Movie Recommendation System

Khuong Viet Tai - 19522151

July 2023

0.1 Introduction

A Movie Recommendation System is a popular application in the fields of machine learning and artificial intelligence. This project focuses on researching and developing algorithms and models to provide personalized movie suggestions based on users' preferences and viewing behavior. The system's objective is to deliver accurate and relevant movie recommendations, enhancing user experience, increasing user engagement, and overall satisfaction when using online movie streaming services.

0.2 Rationale and Significance

With an ever-expanding array of movie options, users often encounter difficulties in finding and selecting films that match their personal tastes. A Movie Recommendation System addresses this issue by automatically proposing movies based on users' individual preferences and past viewing history. This saves time spent searching and ensures users have access to films they might be interested in.

Additionally, the Movie Recommendation System plays a vital role in driving the business activities of online movie platforms. By providing compelling suggestions, the system boosts user interaction, fosters user loyalty, and ultimately increases sales and revenue for content providers.

0.3 Key Research Elements

Data Collection and Processing: The project requires research on methods for gathering and processing movie and user data. It involves identifying basic movie information such as genres, directors, actors, release years, and user ratings. Effective methods for storing and managing this data will also be developed.

Data Analysis and Preprocessing: To better understand the data and recognize user preferences, the project will explore data analysis and preprocessing techniques. Techniques such as factor analysis, classification, and clustering may be employed to address this aspect.

Building Recommendation Models: The project focuses on researching and developing machine learning and artificial intelligence models to generate personalized movie recommendations for users. Methods like Collaborative Filtering, Content-Based Filtering, and Hybrid Filtering will be investigated and applied to improve recommendation quality.

Evaluation and Performance Metrics: Ensuring the effectiveness and accuracy of the recommendation system is crucial. Therefore, the project will design evaluation methods and performance metrics. Metrics such as accuracy, coverage, and user feedback will be employed to measure the system's performance.

0.4 Data Set

The first dataset contains the following features:

- `movie_id` - A unique identifier for each movie.
- `cast` - The names of lead and supporting actors.
- `crew` - The names of Director, Editor, Composer, Writer, etc.

The second dataset has the following features:

- `budget` - The budget in which the movie was made.
- `genre` - The genre of the movie, such as Action, Comedy, Thriller, etc.
- `homepage` - A link to the homepage of the movie.
- `id` - This is, in fact, the `movie_id` as in the first dataset.
- `keywords` - The keywords or tags related to the movie.
- `original_language` - The language in which the movie was made.
- `original_title` - The title of the movie before translation or adaptation.
- `overview` - A brief description of the movie.
- `popularity` - A numeric quantity specifying the movie's popularity.
- `production_companies` - The production house of the movie.
- `production_countries` - The country in which it was produced.
- `release_date` - The date on which it was released.
- `revenue` - The worldwide revenue generated by the movie.
- `runtime` - The running time of the movie in minutes.
- `status` - "Released" or "Rumored".
- `tagline` - The movie's tagline.
- `title` - The title of the movie.
- `vote_average` - The average ratings the movie received.
- `vote_count` - The count of votes received.

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	...	runtime	spoken_languages	status	tagline	title	vote_average	vote_count	title	cast	crew
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id...	en	Avatar	In the 22nd century, a paraplegic Marine is ...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	...	162.0	[{"iso_639_1": "en", "name": "English"}, {"iso...	Released	Enter the World of Pandora.	Avatar	7.2	11800	Avatar	[{"cast_id": 242, "character": "Jake Sully", ...	[{"credit_id": "52fe48009251416c750ac023", "de...
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, h...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"...	...	169.0	[{"iso_639_1": "en", "name": "English"}]	Released	At the end of the world, the adventure begins.	Pirates of the Caribbean: At World's End	6.9	4500	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"name": "Columbia Pictures", "id": 5}, {"nam...	...	148.0	[{"iso_639_1": "fr", "name": "Français"}, {"iso...	Released	A Plan No One Escapes	Spectre	6.3	4466	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950	[{"name": "Legendary Pictures", "id": 923}, {"...	...	165.0	[{"iso_639_1": "en", "name": "English"}]	Released	The Legend Ends	The Dark Knight Rises	7.6	9106	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	en	John Carter	John Carter is a war-weary, former military ca...	43.926995	[{"name": "Walt Disney Pictures", "id": 2}]	...	132.0	[{"iso_639_1": "en", "name": "English"}]	Released	Lost in our world, found in another.	John Carter	6.1	2124	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

