ECE-514 Recommender Systems

Nikos Angelopoulos 2110 - Dimitrios Dallas 2412 - Giorgos Kletsas 2408

January 2020

1 Introduction

1.1 The significance of recommender systems

Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user. Primarily they are directed towards individuals who lack sufficient personal experience and knowledge on a specific matter. In their simplest form, personalized recommendations are offered as ranked lists of items and they try to predict what the most suitable products or services are, based on the user's preferences and constraints. For instance, given the likes of a specific user, he can get a better recommendation for a future choice.

1.2 Our data

Our chosen dataset, which is provided by the following link: MovieLens dataset, consists of users, their ids, movies, their genres and ratings. These can be combined so that we can build our recommendation model.

1.3 The goal

The goal of this project is to construct a Recommender System that works well and provides useful and targeted suggestions for the users.

2 Our work

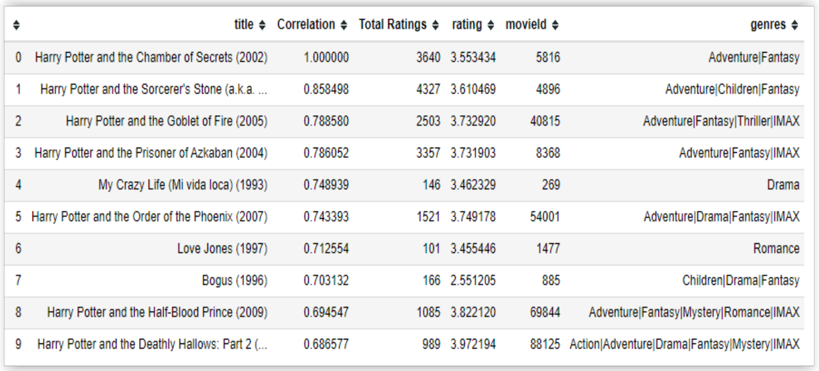
2.1 Getting/cleaning the data

Using the pandas and numpy libraries, we extracted the dataset into two dataframes and cleaned them from unnecessary entries. Then we merged them to a bigger dataframe with specific columns (**userId-movieId-rating-timestamp-title-genres**)

2.2 Recommender Algorithms

2.2.1 Correlation-based Algorithm

Using the corrwith method of the Pandas Dataframe, we computed the pairwise correlation between a specific movie entry and the whole dataframe. We provided two examples(Toy Story (1995) and Harry Potter and the Chamber of Secrets (2002)). Then we picked the movies with the highest correlation that had a more than 100 ratings and recommended them to the users.

