CSC 4740 / 6740 – Data Mining Project Report

Movie Recommender System

Team Members:

Faraz Mirza (002425477)

Atharva Vaidya (002410767)

- **Motivation**

We often see on Netflix that each user gets different movies and TV shows on their homepage. This provides each user with personalized content based on the movies and the TV shows they have watched. We plan to implement this logic so that we can predict which user should be given what recommendations.

An example of recommendation system is such as this:

- User A watches **Game of Thrones** and **Breaking Bad**.
- User B does search on Game of Thrones, then the system suggests Breaking Bad from data collected about user A.

We think this is essential because it becomes very convenient for users with the content being readily available as per their historical use, genres and the users with similar taste of movies.

Recommendation systems are used not only for movies, but on multiple other products and services like Amazon (Books, Items), Pandora/Spotify (Music), Google (News, Search), YouTube (Videos) etc.

- **Dataset**

  The dataset we are using is MovieLens dataset which is one of the most common dataset available for the data of movies and their genres and the user ratings of those movies. We have two files in this dataset:

  - Movies.csv

    This dataset contains the data of movies and their respective genre or genres. This dataset has more than 27,000 records which means genre information for more than 27,000 different movies.

    The features in this dataset are:

    i. MovieID
    ii. Title
    iii. Genre

    Below is the screenshot of the data:

A	B	C
movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller
11	American President, The (1995)	Comedy|Drama|Romance

- Ratings.csv

This dataset contains the data of the ratings given by users to the movies in the movie dataset. This dataset contains more than 20 million records which means the ratings by around 138,000 users. We decided to use the largest available dataset to make sure the recommendation done by our program is more accurate.

The features in this dataset are:

i. UserID
ii. MovieID
iii. Rating
iv. TimeStamp

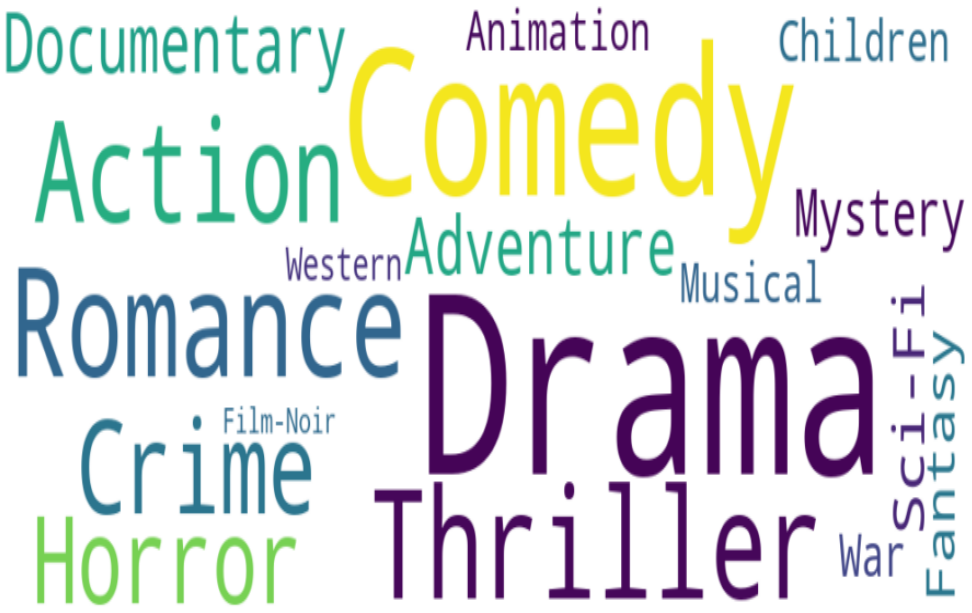
Below is the screenshot of the dataset:

A	B	C	D
userId	movieId	rating	timestamp
1	2	3.5	1112486027
1	29	3.5	1112484676
1	32	3.5	1112484819
1	47	3.5	1112484727
1	50	3.5	1112484580
1	112	3.5	1094785740
1	151	4	1094785734
1	223	4	1112485573
1	253	4	1112484940
1	260	4	1112484826

- **More about Datasets**

  After processing the data and doing some exploratory analysis, here are the most interesting features of this dataset:

  Word-cloud of movie genres:



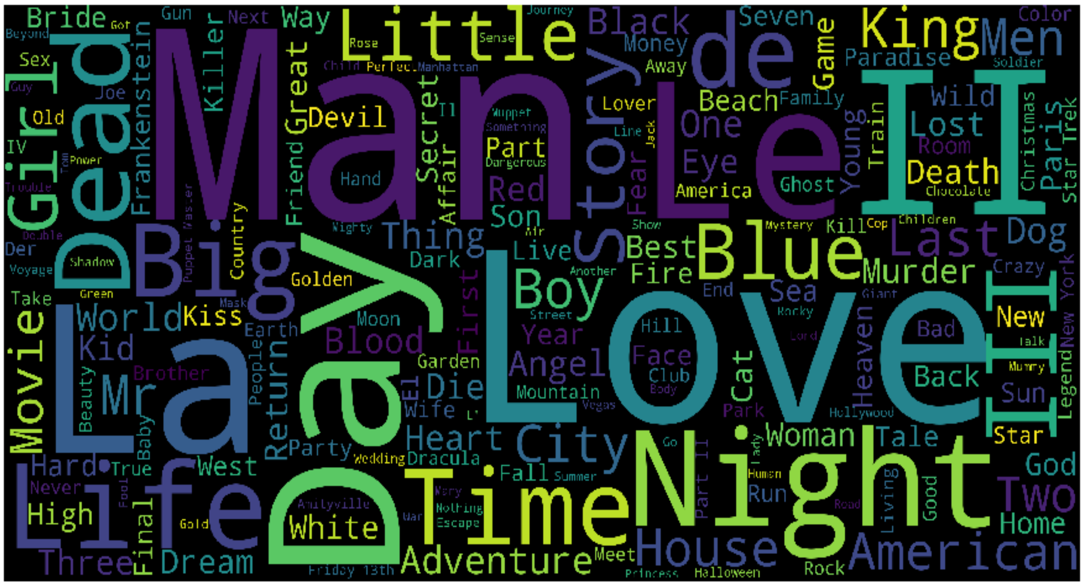
  Below are the most popular movie genres:

  - Drama – 13,344
  - Comedy – 8,374
  - Thriller – 4,178
  - Romance – 4,127
  - Action – 3,520

  Distribution of movie ratings:

As we can observe, almost half the movies have rating of 4 and 5 and the average rating is 3.58 out of 5. Each user rated at least 20 movies.

Word-cloud of movie names:



We can observe that a lot of movie names in the dataset have words like II or III. Also, most commonly occurring words in the movie names are Day, Love, Life, Time and so on.

- <u>**Methods**</u>

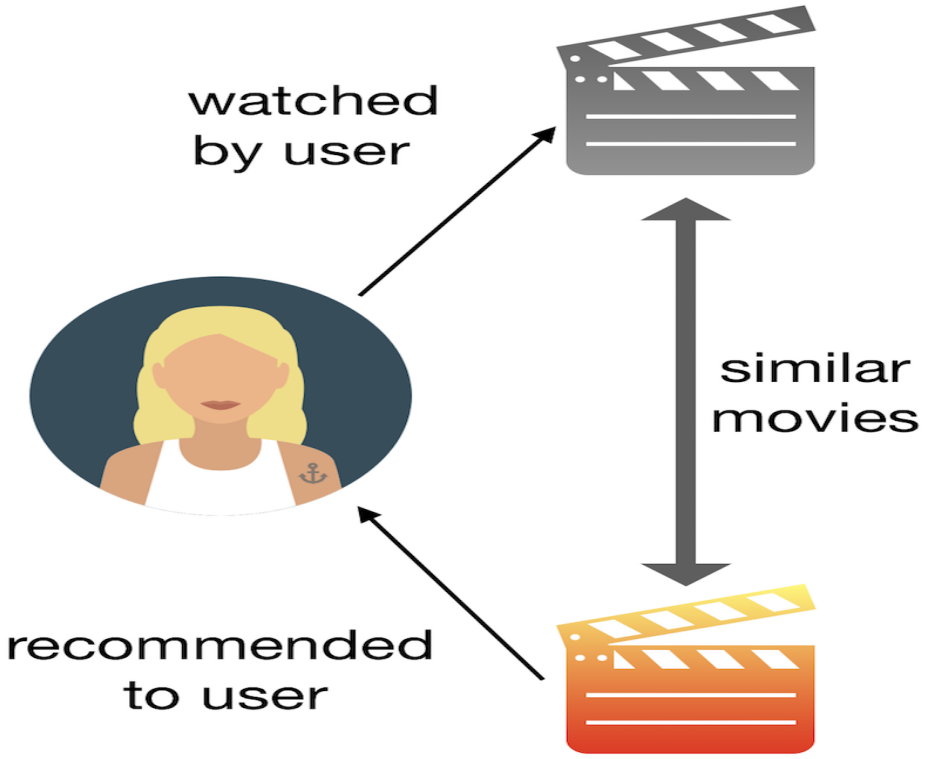
We have used two different methodologies to build this recommendation system.

1. **Content-Based Filtering**

   Content-based filtering works with the similarity of the items being recommended. The logic behind this is that if you like an item, then you will also like a “similar” item. In terms of movies, if a person watched a crime genre movie, the possibility is high that he/she will like other crime genre movies.

We have executed this logic by using the 'Genre' of the movie as the feature to determine the similarity. Based on the data given in the Movie dataset, we try to figure out user's liking and make a suggestion to the user.

Below is the visual description of this method:



We use the concepts of Term Frequency (TF) and Inverse Document Frequency (IDF) in our logic.

- TF = (# of times word X appears in a document) / (Total # of words in the document).
- IDF: If a word appears in many docs (like "the", "and", "of"), the word is not meaningful, so lower its score.

After calculating TF-IDF scores, we use Vector Space Model to compute the proximity based on the angle between the vectors. This is used to determine the similarity between the vectors which are genre(s) in our case.
Then we calculate the cosine similarity to find most similar movies for any of the movies in the dataset.

Below is the snapshot of the result for 'The Godfather'. The code gives the output of 10 movies in the dataset that are most similar to The Godfather with above logic.

```
In [8]: genre_recommendations('The Godfather').head(10)
```

Out[8]:

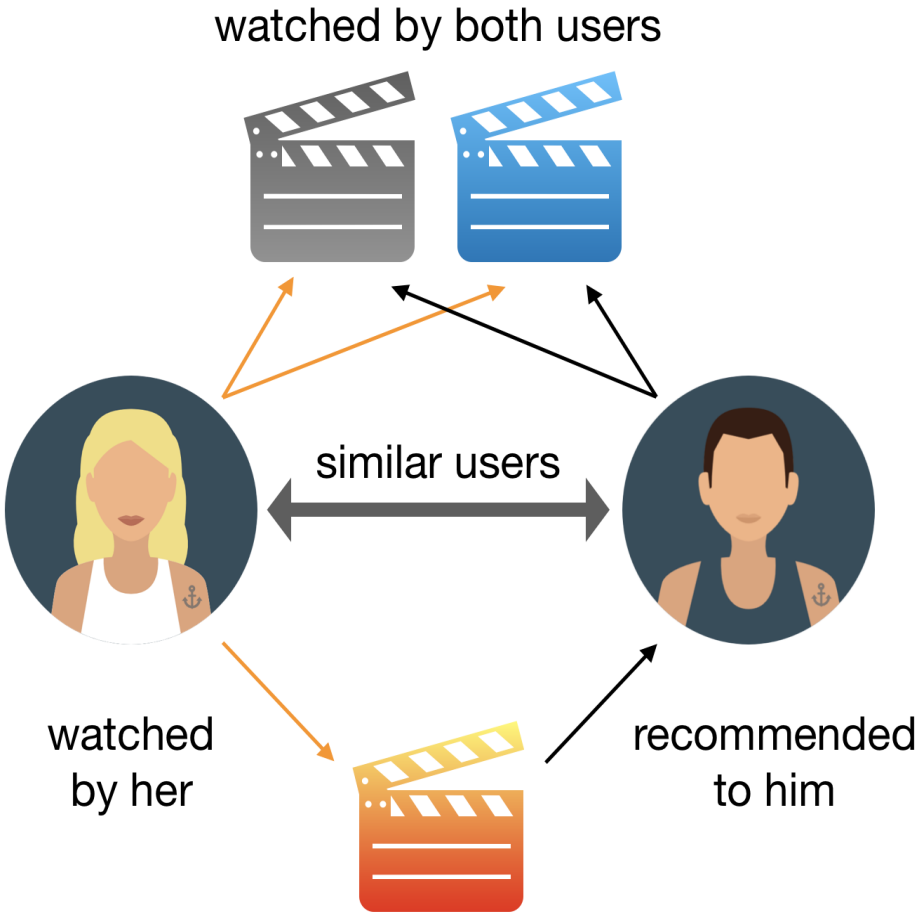
	title
29	Shanghai Triad (Yao a yao yao dao waipo qiao)
35	Dead Man Walking
95	Hate (Haine, La)
115	The Young Poisoner's Handbook
242	The Glass Shield
244	Heavenly Creatures
265	Little Odessa
280	New Jersey Drive
287	Once Were Warriors
315	The Shawshank Redemption

2. Collaborative Filtering

In contrast with content-based filtering, collaborative filtering recommender is based on the similarity in preferences of two or more users. It analyses how similar the preferences of one user is to another and makes recommendations on the basis of that.

We have used Ratings dataset to implement this logic. We set a threshold of at least 2500 reviews for the given movie. We did this so that only those movies will be considered for recommendation who have enough data about other user's preference. This will help to enhance the recommendation.

Below is the visual description of this method:



We have used 'User-User Collaborative Filtering' to implement out logic. We find look-alike based on similarity and recommend movies. For example, if user 1 watched movies A, B and C and user 2 watched movies B, C, D then there's a high possibility that user 1 will like D and user 2 will like A.

We build a similarity matrix which consists of a distance metrics that measure the similarity between two pairs of users.

```
In [12]: movie_matrix = small_data.pivot_table(index='userId', columns='title', values='rating')
         movie_matrix.head(10)
```

Out[12]:

title	(500) Days of Summer	10 Things I Hate About You	101 Dalmatians	101 Dalmatians (One Hundred and One Dalmatians)	12 Angry Men	13 Going on 30	1984 (Nineteen Eighty-Four)	2 Days in the Valley	20,000 Leagues Under the Sea	2001: A Space Odyssey	...	Young Guns II	Young Sherlock Holmes	Zodiac	Zombieland	Zoolander
userId																
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.5	...	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0	...	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	5.0	...	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.0	...	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN

Above sparse matrix has all the movies on the horizontal axis and all the users on vertical axis. Whenever a user queries a movie name, a vector of that movie containing all the user ratings will be generated and a pearson correlation coefficient between that vector and each movie column in that sparse matrix will be computed. Nan values will be ignored and Movies with the highest correlation will be returned.

Below is the snapshot of the result for ‘The Matrix. The code gives the output of 10 movies in the dataset that are most similar to The Matrix with above logic. It provides the correlation score as well.

```
In [14]: get_recommendations('The Matrix')
```

Out[14]:

title	correlation	average_rating
The Matrix	1.000000	4.187186
The Matrix Reloaded	0.519426	3.296389
The Matrix Revolutions	0.450520	3.143747
The Animatrix	0.360588	3.682432
Blade	0.332966	3.355627
Terminator 2: Judgment Day	0.330362	3.931954
Minority Report	0.327980	3.664897
The Lord of the Rings: The Two Towers	0.326290	4.107521
The Lord of the Rings: The Fellowship of the Ring	0.323902	4.137925
Mission: Impossible	0.319353	3.388599

- **Conclusion:**

  We covered two most popular data mining techniques for recommendation engines, Content based and Collaborative filtering. While these two are the quite popular, intuitive and at some extent work very well, there are many limitations to them. The context-based filtering does not take user experiences in account while the collaborative filtering needs to have a good amount of historic data and heavy computations to make a recommendation. The recommendation engines used by companies like Amazon and Netflix are much more complex these days but these are good as a starting point.

- **References:**

  - www.towardsdatascience.com
  - https://medium.com/@james_aka_yale/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223