Dataset info

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 26 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   index                 4803 non-null   int64
 1   budget                4803 non-null   int64
 2   genres                4803 non-null   object
 3   homepage              1712 non-null   object
 4   id                    4803 non-null   int64
 5   keywords              4803 non-null   object
 6   original_language     4803 non-null   object
 7   original_title        4803 non-null   object
 8   overview              4803 non-null   object
 9   popularity            4803 non-null   float64
 10  production_companies  4803 non-null   object
 11  production_countries  4803 non-null   object
 12  release_date          4802 non-null   object
 13  revenue               4803 non-null   int64
 14  runtime               4801 non-null   float64
 15  spoken_languages      4803 non-null   object
 16  status                4803 non-null   object
 17  tagline               3959 non-null   object
 18  title                 4803 non-null   object
 19  vote_average          4803 non-null   float64
 20  vote_count            4803 non-null   int64
 21  tittle                4803 non-null   object
 22  cast                  4803 non-null   object
 23  crew                  4803 non-null   object
 24  director              4803 non-null   object
 25  soup                  4803 non-null   object
dtypes: float64(3), int64(5), object(18)
memory usage: 975.7+ KB
```

0.5 Graphic Filtering

Before getting started with this:

1. We need a metric to score or rate movies.
2. Calculate the score for every movie.
3. Sort the scores and recommend the best-rated movie to the users.

We can use the average ratings of the movie as the score, but using this won't be fair enough since a movie with an 8.9 average rating and only 3 votes cannot be considered better than a movie with a 7.8 average rating but 40 votes. So, I'll be using IMDB's weighted rating (wr) which is given by the following formula:

$$weightedrating(wr) = \left(\frac{v}{v+m} \cdot R\right) + \left(\frac{m}{v+m} \cdot C\right) \tag{1}$$

where:

- v is the number of votes for the movie.
- m is the minimum number of votes required for the movie to be considered.
- R is the average rating of the movie.
- C is the mean vote across the whole report.

Using this weighted rating system helps in considering both the average rating and the number of votes, providing a fairer way to rank and recommend movies.

We see that there are 481 movies which qualify to be in this list. Now, we need to calculate our metric for each qualified movie. To do this, we will define a function, `weighted_rating()`, and define a new feature `score`, of which we'll calculate the value by applying this function to our DataFrame of qualified movies:

```
[20] def weighted_rating(x, m=m, C=C):
        v = x['vote_count']
        R = x['vote_average']
        # Calculation based on the IMDB formula
        return (v/(v+m) * R) + (m/(m+v) * C)
```

Finally, let's sort the DataFrame based on the score feature and output the title, vote count, vote average and weighted rating or score of the top 10 movies.

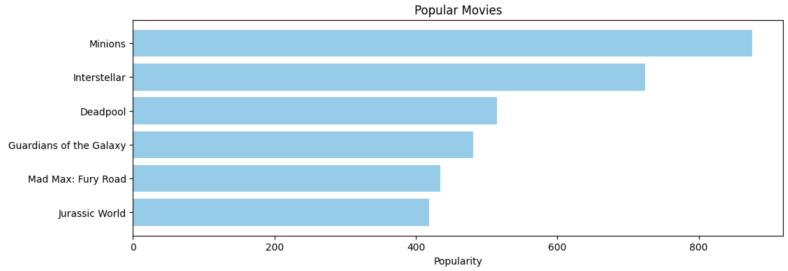
	title	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239
3337	The Godfather	5893	8.4	7.851236
95	Interstellar	10867	8.1	7.809479
809	Forrest Gump	7927	8.2	7.803188
329	The Lord of the Rings: The Return of the King	8064	8.1	7.727243
1990	The Empire Strikes Back	5879	8.2	7.697884

Trending Movie

```
pop= df2.sort_values('popularity', ascending=False)
import matplotlib.pyplot as plt
plt.figure(figsize=(12,4))

plt.barh(pop['title'].head(6),pop['popularity'].head(6), align='center',
        color='skyblue')
plt.gca().invert_yaxis()
plt.xlabel("Popularity")
plt.title("Popular Movies")
```

```
Text(0.5, 1.0, 'Popular Movies')
```



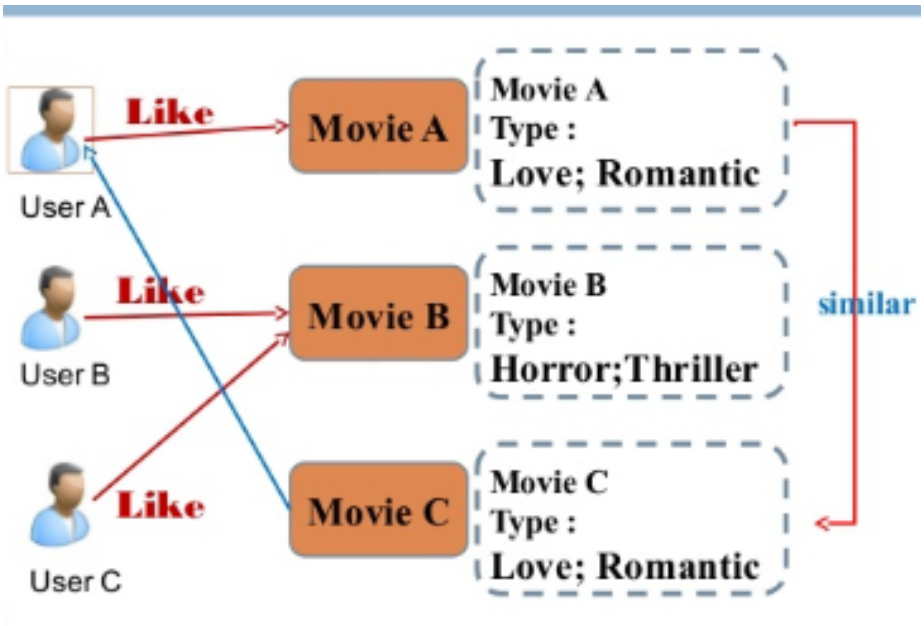
0.6 Content Based Filtering

In this recommender system, the content of the movie (overview, cast, crew, keyword, tagline, etc.) is used to find its similarity with other movies. Then, based on this similarity, the movies that are most likely to be similar are recommended to users.

The recommender system works as follows:

1. Collect relevant information about each movie, such as its overview, cast, crew, keywords, and tagline.
2. Represent each movie as a vector, where each element corresponds to a specific feature or aspect of the movie.
3. Use similarity metrics or algorithms, such as cosine similarity or collaborative filtering, to measure the similarity between pairs of movies based on their feature vectors.
4. Identify the movies that have the highest similarity scores with a given target movie.
5. Recommend the most similar movies to users based on their viewing history or preferences.

By leveraging the movie content and similarity measures, this recommender system can provide personalized movie recommendations to users, enhancing their movie-watching experience.



We will be using the cosine similarity to calculate a numeric quantity that denotes the similarity between two movies. The cosine similarity score is chosen because it is independent of magnitude and is relatively easy and fast to calculate. Mathematically, it is defined as follows:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Using the cosine similarity allows us to measure the similarity between movies in a way that is not affected by their individual magnitudes, making it a suitable choice for finding related movies in the recommender system.

We are going to define a function that takes in a movie title as an input and outputs a list of the 10 most similar movies. Firstly, for this, we need a reverse mapping of movie titles and DataFrame indices. In other words, we need a mechanism to identify the index of a movie in our metadata DataFrame, given its title.