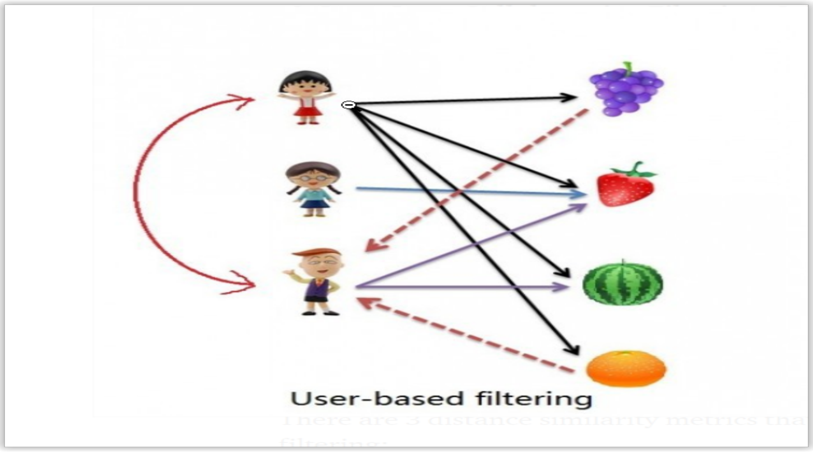


⇕	title ⇕	Correlation ⇕	Total Ratings ⇕	rating ⇕	movieId ⇕	genres ⇕
0	Harry Potter and the Chamber of Secrets (2002)	1.000000	3640	3.553434	5816	Adventure|Fantasy
1	Harry Potter and the Sorcerer's Stone (a.k.a. ...	0.858498	4327	3.610469	4896	Adventure|Children|Fantasy
2	Harry Potter and the Goblet of Fire (2005)	0.788580	2503	3.732920	40815	Adventure|Fantasy|Thriller|IMAX
3	Harry Potter and the Prisoner of Azkaban (2004)	0.786052	3357	3.731903	8368	Adventure|Fantasy|IMAX
4	My Crazy Life (Mi vida loca) (1993)	0.748939	146	3.462329	269	Drama
5	Harry Potter and the Order of the Phoenix (2007)	0.743393	1521	3.749178	54001	Adventure|Drama|Fantasy|IMAX
6	Love Jones (1997)	0.712554	101	3.455446	1477	Romance
7	Bogus (1996)	0.703132	166	2.551205	885	Children|Drama|Fantasy
8	Harry Potter and the Half-Blood Prince (2009)	0.694547	1085	3.822120	69844	Adventure|Fantasy|Mystery|Romance|IMAX
9	Harry Potter and the Deathly Hallows: Part 2 (...	0.686577	989	3.972194	88125	Action|Adventure|Drama|Fantasy|Mystery|IMAX

As we see from the example the highest correlation of the movie is with itself and the recommended are its sequels

Correlation-Algorithm explanation

- 3 people have seen Toy Story 1.
- All of them have rated it high. The 2 of them have seen Toy Story 2 and rated it high too.
- This algorithm recommends Toy Story 2 to the other person.



2.2.2 Content-based filtering Algorithm

The Content-Based Recommender relies on the similarity of the items being recommended. The basic idea is that if you like an item, then you will also like a "similar" item. We used Term Frequency(TF)/Inverse Document Frequency(IDF), specifically TfidfVectorizer that is used to determine the relative importance between movies. TF is simply the frequency of a word in a dataframe. IDF is the inverse of the word frequency among the whole dataframe. Using this method we recommended movies to the users based on the genres of the movies they liked(gave high rating).

Below is the equation to calculate the TF-IDF score:

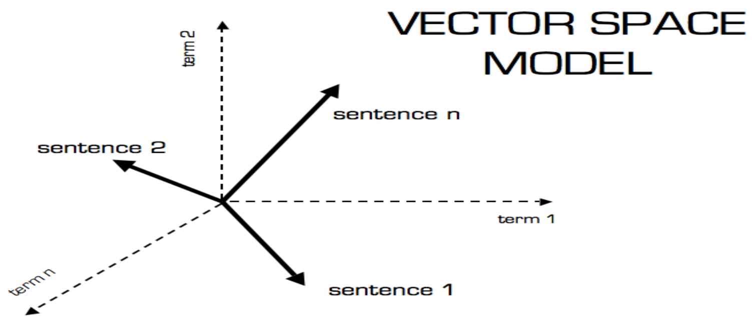
$$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \log\left(\frac{\mathrm{N}}{\mathrm{df}_i}\right)$$

$\mathrm{tf}_{i,j}$ = total number of occurences of i in j

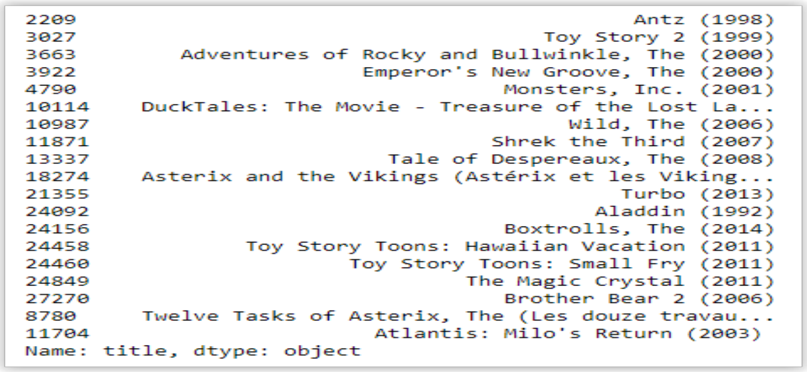
df_i = total number of documents (speeches) containing i

N = total number of documents (speeches)

Using the Vector Space Model which computes the proximity based on the angle between the vectors, we can determine the similarity between the movies. We used Cosine Similarities found by linear kernel from measuring the angle between the genres and the movies.



