "It's a Wonderful Life (1946)"), (3.0, 'It Happened One Night (1934)'), (3.0, 'How t
o Be a Player (1997)'), (3.0, 'Field of Dreams (1989)'), (3.0, 'Face/Off (1997)'), (3.0, 'Die Hard 2 (
1990)'), (3.0, 'Dances with Wolves (1990)'), (3.0, 'Crumb (1994)'), (3.0, 'Con Air (1997)'), (3.0, 'Cl
ient, The (1994)'), (3.0, 'Clear and Present Danger (1994)'), (3.0, "Breakfast at Tiffany's (1961)"),
(3.0, 'Body Snatchers (1993)'), (3.0, 'Batman (1989)'), (3.0, 'American President, The (1995)'), (3.0,
 'Aladdin and the King of Thieves (1996)'), (3.0, 'Addicted to Love (1997)'), (2.0, "My Best Friend's
Wedding (1997)"), (2.0, 'Lost World: Jurassic Park, The (1997)'), (2.0, 'Grand Day Out, A (1992)'), (2
.0, 'Gone with the Wind (1939)'), (2.0, 'Ghost and the Darkness, The (1996)'), (2.0, 'Full Metal Jacke
t (1987)'), (2.0, 'Escape from L.A. (1996)'), (2.0, 'E.T. the Extra-Terrestrial (1982)'), (2.0, 'Buddy
 (1997)'), (2.0, 'Broken Arrow (1996)'), (2.0, 'Austin Powers: International Man of Mystery (1997)'),
(2.0, 'Air Force One (1997)'), (1.0, 'Turbulence (1997)'), (1.0, 'Trainspotting (1996)'), (1.0, 'That
Darn Cat! (1965)'), (1.0, 'Super Mario Bros. (1993)'), (1.0, 'Spawn (1997)'), (1.0, 'Sound of Music, T
he (1965)'), (1.0, 'Pretty Woman (1990)'), (1.0, 'Lost Highway (1997)'), (1.0, 'Lion King, The (1994)'
), (1.0, 'Leave It to Beaver (1997)'), (1.0, 'Last Man Standing (1996)'), (1.0, 'Grumpier Old Men (199
5)'), (1.0, 'Father of the Bride Part II (1995)'), (1.0, 'English Patient, The (1996)'), (1.0, 'Dragon
heart (1996)'), (1.0, 'Crash (1996)'), (1.0, 'Clockwork Orange, A (1971)'), (1.0, 'Cable Guy, The (199
6)'), (1.0, 'Breakdown (1997)'), (1.0, 'Boogie Nights (1997)'), (1.0, 'Beauty and the Beast (1991)'),
(1.0, 'Bean (1997)'), (1.0, 'Anaconda (1997)'), (1.0, 'Air Bud (1997)')]
```

(5.0 rating for movies that substitute me should see, 1.0 for movies that substitute me would hate)

Movies that substitute me should see: (all 5.0 ratings)

1. The Wrong Trousers (1993)
2. When We Were Kings (1996)
3. Toy Story (1995)
4. Tombstone (1993)
5. A Time to Kill (1996)

Movies that substitute me would hate: (all 1.0 ratings)

1. Air Bud (1997)
2. Anaconda (1997)
3. Bean (1997)
4. Beauty and the Beast (1991)
5. Boogie Nights (1997)

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?

To start this problem I picked out my personal favorite movie from the list, 'Return of the Jedi (1983)' and my least favorite movie, 'Batman & Robin (1997)'. I don't think that I would need to explain myself for either of them…especially the latter.

In 'recommendations.py' there is a function called 'calculateSimilarItems' that takes the entire MovieLens list and gives the n amount of most correlated and n amount of least correlated films for each movie. I ran the function twice to get the most correlated first and the least correlated after. All of the compilation data is stored in 'movieresult.txt'.

```python
def calculateSimilarItems(prefs, n=10):
    '''
    Create a dictionary of items showing which other items they are
    most similar to.
