HW 7 - Recommendation Systems

Adeniran Adeniyi

DUE Sunday, April 11, 2021 before 11:59pm

Q1

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?
- what are their bottom 3 (least favorite) films?

Answer

```python
# -*- coding: utf-8 -*-
"""
Created on Wed Apr  7 20:09:11 2021

@author: aadeniran
"""
import pandas as pd

def closetToMeAgeGendOccu(item,data):

    #load movies data into variables
    movies = {}
    for line in open(str(item)):
        (id, title) = line.split('|')[0:2]
        movies[id] = title
    # Load data
    prefs = {}
    for line in open(str(data)):
        (user, movieid, rating, ts) = line.split('\t')
        prefs.setdefault(user, {})
        prefs[user][movies[movieid]] = float(rating)
    #Load in users
    users = []
    for line in open('movie/u.user'):
        (user, age, gender, occupation, zipcode) = line.split('|')
        dictA = {'user': user, 'age':age, 'gender': gender, 'occupation
    ' :occupation }
        users.append(dictA)

    count = 1
    #hold 3 similar user in term of age gender and category
    similarUser =[]
    #Find all users that have things common with me
    for a in users:
        """
        Here i parsed in my age gender and occupation
        to find users of my age gender and occupation category.
        I take only 3 users
        """
        if a['age'] == '24' and a['gender'] == 'M' and a['occupation']
  == 'student' and count < 4 :
            similarUser.append(a['user'])
            count += 1

    #print user movie likes
    for a in similarUser:
        """
        Based on similar users like me,
        I obtained these similar users top 3 favorite films
        """
        if a in prefs:
            #this gets the movie names, and the movie rating per user
            p = pd.DataFrame(list(prefs[a].items()),columns=["Movie
  Names","Rating"])
            #this sort the users movie list based on rating in
  decending order
            p.sort_values("Rating",ascending=False, inplace=True)
            #gets the top3 films
            top3 = p.head(3)
            #gets the bottom 3 films
            bottom3 = p.tail(3)
            #print("Top 3 films for User {}:".format(a))
            #print(top3)
            #print("\n\n")
            #print("Bottom 3 films for User {}:".format(a))
            #print(bottom3)
            #print("\n")
            top3.to_csv("Q1/"+a+"top3.csv",index=False)
            bottom3.to_csv("Q1/"+a+"bottom.csv",index=False)



"""
Question 1 solution
"""
```

```python
closetToMeAgeGendOccu('movie/u.item','movie/u.data')
```

Listing 1: movieLen.py

Table 1: Top 3 movies for user 73

Movie Names	Ratings
Three Colors: Red (1994)	5.0
Godfather: Part II, The (1974)	5.0
2001: A Space Odyssey (1968)	5.0

Table 2: Bottom 3 movies for user 73

Movie Names	Ratings
Saint, The (1997)	2.0
Home Alone 3 (1997)	1.0
Home Alone (1990)	1.0

Table 3: Top 3 movies for user 301

Movie Names	Ratings
It's a Wonderful Life (1946)	5.0
Empire Strikes Back, The (1980)	5.0
Star Wars (1977)	5.0

Table 4: Bottom 3 movies for user 301

Movie Names	Ratings
Ready to Wear (Pret-A-Porter) (1994)	1.0
Dirty Dancing (1987)	1.0
Natural Born Killers (1994)	1.0

Table 5: Top 3 movies for user 369

Movie Names	Ratings
Dead Poets Society (1989)	5.0
Return of the Jedi (1983)	5.0
Wallace & Gromit: The Best of Aardman Animation (1996)	5.0

Table 6: Bottom 3 movies for user 369

Movie Names	Ratings
Beautician and the Beast, The (1997)	3.0
How to Be a Player (1997)	2.0
Booty Call (1997)	2.0

Discussion

I created a function called closetToMeAgeGendOccu(),in line 9. This function takes two parameters of the location of u.item and u.data path. u.user was not going through as a parameter so i parsed it directly. From lines 11-27 I read in the datas from each file and store it into dictionary varaibles. In Line 29 -41, I found users that were just as similar to me in age gender and occupation. I limited this search to find only just 3 similar users to me on line 35. These users where stored in a list variable called similarUser. line 31 is the variable declaration, while line 40 appends new users to the list in the created variable

Lines 44 to 65, process the outcomes for top 3 movies and bottom 3 movies for each users that I found. Using a for loop to retrieve each user, parse this result to ensure that the said user is in prefs dictionary. If present, retrieve the row of the dictionary of that particular user. Using a pandas dataFrame in Line 51, parse a list of the row item, so that we can easily sort the data based on values in Rating column in descending order. Retrieve the first three data using pandas function .head(3) and the last three data using .tail(3). Save each result as a pandas dataFrame and convert the result to a saved csv file in Q1.

My best substitute user is 301, the movies he hates I hate and the movies he enjoys I find them interesting too

Q2

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

Answer

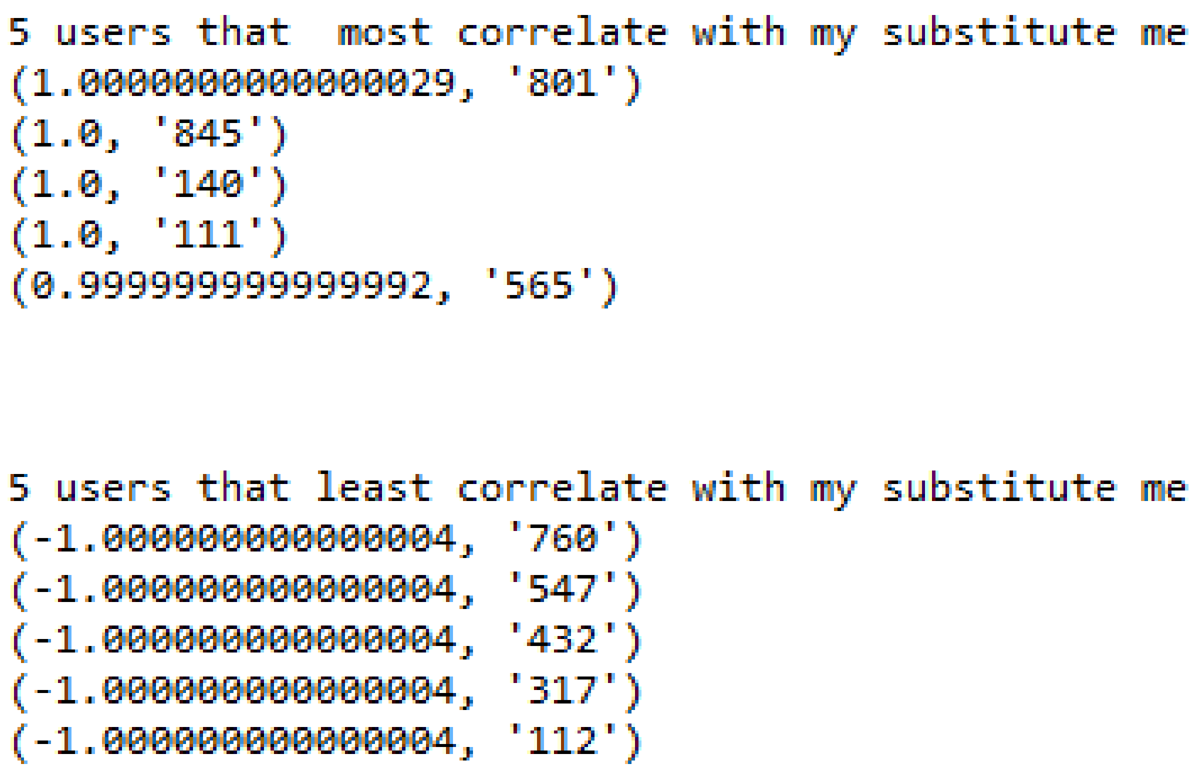
```python
# -*- coding: utf-8 -*-
"""
Created on Thu Apr  8 00:46:12 2021

@author: aadeniran
"""
from math import sqrt

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 getRecommendations(prefs, person, similarity=sim_pearson):
    """
     Gets recommendations for a person by using a weighted average
     of every other users rankings
    """
    totals = {}
    simSums = {}
    for other in prefs:
        # Dont 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 havent 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

def topMatches(prefs,person,n=5,similarity=sim_pearson,):
    '''
    Returns the best matches for person from the prefs dictionary.
```

```python
    Number of results and similarity function are optional params. bottomMatches
    '''

    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 lowest 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[len(scores)-n: len(scores)]

def transformPrefs(prefs):
    '''
    Transform the recommendations into a mapping where persons are described
    with interest scores for a given title e.g. {title: person} instead of
    {person: title}.
    '''

    result = {}
    for person in prefs:
        for item in prefs[person]:
            result.setdefault(item, {})
            # Flip item and person
            result[item][person] = prefs[person][item]
    return result

def loadMovieLens():
    # Get movie titles
    movies = {}
    for line in open("movie/u.item"):
        (id, title) = line.split("|")[0:2]
        movies[id] = title
    # Load data
    #prefs = {}
    for line in open("movie/u.data"):
        (user, movieid, rating, ts) = line.split("\t")
        prefs.setdefault(user, {})
        prefs[user][movies[movieid]] = float(rating)
    #Load in users
    #users = []
    for line in open("movie/u.user"):
        (user, age, gender, occupation, zipcode) = line.split("|")
        dictA = {"user": user, "age":age, "gender": gender, "occupation
  " :occupation }
    users.append(dictA)


prefs ={}
users =[]
loadMovieLens()

top5 = topMatches(prefs, "301", n = 5, similarity = sim_pearson)
bottom5 = bottomMatches(prefs, "928", n = 5, similarity = sim_pearson)
"""
Question 2
(1.0000000000000029, '801')
(1.0, '845')
(1.0, '140')
(1.0, '111')
(0.999999999999992, '565')


(-1.000000000000004, '760')
(-1.000000000000004, '547')
(-1.000000000000004, '432')
(-1.000000000000004, '317')
(-1.000000000000004, '112')
"""

print("5 users that  most correlate with my substitute me ")
print(*top5, sep='\n')
print("\n\n")
print("5 users that least correlate with my substitute me ")
print(*bottom5, sep='\n')

"""
Question 3
(5.0, 'Unforgettable (1996)')
(5.0, 'Net, The (1995)')
(5.0, 'Murder in the First (1995)')
```

```
(5.0, 'Murder at 1600 (1997)')
(5.0, 'Just Cause (1995)')


(2.0, 'Diabolique (1996)')
(2.0, 'Devil in a Blue Dress (1995)')
(2.0, 'Breakdown (1997)')
(1.0, 'Thinner (1996)')
(1.0, 'Albino Alligator (1996)')
"""
recommend = getRecommendations(prefs,"301")
print("\nTop 5 movies recommendations for substitute me: ")
print(*recommend[0:5], sep="\n")
print("\nBottom 5 movies recommendations for substitute me:")
print(*recommend[len(recommend)-5: len(recommend)], sep="\n")


"""
Question 4

[(1.000000000000004, 'Young Guns II (1990)'), (1.000000000000004, '
    Reality Bites (1994)'), (1.000000000000001, 'Ran (1985)'),
    (1.000000000000001, 'Butch Cassidy and the Sundance Kid (1969)'),
    (1.0000000000000009, 'Chinatown (1974)')]



[(-1.0, 'Big Night (1996)'), (-1.0, 'Barbarella (1968)'), (-1.0, '
    Another Stakeout (1993)'), (-1.0000000000000007, "Fathers' Day
    (1997)"), (-1.000000000000004, "Devil's Own, The (1997)")]

"""
print("\n\n")
movies = transformPrefs(prefs)
mybestmovie = "Vampire in Brooklyn (1995)"
topFive = topMatches(movies, mybestmovie)

print("My best 5 recommended movies")
print(topFive)

print("My worst 5 recommended movies")
worstFive = bottomMatches(movies,mybestmovie)
print("\n")
print(worstFive)
```

Listing 2: question2.py



```
5 users that  most correlate with my substitute me
(1.0000000000000029, '801')
(1.0, '845')
(1.0, '140')
(1.0, '111')
(0.9999999999999992, '565')



5 users that least correlate with my substitute me
(-1.000000000000004, '760')
(-1.000000000000004, '547')
(-1.000000000000004, '432')
(-1.000000000000004, '317')
(-1.000000000000004, '112')
```

Figure 1: The resulting out put for Question 2

Discussion

The Final output for the top 5 users that were most correlated to my substitute me were 801,845,140,111, and 565 with their respective correlating score of 1.0000000000000029,1.0,1.0,1.0 and 0.9999999999999992.

For the bottom 5 users that were most correlated to my substitute me were, 760,547,432,317,and 112. Their correlative score was the same, -1.000000000000004.

For the driver lines 139 to 165 generated the resulting output. First I loaded the u.items, u.data and u.user in this function and stored them in a dictionary, dictionary and list variables respectively. I called the topMatches function which I parsed in my substitute me user id in.

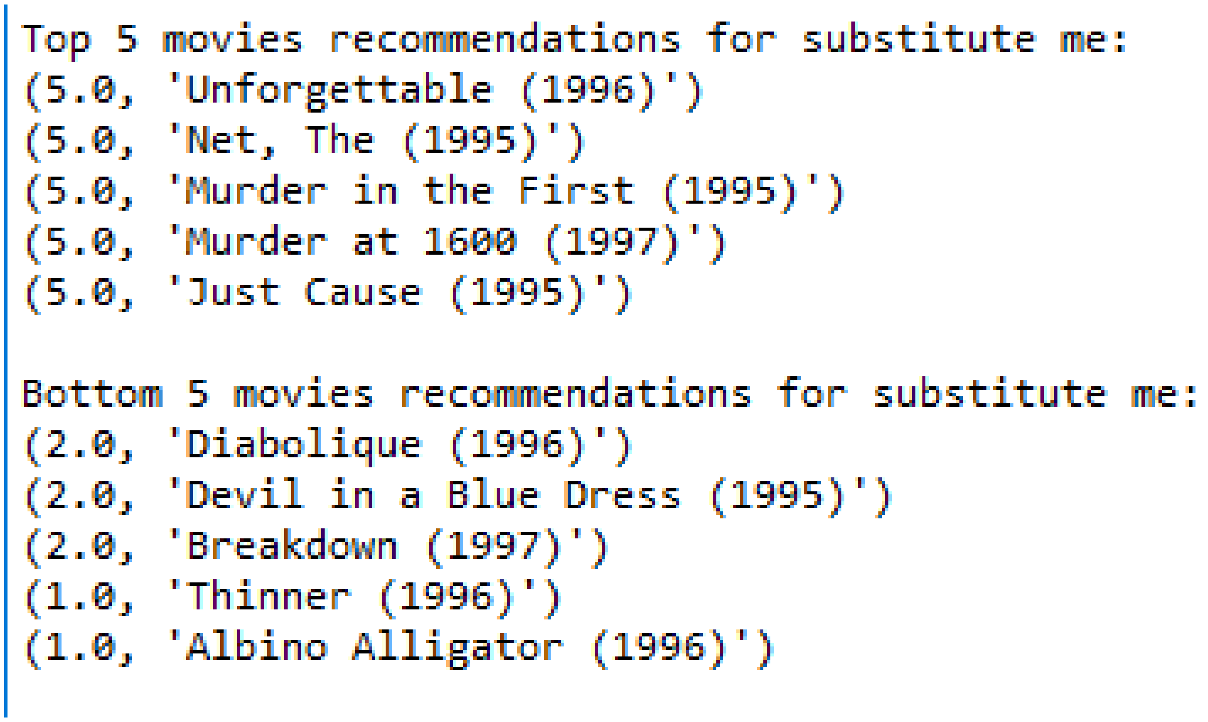
Q3

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).

Answer

Reference question2.py in Listing 2



```
Top 5 movies recommendations for substitute me:
(5.0, 'Unforgettable (1996)')
(5.0, 'Net, The (1995)')
(5.0, 'Murder in the First (1995)')
(5.0, 'Murder at 1600 (1997)')
(5.0, 'Just Cause (1995)')

Bottom 5 movies recommendations for substitute me:
(2.0, 'Diabolique (1996)')
(2.0, 'Devil in a Blue Dress (1995)')
(2.0, 'Breakdown (1997)')
(1.0, 'Thinner (1996)')
(1.0, 'Albino Alligator (1996)')
```