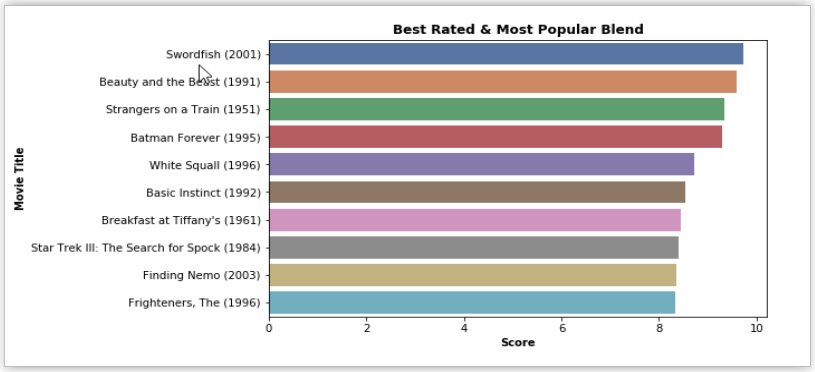
Then we created the recommendation function that sorts the similarity scores and returns the top 20 movies that are more similar to the selected movie by genre.



```
2209                                              Antz (1998)
3027                                       Toy Story 2 (1999)
3663            Adventures of Rocky and Bullwinkle, The (2000)
3922                          Emperor's New Groove, The (2000)
4790                                Monsters, Inc. (2001)
10114    DuckTales: The Movie - Treasure of the Lost La...
10987                                       Wild, The (2006)
11871                                 Shrek the Third (2007)
13337                          Tale of Despereaux, The (2008)
18274    Asterix and the Vikings (Astérix et les Viking...
21355                                           Turbo (2013)
24092                                         Aladdin (1992)
24156                                  Boxtrolls, The (2014)
24458                 Toy Story Toons: Hawaiian Vacation (2011)
24460                         Toy Story Toons: Small Fry (2011)
24849                               The Magic Crystal (2011)
27270                                  Brother Bear 2 (2006)
8780     Twelve Tasks of Asterix, The (Les douze travau...
11704                          Atlantis: Milo's Return (2003)
Name: title, dtype: object
```

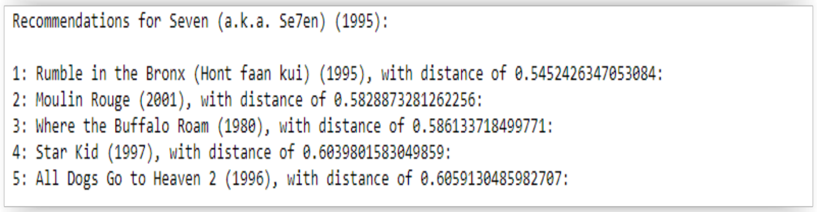
2.2.3 Popularity Algorithm

Given the total votes and the ratings of the users, we created a function that gives the average popularity of a movie. Specifically, we used a scaler for the two columns so that we can compare them better. Finally we recommended the top 20 movies.



2.2.4 kNN Algorithm

kNN is a machine learning algorithm to find clusters of similar users based on common movie ratings, and make predictions using the average rating of top-k nearest neighbors.
We use unsupervised algorithms with sklearn.neighbors. The algorithm we use to compute the nearest neighbors is “brute”, and we specify “metric=cosine” so that the algorithm will calculate the cosine similarity between rating vectors. Finally, we fit the model.
The kNN algorithm measures distance to determine the “closeness” of instances. It then classifies an instance by finding its nearest neighbors, and picks the most popular class among the neighbors.
We chose to find the closest five neighbors of the movie Se7en (1995) and calculated their distances.



```
Recommendations for Seven (a.k.a. Se7en) (1995):

1: Rumble in the Bronx (Hont faan kui) (1995), with distance of 0.5452426347053084:
2: Moulin Rouge (2001), with distance of 0.5828873281262256:
3: Where the Buffalo Roam (1980), with distance of 0.586133718499771:
4: Star Kid (1997), with distance of 0.6039801583049859:
5: All Dogs Go to Heaven 2 (1996), with distance of 0.6059130485982707:
```

References

[1] Jianrui Chen, Chunxia Zhao, Uliji, Lifang Chen. Collaborative filtering recommendation algorithm based on user correlation and evolutionary clustering
[2] Hui Li, Fei Cai, Zhifang Liao. Content-Based Filtering Recommendation Algorithm Using HMM
[3] Bei-Bei Cui. Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm