Movie Recommendation System

Project Report

ITCS 5156 - APPLIED MACHINE LEARNING

Submitted by

Anoosh Guddehithlu Prathap Kumar

Submitted to

Minwoo “Jake” Lee

Primary Paper: “Personalize Movie Recommendation System”
[Authors: Shujia Liang, Lily Liu, Tianyi Liu
Year: December 4, 2018
Publication: Personalize Movie Recommendation System CS 229 Project Final Writeup]

Github link: https://github.com/Anooshgp/ITCS-5156.git

INTRODUCTION

Based on the person's prior movie ratings, we deploy machine learning model to implement a movie recommendation system. Coming to movies each and every person has his/her own interests which depends on genres, cast, languages etc. but this is not expressed in a single score, when movies are surfed in Google or any other website. Our system enables people to quickly pick the movies that appeal to them, irrespective of how diverse their interests/preferences are.

Content-based Filtering and Collaborative Filtering are two categories of recommendation systems. Features of items are employed in content-based filtering recommendations to propose additional products that are similar to what a person like based on their prior activities or comments. It works by gathering data from the user, either expressly (ratings) or unintentionally (data) (clicking on a link). User-based collaborative filtering and item-based collaborative filtering are the two types of collaborative filtering. User-based filtering is used to find people who are similar. Users with comparable interests and tastes in movies, literature, and other media are referred to as "similar users." Item-based filtering, on the other hand, looks for objects that are comparable. As collaborative filtering uses user movie ratings as input, KNN and matrix factoriaztion, it is the better method than content based in terms of prediction error and computation time.

Problem Statement

Movie Recommendation System recommends movies similar to the movie user likes and reviews given by the user for that movie. The capacity to quickly discover one's favorite film amid a large number of choices has been a key issue with the expansion of information. When a customer does not have a certain film in mind, a tailored recommendation system might be quite helpful. We are using filtering method, Cosine Similarity , and KNN utilizing cosine distance metric to achieve the task.

We are using MovieLens dataset which has (ml-latest-small) describes 5-star rating and free-text tagging activity, It contains 100836 ratings and 3683 tag applications across 9742 movies.

Uniqueness about my project is that, Prior versions of MovieLens dataset consists of precomputed cross-folds or scripts to perform this task, but we don’t use either of these features in our dataset.

Motivation and Challenges

- On the Internet, the number of choices are huge, there is a need to filter, order and proficiently deliver related information in order to improve the problem of surplus information, which has created a problem to many Internet users.
- Recommendation System solve this problem by searching through huge volume of dynamically generated information to offer the users with modified content.
- Similarly when we talk about Entertainment, movies comes to our mind. There are thousands of movies in the Internet from decades and still counting. Users gets confused what to watch, so based on user rating, genres, preferences and etc. it is beneficial to build a Movie Recommendation System.

These points indicates the motivation wheres challenges include,

- The data set may contain vast number of movies, but not all, that too not in all languages. But to achieve that it takes large amount of datasets, processing power, computational tasks and cost.
- For collaborative filtering, it cannot handle fresh items and its difficult to include side features like queries or items.

Concise Summary of Approach

- This dataset (ml-latest-small) contains 5-star rating and free-text labeling activity from the movie recommendation service [MovieLens](http://movielens.org). Over 9742 movies, it has 100836 ratings and 3683 tag applications. Between March 29, 1996, and September 24, 2018, 610 people produced this data. On September 26, 2018, this dataset was created.
- The users were chosen at random. All of the individuals that were chosen had rated at least 20 films. There is no demographic information provided. An id is assigned to each user, and no additional information is supplied.
- The files 'links.csv','movies.csv', 'ratings.csv', and 'tags.csv' contain the information. Following are more details on the contents and use of each of these files.
- Use case: Works with any case (lower or upper), The word "the" is always ignored and we can put the year of the movie inside brackets like: (2018)
- Genres included: *Action *Adventure *Animation *Children's *Comedy *Crime * Documentary *Drama * Fantasy *Film-Noir *Horror *Musical *Mystery *Romance *Sci-Fi *Thriller *War *Western *(no genres listed) too.
- We are using filtering method, Cosine Similarity , and KNN utilizing cosine distance metric to achieve the task.

BACKGROUND/RELATED WORK

The following are some methods of common recommender system approaches:

1. Content-based filtering
2. Collaborative filtering
3. Hybrid filtering

Content-based filtering:

This method sorts the things according to the user's preferences. It returns a result depending on what the user has already rated.

The Vector Space Model is the method used to model this approach (VSM). It deduces the item's similarity from its description and introduces the TF-IDF (Term Frequency-Inverse Document Frequency) idea.

Cosine Similarity: The angle of cosine between two things is measured by cosine similarity between two objects. It uses a normalized scale to compare two texts. It is possible to do so by calculating the dot product of the two identities.

Collaborative filtering solves some of the problems associated with content-based filtering by recommending items that comparable users enjoy and avoiding the need to gather data on each item by using the underlying structure of users' preferences. The neighborhood model and latent factor models are the two main techniques to collaborative filtering. The neighborhood model suggests the things that are nearest to the user's top rated items. Matrix factorization is an example of a latent factor model.

The unexplored territory of movie and user features We alter this approach by using a non-linear kernel and a variety of various parameters.

Schemes should be updated and their performance evaluated. Time dependency may also be incorporated into matrix factorization in order to track how consumers' preferences vary over time.

Many attempts have been made to combine the two methodologies, resulting in a number of hybrid approaches. Incorporating content information in collaborative filtering, where the content of a rated item is utilized to estimate the user's preference for other things, is one such strategy. This is especially useful when users only rated a tiny percentage of the total population of things, because the information of the unrated objects may be deduced through content-based filtering rather than having only a missing value. Other hybrid strategies involve generating a unified probabilistic mode or combining predictions from both methods in a linear fashion.

Literature Survey and results from some of the papers:

1) **Title**: Movie Recommendation System using Cosine Similarity and KNN

Authors: Ramni Harbir Singh, Sargam Maurya, Tanisha Tripathi, Tushar Narula, Gaurav Srivastav.

Year: June 2020

Publication: International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 – 8958 (Online), Volume-9 Issue-5, June 2020

This paper describes an approach which offers generalized recommendations to every user, based on movie popularity and/or genre. Content-Based Recommender System is implemented using various deep learning approaches. This paper also gives an insight into problems which are faced in content-based recommendation system.

Result:

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

Fig: Movies with different genres are takes and their similarity is taken with the help of cosine similarity

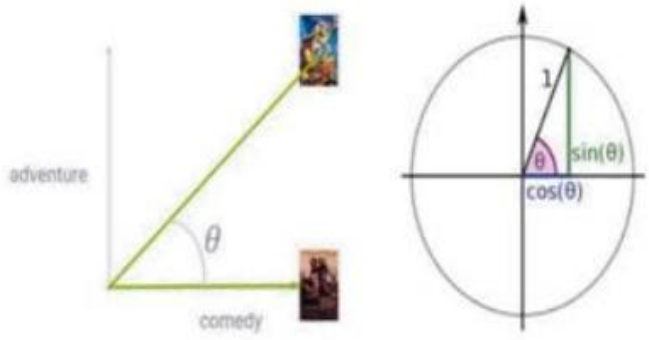

Fig: Cosine similarity

2) **Title**: Comprehensive Movie Recommendation System

Authors: Hrisav Bhowmick, Ananda Chatterjee, Jaydip Sen.

Year: 23 Dec 2021, **Publication:** arXiv:2112.12463 **[cs.IR]**

A recommender system, also known as a recommendation system, is a type of information filtering system that attempts to forecast a user's rating or preference for an item.Movie Recommendation System prototype based on the Genre, Pearson Correlation Coefficient, Cosine Similarity, KNN-Based, Content-Based Filtering using TFIDF and SVD, Collaborative Filtering using TFIDF and SVD, Surprise Library based recommendation system technology.

Results:

A. Genre Based Recommendation

Table I.

Title	Genres	Score
Fight Club (1999)	Action Crime Drama Thriller	4.241499
Star Wars: Episode IV - A New Hope (1977)	Action-Adventure Sci-Fi	4.204796
Dark Knight, The (2008)	Action Crime Drama IMAX	4.194470
Princess Bride, The (1987)	Action-Adventure Comedy Fantasy Romance	4.186827
Star Wars: Episode V The Empire Strikes Back	Action-Adventure Sci-Fi	4.185033

B. Pearson Correlation Coefficient Based Recommendation

Table II.

Title	Score
Gossip (2000)	1.0
White Man's Burden (1995)	1.0
Rapid Fire (1992)	1.0
Imaginary Heroes (2004)	1.0
Killer Elite (2011)	1.0

C. Cosine Similarity-Based Recommendation

Table III.

Title	Score
Leaving Las Vegas (1995)	1.0
Persuasion (1995)	1.0
How to Make an American Quilt (1995)	1.0
Bed of Roses (1996)	1.0
Angels and Insects (1995)	1.0

D. KNN algorithm (with Cosine distance metric) Based Recommendations

Table IV

Title	Score
Guyver: Dark Hero (1994)	0.3035
Harrison Bergeron (1995)	0.3638
Real Life (1979)	0.3824
The Punisher: Dirty Laundry (2012)	0.3860
House of Cards (1993)	0.3874

3) **Title**: Personalize Movie Recommendation System

Authors: Shujia Liang, Lily Liu, Tianyi Liu

Year: December 4, 2018

Publication: Personalize Movie Recommendation System CS 229 Project Final Writeup

Current recommender systems generally fall into two categories: content-based filtering and collaborative filtering. The paper explains both approaches. Content-based filtering, uses movie features such as actors, directors, movie description, and keywords as inputs and TF-IDF and doc2vec to calculate the similarity between movies. Collaborative filtering uses 'observed users' movie rating as input and also use K-nearest neighbors and matrix factorization to predict user's movie ratings. When comparing collaborative filtering seems to perform better than content-based filtering in terms of prediction error and computation time.

Results:

Movie Title	Avg Votes	Num Votes	Weighted Score
The Shawshank Redemption	8.5	8358.0	8.445871
The Godfather	8.5	6024.0	8.425442
Dilwale Dulhania Le Jayenge	9.1	661.0	8.421477
The Dark Knight	8.3	12269.0	8.265479
Fight Club	8.3	9678.0	8.256387
Pulp Fiction	8.3	8670.0	8.251408
Schindler's List	8.3	4436.0	8.206643
Whiplash	8.3	4376.0	8.205408
Spirited Away	8.3	3968.0	8.196059
Life Is Beautiful	8.3	3643.0	8.187177

Table 1: Top 10 most popular movies by weighted score

Similar Movies	Sim Score
Octopussy	0.8358
Transporter	0.8298
Safe house	0.8287
Unlocked	0.8106
Undercover Man	0.8062
Push	0.8033
Sniper 2	0.8007
Patriot Games	0.8005
2047 Sights Death	0.7952
Interceptor	0.7951

Table 2: Movies most similar to Skyfall

Learning Rate	RMSE
0.020	0.927403
0.025	0.927003
0.030	0.927009
0.035	0.927358
0.040	0.927994
0.05	0.929483
0.1	0.938572
0.2	0.950744
0.4	0.970215

Table 3: Learning rate with test RMSE

Some Disadvantages:

Cold Start: The difficulty with the cold start is that when a person registers for the first time, he hasn't seen any movies. As a result, the recommendation algorithm is unable to provide results based on any movie. This is known as the cold-start issue. As a result of the lack of a previous record, the recommendation system encounters this issue. This occurs once for each new user.

Data Sparsity: The data sparsity problem occurs when a user has only rated a few products, making it difficult for the recommendation engine to provide correct results. The findings provided in this challenge are not very close to the desired outcome. When the system fails to produce satisfactory results, it generates poor suggestions. Furthermore, data scarcity inevitably leads to coverage issues.

Scalability: This is another issue that comes up when it comes to recommendations. In this case, item encoding is done in a linear fashion. When the data set is constrained in size, the system performs well. As the data set grows, the recommendation algorithm struggles to provide reliable recommendations based on different movie genres.

Relation to Our Approach:

As I am choosing user rating on movies as the input it falls under Collaborative filtering. Research study says it is more accurate than content based filtering.

Prior versions of MovieLens dataset consists of precomputed cross-folds or scripts to perform this task, but we don’t use either of these features in our dataset.

METHOD

Dataset and Architecture:

User IDs: MovieLens users were chosen at random to be included in the study. Their identifiers have been removed. The user ids in 'ratings.csv' and 'tags.csv' are the same (i.e., the same id refers to the same user across the two files).

Movie IDs: The collection only includes movies with at least one rating or tag. These movie ids are the same as those on the MovieLens website (id '1' corresponds to the URL, for example). 'ratings.csv', 'tags.csv','movies.csv', and 'links.csv' all have the same movie ids (i.e., the same id refers to the same movie across these four data files).

Data File Structure for Ratings (ratings.csv):

The file 'ratings.csv' contains all of the ratings. After the header row, each line in this file represents one user's rating of one movie, and has the following format: userId,movieId,rating,timestamp

The lines in this file are sorted by userId first, then by movieId inside each user. Ratings are given on a scale of one to five stars, with half-star increments (0.5 stars - 5.0 stars). The timestamps reflect seconds since midnight on January 1, 1970, in Coordinated Universal Time (UTC).

Data File Structure Tags (tags.csv):

The file 'tags.csv' contains all of the tags. After the header row, each line of this file represents one tag added to one video by one user, and follows this format: userId,movieId,tag,timestamp
The lines in this file are sorted by userId first, then by movieId inside each user. User-generated metadata about movies is known as tags. A single word or a brief phrase is usually used for each tag. Each user determines the meaning, value, and purpose of a given tag.The timestamps reflect seconds since midnight on January 1, 1970, in Coordinated Universal Time (UTC).

Movies Data File Structure(movies.csv):

The file'movies.csv' contains movie information. After the header row, each line of this file represents one movie and has the following format: movieId,title,genres
Movie titles are entered manually or imported from <https://www.themoviedb.org/>, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.

Links Data File Structure (links.csv):

The file 'links.csv' contains identifiers that may be used to link to various sources of movie

data. After the header row, each line of this file represents one movie and has the following format: movieId,imdbId,tmdbId. https://movielens.org> uses movieId as an identification for movies. The movie Toy Story, for example, has the URL https://movielens.org/movies/1.

http://www.imdb.com> uses an identification for movies called imdbId. The movie Toy Story, for example, has the URL http://www.imdb.com/title/tt0114709/.

https://www.themoviedb.org> uses tmdbId as a movie identification. The movie Toy Story, for example, has the URL https://www.themoviedb.org/movie/862.

The terms of each provider apply to the use of the resources mentioned above.

Steps Involved:

- Importing packages and inspecting datasets:

Start by inspecting our dataset

In [2]:
```
links_df = pd.read_csv('data/links.csv')
links_df.head()
```

Out[2]:

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

In [3]:
```
movies_df = pd.read_csv('data/movies.csv')
movies_df.head()
```

Out[3]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [4]:
```
ratings_df = pd.read_csv('data/ratings.csv')
ratings_df.head()
```

Out[4]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [5]:
```
tags_df = pd.read_csv('data/tags.csv')
tags_df.head()
```

Out[5]:

	userId	movieId	tag	timestamp
0	2	60756	funny	1445714994
1	2	60756	Highly quotable	1445714996
2	2	60756	will ferrell	1445714992
3	2	89774	Boxing story	1445715207
4	2	89774	MMA	1445715200

The Algorithm works like this, If the user likes a movie (let the movie be M_i), then the algorithm finds other users who liked the same movie (call these users U).

After that, the algorithm finds the most liked movies by each U user (call them M_j), sorts them by rating and the last (highest rated) 5 movies are selected for recommendation.

Let's go on the assumption that if person A likes movies $M_0, M_1, \ldots, M_i$ then there goes person B who likes one or more movies from M_i let's call them M_j.

This would mean that A and B has a movie that they both liked, therefore other movies from both M_j and M_i can be liked by both A and B with high probability.

In [6]:
```
df = movies_df.merge(ratings_df, on='movieId')
```

In [7]:
```
M_j = 'John Wick (2014)' # Title as input, now it's just one movie
recommended_movies = []

# Find the movie in the database, and sort it by rating
movie_db = df[df['title'] == M_j]\
           .sort_values(by='rating', ascending=False)

# Get the first 5 users who liked this movie
for user in movie_db.iloc[:5]['userId'].values:

    # Get the rated movies for this user
    rated_movies = df[df['userId'] == user]

    # Get the five biggest rated movie by this user
    rated_movies = rated_movies[rated_movies['title'] != M_j]\
                   .sort_values(by='rating', ascending=False)\
                   .iloc[:5]

    # Add these to the recommendations
    recommended_movies.extend(list(rated_movies['title'].values))

recommended_movies = np.unique(recommended_movies)

for movie in recommended_movies:
    print(movie)
```

- Check for the movie name in the database and Exclude "the": The word ‘the’ is ignored for user convenience.
- Find the most similars by name. Do not add movie to the list if it has already been added
- Choose the biggest pointed five.
- Ask user for input and find the movies in the database and sort it by rating.
- Get the first 5 users who liked this movie and get rated movies for this user.
- Get the 5 biggest rated movie by this user: If we type “Avengers:”,

```
C:\Users\Anoosh Gp\OneDrive\Desktop\Movie Recommendation System>python .\rmovie.py "Avengers:"

Movie with title "Avengers:", can't be found in the "data/movies.csv" file.
These are the most similar movie titles:

-----------------------------
Index   Movie title
[1]:    Avengers: Age of Ultron (2015)
[2]:    Avengers: Infinity War - Part I (2018)
[3]:    Toy Story (1995)
[4]:    Jumanji (1995)
[5]:    Grumpier Old Men (1995)
-----------------------------
```

- Add these to the recommendations, weight each movie and sort them based on score

```
Movie index: 2

------------------------------
Movie recommendations:

[1]:    Avengers: Infinity War - Part I (2018)
[2]:    Star Wars: Episode V - The Empire Strikes Back (1980)
[3]:    Incredibles, The (2004)
[4]:    Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
[5]:    Indiana Jones and the Temple of Doom (1984)
[6]:    Deadpool 2 (2018)
[7]:    Good, the Bad and the Ugly, The (Buono, il brutto, il cattivo, Il) (1966)
[8]:    Willow (1988)
[9]:    WALL·E (2008)
[10]:   Inglourious Basterds (2009)
[11]:   Arrival (2016)
[12]:   Lord of the Rings: The Fellowship of the Ring, The (2001)
[13]:   Toy Story (1995)
[14]:   Dark Knight, The (2008)
[15]:   Little Miss Sunshine (2006)
[16]:   Paddington 2 (2017)
[17]:   Three Billboards Outside Ebbing, Missouri (2017)
[18]:   Planet Earth II (2016)
[19]:   Pan's Labyrinth (Laberinto del fauno, El) (2006)
[20]:   Wonder (2017)
[21]:   Pursuit of Happyness, The (2006)
[22]:   Ratatouille (2007)
[23]:   Planet Earth (2006)
[24]:   Addams Family, The (1991)
------------------------------
```

EXPERIMENTS

Explanation of experimental setup:

We have used Jupyter notebook to perform Data Preprocessing, Exploratory Data Analysis, Model Training and Model Evaluation. There are different libraries used for this experiment: Pandas, Numpy, Sklearn, Argparse etc. There are two files included, ipynb file can run on notebook and py file explains the implementation that runs on command prompt.

Test Results:

1)

C:\Movie Recommendation System >python .\rmovie.py **"Star Wars"**

These are the most similar movie titles:

Index Movie title

[1]: Star Wars: The Clone Wars (2008)

[2]: The Star Wars Holiday Special (1978)

[3]: Rogue One: A Star Wars Story (2016)

[4]: Solo: A Star Wars Story (2018)

[5]: Star Wars: Episode IV - A New Hope (1977)

Please choose one by providing the index for the movie that is most likely what you search for. (Type 0 and enter for exit)

Movie index: 5

Movie recommendations:

[1]: Star Wars: Episode V - The Empire Strikes Back (1980)

[2]: Princess Bride, The (1987)

[3]: Indiana Jones and the Last Crusade (1989)

[4]: From Russia with Love (1963)

[5]: Outlaw Josey Wales, The (1976)

[6]: Close Encounters of the Third Kind (1977)

[7]: Ghost in the Shell (Kôkaku kidôtai) (1995)

[8]: Monty Python and the Holy Grail (1975)

[9]: Battle Royale (Batoru rowaiaru) (2000)

[10]: Excalibur (1981)

[11]: Donnie Darko (2001)

[12]: No Man's Land (2001)

[13]: Suspiria (1977)

[14]: Borat: Cultural Learnings of America for Make Benefit Glorious Nation of Kazakhstan (2006)

[15]: Snatch (2000)

[16]: M*A*S*H (a.k.a. MASH) (1970)

[17]: Pink Floyd: The Wall (1982)

[18]: Fargo (1996)

[19]: Philadelphia Story, The (1940)

[20]: Pursuit of Happyness, The (2006)

[21]: Schindler's List (1993)

[22]: Enemy at the Gates (2001)

[23]: 12 Angry Men (1957)

[24]: Koyaanisqatsi (a.k.a. Koyaanisqatsi: Life Out of Balance) (1983)

---

2)

C:\Movie Recommendation System>python .\rmovie.py "Lord of the rings"

These are the most similar movie titles:

Index Movie title

[1]: Lord of the Rings: The Fellowship of the Ring, The (2001)

[2]: Lord of the Rings: The Return of the King, The (2003)

[3]: Legend of 1900, The (a.k.a. The Legend of the Pianist on the Ocean) (Leggenda del pianista sull'oceano) (1998)

[4]: Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)

[5]: Fall of the House of Usher, The (House of Usher) (1960)

Please choose one by providing the index for the movie that is most likely what you search for. (Type 0 and enter for exit)

Movie index: 2

Movie recommendations:

[1]: Harry Potter and the Deathly Hallows: Part 2 (2011)

[2]: Pirates of the Caribbean: Dead Man's Chest (2006)

[3]: Gladiator (2000)

[4]: Taken 2 (2012)

[5]: Toy Story (1995)

[6]: Illusionist, The (2006)

[7]: Lord of the Rings: The Fellowship of the Ring, The (2001)

[8]: Hobbit: The Desolation of Smaug, The (2013)

[9]: Girl with the Dragon Tattoo, The (2011)

[10]: Godfather: Part II, The (1974)

[11]: English Vinglish (2012)

[12]: Bourne Ultimatum, The (2007)

[13]: Shawshank Redemption, The (1994)

[14]: Blue Velvet (1986)

[15]: John Wick (2014)

[16]: In Bruges (2008)

[17]: Beautiful Mind, A (2001)

[18]: Contact (1997)

[19]: In the Name of the Father (1993)

[20]: Now You See Me (2013)

[21]: Office Space (1999)

---

3) Now we have a different scenario, user can put the year of release beside name of the movie. If we want a recommendation with high imdb ratings similar to movie "Batman: Gotham Knight(2008)", this directly gives recommendation, we don't have to choose 1 in 5 movies that we want unlike the outputs shown above, as we are assured with the movie title.

C:> python .\rmovie.py "Batman: Gotham Knight (2008)"

Movie recommendations:

[1]: Scarface (1983)

[2]: Inception (2010)

[3]: Fight Club (1999)

[4]: Rare Exports: A Christmas Tale (Rare Exports) (2010)

[5]: Willow (1988)

[6]: Kill Bill: Vol. 2 (2004)

[7]: Inglourious Basterds (2009)

[8]: Toy Story (1995)

[9]: Indiana Jones and the Temple of Doom (1984)

[10]: WALL·E (2008)

[11]: Forrest Gump (1994)

[12]: Addams Family, The (1991)

[13]: M*A*S*H (a.k.a. MASH) (1970)

[14]: Dr. Horrible's Sing-Along Blog (2008)

---

4)

C:Movie Recommendation System>python .\rmovie.py "Interstellar (2014)"

Movie recommendations:

[1]: Arrival (2016)

[2]: 300 (2007)

[3]: Dark Knight, The (2008)

[4]: Deadpool 2 (2018)

[5]: Vanilla Sky (2001)

[6]: Predator (1987)

[7]: Road to Perdition (2002)

[8]: The Revenant (2015)

[9]: Lord of the Rings: The Fellowship of the Ring, The (2001)

[10]: Amelie (Fabuleux destin d'Amélie Poulain, Le) (2001)

[11]: 40-Year-Old Virgin, The (2005)

[12]: Indiana Jones and the Temple of Doom (1984)

[13]: American Made (2017)

[14]: Assassination of Jesse James by the Coward Robert Ford, The (2007)

[15]: Paddington 2 (2017)

[16]: Planet Earth II (2016)

[17]: American History X (1998)

[18]: Planet Earth (2006)

[19]: It (2017)

[20]: Shining, The (1980)

[21]: Pineapple Express (2008)

[22]: Lock, Stock & Two Smoking Barrels (1998)

[23]: Toy Story (1995)

[24]: Beautiful Mind, A (2001)

---

5)

Another interesting case, if we mention a movie name plus year, but does not have a genre, even then it recomends movie similar to it.