Figure 2: The resulting output for Question 3

Discussion

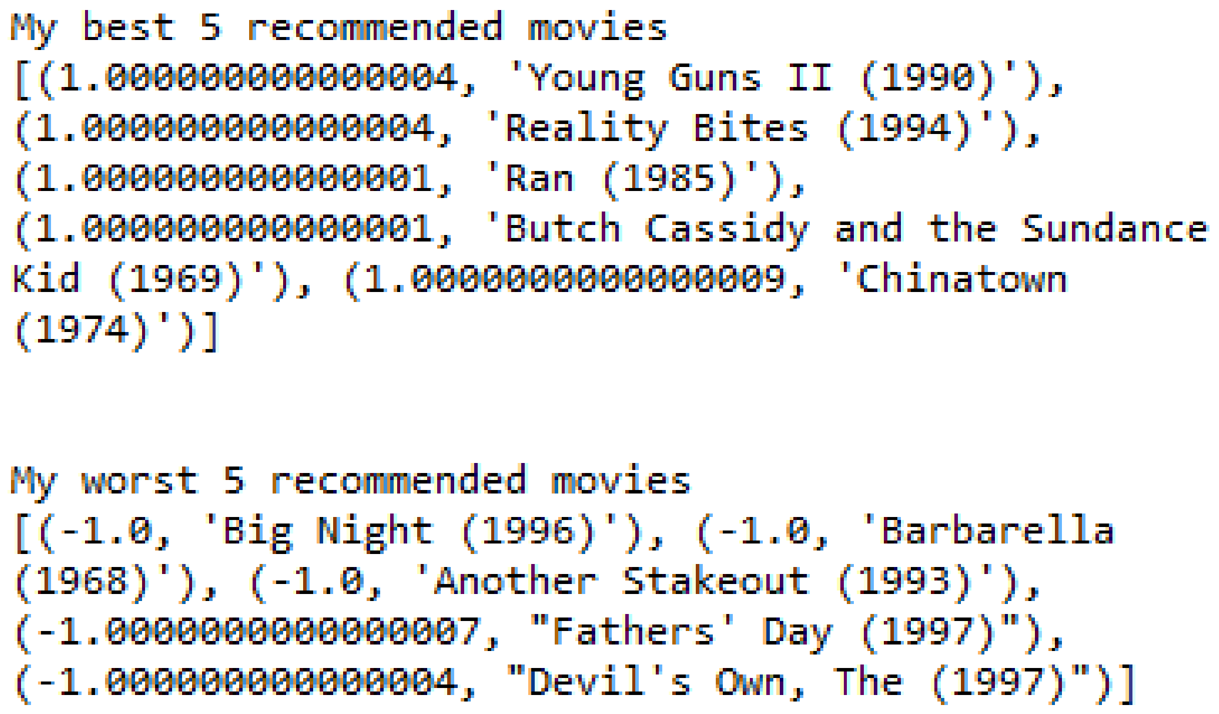
*In line 182 to 186, function getRecommendations(prefs,"301") was to get my substiture user recommended list, I Used *recommend[0:5] to get first 5 recommendation. This code was gotten from* `https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py`
Top 5 movies recommendations for substitute me, Unforgettable (1996), Net The (1995), Murder in the First (1995), Murder at 1600 (1997), Just Cause (1995)
*Bottom 5 movies recommendations for substitute me,in line 186 *recommend[len(recommend)-5: len(recommend)] used it to extract the bottom 5 movie recommendations for substitute me Diabolique (1996), Devil in a Blue Dress (1995), Breakdown (1997), Thinner (1996), Albino Alligator (1996)*

Q4

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

Reference question2.py in Listing 2



```
My best 5 recommended movies
[(1.0000000000000004, 'Young Guns II (1990)'),
(1.0000000000000004, 'Reality Bites (1994)'),
(1.0000000000000001, 'Ran (1985)'),
(1.0000000000000001, 'Butch Cassidy and the Sundance
Kid (1969)'), (1.0000000000000009, 'Chinatown
(1974)')]


My worst 5 recommended movies
[(-1.0, 'Big Night (1996)'), (-1.0, 'Barbarella
(1968)'), (-1.0, 'Another Stakeout (1993)'),
(-1.0000000000000007, "Fathers' Day (1997)"),
(-1.0000000000000004, "Devil's Own, The (1997)")]
```

Figure 3: The resulting output for Question 4

Discussion

I first transformed the prefs dictionary to movies. I then selected my best (Vampire in Brooklyn (1995)) movie from the whole list in u.data. The function topMatches and bottomMatch handled the result of the questions 3

Top 5 movies recommend from (Vampire in Brooklyn (1995) Young Guns II (1990) `https://www.youtube.com/watch?v=r-FmfxLy7fo` *I like the movie, a good old movie to me.*

Reality Bites (1994) `https://www.youtube.com/watch?v=xDYGo0UgIVM` *It seem interesting and I would watch it. Just an average movie for me.*
Ran (1985) `https://www.youtube.com/watch?v=YwP_kXyd-Rw` *I love this one also, war action my favorite.*
Butch Cassidy and the Sundance Kid (1969) `https://www.youtube.com/watch?v=YdJW2UxvSFQ`
I do love cow boys like movies
Chinatown (1974) `https://www.youtube.com/watch?v=20FfiP7g4tU` *Quite boring but I would enjoy the detective part to movie*

Worst 5 movies recommended from (Vampire in Brooklyn (1995)

Big Night (1996) `https://www.youtube.com/watch?v=Yd8gK6EgpLM` *looks boring to me because its a chef movie*
Barbarella (1968) `https://www.youtube.com/watch?v=M-fJg08wBKw` *A complete no to me because its too old and i dont find it interesting at all*
Another Stakeout (1993) `https://www.youtube.com/watch?v=Gpm4lGyOVYc` *I like this one, action, detective kind of movies*
Fathers' Day(1997) `https://www.youtube.com/watch?v=xsQfKt08Xlk` *This kind of movies does not capture my interest at all. Too boring.*
Devil's Own, The (1997)`Devil\OT1\textquoterightsOwn,The(1997)` *What I cant believe this is a least favourite movie. I completely love the action in this movie just the first second of seeing it. As usual war movies gets me all the time.]*

References

- `https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py`
- `https://www.example.com/reallyreallyreally-extra-long-URI/`