    '''
    outputFile = open('movieresult.txt', 'w')
    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 = bottomMatches(itemPrefs, item, n=n, similarity=sim_distance)
        result[item] = scores
    outputFile.write(str(result))
    outputFile.close()
    return result
```

First compilation:

```
>>> recommendations.calculateSimilarItems(pref, n=5)
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
{'Adventures of Pinocchio, The (1996)': [(1.0, 'Year of the Horse (1997)'), (1.0, "Wooden Man's Bride, The (Wu Kui) (1994)"), (1.0, 'Wonderland (1997)'), (1.0, 'Withnail and I (1987)'), (1.0, 'Wild Reeds (1994)
')], 'To Cross the Rubicon (1991)': [(1.0, 'Yankee Zulu (1994)'), (1.0, 'Wyatt Earp (1994)'), (1.0, 'Wrong Trousers, The (1993)'), (1.0, 'Woman in Question, The (1950)'), (1.0, 'Wolf (1994)')], 'Birdcage, The (
1996)': [(1.0, '\xc1 k\xf6ldum klaka (Cold Fever) (1994)'), (1.0, 'You So Crazy (1994)'), (1.0, 'Witness (1985)'), (1.0, 'Wings of Courage (1995)'), (1.0, 'Wife, The (1995)')], 'B. Monkey (1998)': [(1.0, 'Saint
, The (1997)'), (1.0, 'Red Corner (1997)'), (1.0, 'Mrs. Dalloway (1997)'), (1.0, 'Mimic (1997)'), (1.0, 'Mad City (1997)')], 'Little Big League (1994)': [(1.0, 'Zeus and Roxanne (1997)'), (1.0, 'Yankee Zulu (19
94)'), (1.0, 'Woman in Question, The (1950)'), (1.0, 'Withnail and I (1987)'), (1.0, 'Wings of Desire (1987)')], 'Magnificent Seven, The (1954)': [(1.0, 'Yankee Zulu (1994)'), (1.0, 'Woman in Question, The (195
0)'), (1.0, 'Wife, The (1995)'), (1.0, 'Whole Wide World, The (1996)'), (1.0, 'When the Cats Away (Chacun cherche son chat) (1996)')], 'Chairman of the Board (1998)': [(1.0, 'Wings of the Dove, The (1997)'), (1
.0, 'Welcome To Sarajevo (1997)'), (1.0, 'Swept from the Sea (1997)'), (1.0, 'Spawn (1997)'), (1.0, 'Shadow Conspiracy (1997)')], 'Mortal Kombat (1995)': [(1.0, 'Yankee Zulu (1994)'), (1.0, 'Woman in Question,
The (1950)'), (1.0, 'Withnail and I (1987)'), (1.0, 'Winter Guest, The (1997)'), (1.0, 'When a Man Loves a Woman (1994)')], 'Tin Men (1987)': [(1.0, '\xc1 k\xf6ldum klaka (Cold Fever) (1994)'), (1.0, 'Yankee Zu
lu (1994)'), (1.0, 'Wonderland (1997)'), (1.0, 'Woman in Question, The (1950)'), (1.0, 'Wings of Courage (1995)')], 'Endless Summer 2, The (1994)': [(1.0, 'Yankee Zulu (1994)'), (1.0, 'Woman in Question, The (1
950)'), (1.0, 'Wolf (1994)'), (1.0, 'Withnail and I (1987)'), (1.0, 'With Honors (1994)')], 'Village of the Damned (1995)': [(1.0, 'Winter Guest, The (1997)'), (1.0, 'Wife, The (1995)'), (1.0, "Widows' Peak (19
94)"), (1.0, 'Vermin (1998)'), (1.0, 'Underworld (1997)')], 'Careful (1992)': [(1.0, 'Yankee Zulu (1994)'), (1.0, 'Wyatt Earp (1994)'), (1.0, 'Wrong Trousers, The (1993)'), (1.0, 'Woman in Question, The (1950)'
), (1.0, 'Wolf (1994)')], 'Air Bud (1997)': [(1.0, "Wooden Man's Bride, The (Wu Kui) (1994)"), (1.0, 'Women, The (1939)'), (1.0, 'Winter Guest, The (1997)'), (1.0, 'Wild Reeds (1994)'), (1.0, "White Man's Burde
n (1995)")], 'Johnny 100 Pesos (1993)': [(1.0, "Young Poisoner's Handbook, The (1995)"), (1.0, 'Year of the Horse (1997)'), (1.0, "Wooden Man's Bride, The (Wu Kui) (1994)"), (1.0, 'Wonderland (1997)'), (1.0, 'W
ishmaster (1997)')], 'Bye Bye, Love (1995)': [(1.0, 'Wings of the Dove, The (1997)'), (1.0, "Widows' Peak (1994)"), (1.0, 'Welcome to the Dollhouse (1995)'), (1.0, 'Walkabout (1971)'), (1.0, 'Waiting to Exhale
(1995)')], 'Men With Guns (1997)': [(1.0, '\xc1 k\xf6ldum klaka (Cold Fever) (1994)'), (1.0, 'Young Frankenstein (1974)'), (1.0, 'Wizard of Oz, The (1939)'), (1.0, 'Wings of the Dove, The (1997)'), (1.0, 'Wild
Things (1998)')], 'Pompatus of Love, The (1996)': [(1.0, 'Young Guns II (1990)'), (1.0, 'Year of the Horse (1997)'), (1.0, 'Wyatt Earp (1994)'), (1.0, "Wooden Man's Bride, The (Wu Kui) (1994)"), (1.0, 'Wonderla
nd (1997)')], '2 Days in the Valley (1996)': [(1.0, '\xc1 k\xf6ldum klaka (Cold Fever) (1994)'), (1.0, 'Year of the Horse (1997)'), (1.0, "Wooden Man's Bride, The (Wu Kui) (1994)"), (1.0, 'Wonderland (1997)'),
(1.0, 'Wild Things (1998)')], 'Jerky Boys, The (1994)': [(1.0, 'Yankee Zulu (1994)'), (1.0, 'Wyatt Earp (1994)'), (1.0, 'Woman in Question, The (1950)'), (1.0, 'Wolf (1994)'), (1.0, 'Withnail and I (1987)')], '
Close Shave, A (1995)': [(1.0, "Wooden Man's Bride, The (Wu Kui) (1994)"), (1.0, 'Wonderland (1997)'), (1.0, 'Winter Guest, The (1997)'), (1.0, 'Wild Reeds (1994)'), (1.0, 'Welcome To Sarajevo (1997)')], 'Amade
```

Second compilation:

```
>>> recommendations.calculateSimilarItems(pref, n=5)
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
{'Adventures of Pinocchio, The (1996)': [(0, "'Til There Was You (1997)"), (0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (0, '8 Heads in a Duffel Bag (1997)'), (0, 'A Chef in Love (1996)'), (0, 'Across th
e Sea of Time (1995)')], 'To Cross the Rubicon (1991)': [(0, "'Til There Was You (1997)"), (0, '101 Dalmatians (1996)'), (0, '187 (1997)'), (0, '2 Days in the Valley (1996)'), (0, '3 Ninjas: High Noon At Mega M
ountain (1998)')], 'Birdcage, The (1996)': [(0, 'Aiqing wansui (1994)'), (0, 'All Things Fair (1996)'), (0, 'Aparajito (1956)'), (0, 'B. Monkey (1998)'), (0, 'Babyfever (1994)')], 'B. Monkey (1998)': [(0, "'Til
 There Was You (1997)"), (0, '1-900 (1994)'), (0, '101 Dalmatians (1996)'), (0, '12 Angry Men (1957)'), (0, '2 Days in the Valley (1996)')], 'Little Big League (1994)': [(0, "'Til There Was You (1997)"), (0, '1
87 (1997)'), (0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (0, '8 Heads in a Duffel Bag (1997)'), (0, '8 Seconds (1994)')], 'Magnificent Seven, The (1954)': [(0, '8 Heads in a Duffel Bag (1997)'), (0, 'Ai
qing wansui (1994)'), (0, 'Albino Alligator (1996)'), (0, 'All Over Me (1997)'), (0, 'All Things Fair (1996)')], 'Chairman of the Board (1998)': [(0, '1-900 (1994)'), (0, '12 Angry Men (1957)'), (0, '2 Days in
the Valley (1996)'), (0, '20,000 Leagues Under the Sea (1954)'), (0, '2001: A Space Odyssey (1968)')], 'Mortal Kombat (1995)': [(0, "'Til There Was You (1997)"), (0, '8 Seconds (1994)'), (0, 'A Chef in Love (19
96)'), (0, 'Aiqing wansui (1994)'), (0, 'All Dogs Go to Heaven 2 (1996)')], 'Tin Men (1987)': [(0, '8 Heads in a Duffel Bag (1997)'), (0, '8 Seconds (1994)'), (0, 'Aiqing wansui (1994)'), (0, 'Albino Alligator
(1996)'), (0, 'All Things Fair (1996)')], 'Endless Summer 2, The (1994)': [(0, "'Til There Was You (1997)"), (0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (0, '8 Seconds (1994)'), (0, 'A Chef in Love (199
6)'), (0, 'Above the Rim (1994)')], 'Village of the Damned (1995)': [(0, "'Til There Was You (1997)"), (0, '8 Heads in a Duffel Bag (1997)'), (0, '8 Seconds (1994)'), (0, 'A Chef in Love (1996)'), (0, 'Above th
e Rim (1994)')], 'Careful (1992)': [(0, "'Til There Was You (1997)"), (0, '101 Dalmatians (1996)'), (0, '187 (1997)'), (0, '2 Days in the Valley (1996)'), (0, '3 Ninjas: High Noon At Mega Mountain (1998)')], 'A
ir Bud (1997)': [(0, '8 1/2 (1963)'), (0, '8 Seconds (1994)'), (0, 'Affair to Remember, An (1957)'), (0, 'Aiqing wansui (1994)'), (0, 'Albino Alligator (1996)')], 'Johnny 100 Pesos (1993)': [(0, "'Til There Was
 You (1997)"), (0, '12 Angry Men (1957)'), (0, '20,000 Leagues Under the Sea (1954)'), (0, '2001: A Space Odyssey (1968)'), (0, '3 Ninjas: High Noon At Mega Mountain (1998)')], 'Bye Bye, Love (1995)': [(0, "'Ti
l There Was You (1997)"), (0, '1-900 (1994)'), (0, '187 (1997)'), (0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (0, '8 Seconds (1994)')], 'Men With Guns (1997)': [(0, "'Til There Was You (1997)"), (0, '1-
900 (1994)'), (0, '101 Dalmatians (1996)'), (0, '20,000 Leagues Under the Sea (1954)'), (0, '3 Ninjas: High Noon At Mega Mountain (1998)')], 'Pompatus of Love, The (1996)': [(0, '20,000 Leagues Under the Sea (1
954)'), (0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (0, '39 Steps, The (1935)'), (0, '8 1/2 (1963)'), (0, '8 Heads in a Duffel Bag (1997)')], '2 Days in the Valley (1996)': [(0, '8 Heads in a Duffel Bag
 (1997)'), (0, 'Aiqing wansui (1994)'), (0, 'All Things Fair (1996)'), (0, "Amityville 1992: It's About Time (1992)"), (0, 'Amityville Curse, The (1990)')], 'Jerky Boys, The (1994)': [(0, "'Til There Was You (1
997)"), (0, '101 Dalmatians (1996)'), (0, '187 (1997)'), (0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (0, '39 Steps, The (1935)')], 'Close Shave, A (1995)': [(0, '8 Heads in a Duffel Bag (1997)'), (0, '8
 Seconds (1994)'), (0, 'A Chef in Love (1996)'), (0, 'Above the Rim (1994)'), (0, 'Afterglow (1997)')], 'Amadeus (1984)': [(0, 'American Strays (1996)'), (0, 'August (1996)'), (0, 'B. Monkey (1998)'), (0, 'Ball
ad of Narayama, The (Narayama Bushiko) (1958)'), (0, 'Big Bang Theory, The (1994)')], 'Critical Care (1997)': [(0, "'Til There Was You (1997)"), (0, '8 Heads in a Duffel Bag (1997)'), (0, '8 Seconds (1994)'), (
0, 'A Chef in Love (1996)'), (0, 'Above the Rim (1994)')], 'Clockwork Orange, A (1971)': [(0, 'Aiqing wansui (1994)'), (0, 'American Strays (1996)'), (0, 'August (1996)'), (0, 'B. Monkey (1998)'), (0, 'Big Bang
 Theory, The (1994)')], 'Country Life (1994)': [(0, "'Til There Was You (1997)"), (0, '101 Dalmatians (1996)'), (0, '187 (1997)'), (0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (0, '39 Steps, The (1935)')
], 'Boys in Venice (1996)': [(0, "'Til There Was You (1997)"), (0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (0, '39 Steps, The (1935)'), (0, '8 Heads in a Duffel Bag (1997)'), (0, '8 Seconds (1994)')], '