C:>python .\rmovie.py "A Midsummer Night's Dream (2016)"

Movie recommendations:

[1]: How to Train Your Dragon 2 (2014)

[2]: Avatar (2009)

[3]: Star Wars: Episode I - The Phantom Menace (1999)

[4]: Star Wars: Episode II - Attack of the Clones (2002)

[5]: Guardians of the Galaxy (2014)

Deep analysis/discussion about the results:

So we have seen many intersting cases how the system is working. If we search for a movie we will get highest rated 5 movies, because the algorithm finds users who has rated that movie, it actually printing most liked movies by each user sorts them by rating and the last(highest rated) 5 movies are selected for recommendation.

One case we've seen that, "the" word can be ignored, it will work without any complications.

Another case, if we put year beside movie name, it works and directly recommends, we don't have to choose again.

Lastly movies similar to the movie without genre in the database is recommended too.

CONCLUSION

We have successfully shown the modeling of a movie recommendation system. The algorithm is more accurate than other distance measurements, and it has a lower level of complexity.

In the realm of the internet, recommendation systems have become the most important source of useful and trustworthy information. Simple ones take into account one or a few criteria, whilst more complicated ones employ several parameters to filter the results and make them more user-friendly. A good movie recommendation system may be constructed using sophisticated deep learning and various filtering approaches such as collaborative filtering and hybrid filtering. This might be a significant step forward in the development of this model, as it will not only become more efficient to use, but will also boost the commercial value.

Thoughts about the project:

What I have learnt:

I mainly learnt about the application, how Movie Recommendation plays a vital role in this era of internet. I've learnt about the MovieLens dataset, its structure with movie details.i.e. here we had ratings, tags, movies, links csv files in the dataset. I got good hands on experience on python and working better with data which improved our data analysing skills.

Challenges and Future Work:

The data set may contain vast number of movies, but not all, that too not in all languages. But to achieve that it takes large amount of datasets, processing power, computational tasks and cost. Secondly, this looks simple for now, but I would love to make the application web based using front end design and APIs in the future.

My Contributions:

- Prior versions of MovieLens dataset consists of precomputed cross-folds or scripts to perform this task, but we don't use either of these features in our dataset.
- I have enabled different use cases, like
  Word "the" can be ignored.
  Upper cases and lower cases doesn't matter.
  Can mention year beside movie name.

I referred the following github links to implement this project:

https://github.com/rudrajikadra/Movie-Recommendation-System-Using-Python-and-Pandas/blob/master/Movie%20Recommender%20System.ipynb

https://github.com/Chaitanyakaul97/movie-recommendation-system/blob/master/Movie%20recomendation%20system.ipynb

https://github.com/topics/movie-recommendation-system

REFERENCES

[1] A. Tuzhilin and G. Adomavicius. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge & Data Engineering, vol. 17, no.6, pp. 734-749, 2005.

[2] M. Muozorganero, G. A. Ramezgonzlez, P. J. Muozmerino, and C. D Kloos, "A collaborative recommender system based on space-time similarities," vol. 9, no. 3, pp. 81–87, 2010.

[3] G. Salton. Automatic Text Processing. Addison-Wesley (1989)

[4] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proceedings of the 10th international conference on World Wide Web. ACM, 2001, pp. 285–295.

[5] Caselles-Dupr´e, Hugo et al. Word2vec applied to recommendation: hyperparameters matter. RecSys. 2018.

[6] G. Wang, "Survey of personalized recommendation system," Computer Engineering & Applications, 2012.

[7] I. Soboroff and C. Nicholas. Combining Content and Collaboration in Text Filtering. Proc. Int'l Joint Conf. Artificial Intelligence Workshop: Machine Learning for Information Filtering, Aug. 1999.

[8] Robert Bell, Yehuda Koren, and Chris Volinsky. Modelling relationships at multiple scales to improve the accuracy of large recommender systems. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 95–104. ACM, 2007.

[7]. Suvir Bhargav. Efficient features for movie recommendation systems. 2014.

[8]. Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. Recommender systems: an introduction. Cambridge University Press, 2010.

[9] http://grouplens.org/datasets/ and http://movielens.org/

[10] https://doi.org/10.1145/2827872

[11] https://github.com/

[12] https://stackoverflow.com/