Wings of Courage (1995)': [(0, "'Til There Was You (1997)"), (0, '1-900 (1994)'), (0, '101 Dalmatians (1996)'), (0, '187 (1997)'), (0, '2 Days in the Valley (1996)')], 'Fifth Element, The (1997)': [(0, '3 Ninja
s: High Noon At Mega Mountain (1998)'), (0, 'All Things Fair (1996)'), (0, 'Amityville: Dollhouse (1996)'), (0, 'An Unforgettable Summer (1994)'), (0, 'Angel on My Shoulder (1946)')], 'Heaven & Earth (1993)': [
(0, '1-900 (1994)'), (0, '8 1/2 (1963)'), (0, '8 Heads in a Duffel Bag (1997)'), (0, '8 Seconds (1994)'), (0, 'A Chef in Love (1996)')], 'Paris, Texas (1984)': [(0, "'Til There Was You (1997)"), (0, '8 Heads in
 a Duffel Bag (1997)'), (0, '8 Seconds (1994)'), (0, 'A Chef in Love (1996)'), (0, 'Above the Rim (1994)')], 'Tombstone (1993)': [(0, 'A Chef in Love (1996)'), (0, 'Aiqing wansui (1994)'), (0, 'All Things Fair
(1996)'), (0, 'American Strays (1996)'), (0, 'Angel Baby (1995)')], 'Lawnmower Man 2: Beyond Cyberspace (1996)': [(0, "'Til There Was You (1997)"), (0, '8 Heads in a Duffel Bag (1997)'), (0, '8 Seconds (1994)')
, (0, 'A Chef in Love (1996)'), (0, 'Above the Rim (1994)')], 'Howling, The (1981)': [(0, "'Til There Was You (1997)"), (0, '1-900 (1994)'), (0, '8 Heads in a Duffel Bag (1997)'), (0, 'A Chef in Love (1996)'),
(0, 'Aiqing wansui (1994)')], 'Mouse Hunt (1997)': [(0, '1-900 (1994)'), (0, '8 Seconds (1994)'), (0, 'Aiqing wansui (1994)'), (0, 'Albino Alligator (1996)'), (0, 'All Things Fair (1996)')], 'Of Human Bondage (
1934)': [(0, "'Til There Was You (1997)"), (0, '1-900 (1994)'), (0, '187 (1997)'), (0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (0, '8 Heads in a Duffel Bag (1997)')], 'Queen Margot (Reine Margot, La) (1
```

Finally, with all of the data in the text file, I then took out all of the information pertaining to both of my chosen movies and came out with these results: (1.0 for most correlated, 0 for least correlated)

'Return of the Jedi (1983)': Most correlated

[(1.0, 'klaka (Cold Fever) (1994)'),

(1.0, "Wooden Man's Bride, The (Wu Kui) (1994)"),

(1.0, 'Witness (1985)'),

(1.0, 'Wings of Courage (1995)'),

(1.0, 'Wedding Gift, The (1994)')],

'Return of the Jedi (1983)': Least correlated

[(0, 'All Things Fair (1996)'),

(0, 'August (1996)'),

(0, 'B. Monkey (1998)'),

(0, 'Big Bang Theory, The (1994)'),

(0, 'Bird of Prey (1996)')]

'Batman & Robin (1997)': Most correlated

[(1.0, 'Van, The (1996)'),

(1.0, 'Vampire in Brooklyn (1995)'),

(1.0, 'Two if by Sea (1996)'),

(1.0, 'Twin Town (1997)'),

(1.0, 'Turning, The (1992)')],

'Batman & Robin (1997)': Least correlated

[(0, '3 Ninjas: High Noon At Mega Mountain (1998)'),

(0, '8 Seconds (1994)'),

(0, 'Afterglow (1997)'),

(0, 'All Things Fair (1996)'),

(0, 'American Buffalo (1996)')],

Based on the results that I received for 'Return of the Jedi (1983)' and 'Batman & Robin (1997)', I do not agree with the results on the sole grounds that I do not think that I have heard of any single one of the movies listed under either the most correlated or least correlated movies. I think it has something to do with my age being only 21.

Recommendations.py used and referenced from:

“https://github.com/Jberl002/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py”