```
#Import TfIdfVectorizer from scikit-learn
from sklearn.feature_extraction.text import TfidfVectorizer

#Define a TF-IDF Vectorizer Object. Remove all english stop words such as 'the', 'a'
tfidf = TfidfVectorizer(stop_words='english')

#Replace NaN with an empty string
df2['overview'] = df2['overview'].fillna('')

#Construct the required TF-IDF matrix by fitting and transforming the data
tfidf_matrix = tfidf.fit_transform(df2['overview'])

#Output the shape of tfidf_matrix
tfidf_matrix.shape
```

```
(4803, 20978)
```

We are now in a good position to define our recommendation function. These are the following steps we'll follow:

1. Get the index of the movie given its title.
2. Get the list of cosine similarity scores for that particular movie with all movies. Convert it into a list of tuples where the first element is its position, and the second is the similarity score.
3. Sort the aforementioned list of tuples based on the similarity scores; that is, the second element.
4. Get the top 10 elements of this list. Ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself).
5. Return the titles corresponding to the indices of the top elements.

By following these steps, the recommendation function will identify the top 10 movies that are most similar to a given movie title, thereby providing personalized movie recommendations to users based on movie similarity.

```python
# Function that takes in movie title as input and outputs most similar movies
def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return df2['title'].iloc[movie_indices]
```

```
[29] get_recommendations('The Dark Knight Rises')
```

```
65                            The Dark Knight
299                            Batman Forever
428                            Batman Returns
1359                                   Batman
3854    Batman: The Dark Knight Returns, Part 2
119                              Batman Begins
2507                                 Slow Burn
9           Batman v Superman: Dawn of Justice
1181                                       JFK
210                             Batman & Robin
Name: title, dtype: object
```

0.7 Collaborative Filtering

article [utf8]inputenc

Our content-based engine suffers from some severe limitations. It is only capable of suggesting movies which are close to a certain movie. That is, it is not capable of capturing tastes and providing recommendations across genres.

Also, the engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user. Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that movie, regardless of who she/he is.

Therefore, in this section, we will use a technique called Collaborative Filtering to make recommendations to Movie Watchers. Collaborative Filtering can be categorized into two types:

1. User-based filtering: These systems recommend products to a user that similar users have liked. For measuring the similarity between two users, we can either use Pearson correlation or cosine similarity. This filtering technique can be illustrated with an example. In the following matrices, each row represents a user, while the columns correspond to different movies except the

last one, which records the similarity between that user and the target user. Each cell represents the rating that the user gives to that movie. Assume user E is the target.

Although computing user-based CF is very simple, it suffers from several problems. One main issue is that users' preferences can change over time, which may lead to bad performance if we precompute the matrix based on their neighboring users.

2. Item-based Collaborative Filtering: Instead of measuring the similarity between users, the item-based CF recommends items based on their similarity with the items that the target user rated. Likewise, the similarity can be computed with Pearson Correlation or Cosine Similarity. The major difference is that, with item-based collaborative filtering, we fill in the blank vertically, as opposed to the horizontal manner that user-based CF does.

It successfully avoids the problem posed by dynamic user preference as item-based CF is more static. However, several problems remain for this method. First, the main issue is scalability. The computation grows with both the customer and the product, and the worst-case complexity is O(mn) with m users and n items. Additionally, sparsity is another concern. In extreme cases, we can have millions of users, and the similarity between two fairly different movies could be very high simply because they have a similar rank for the only user who ranked them both.

0.8 Conclusion

In conclusion, the "Movie Recommendation System" is a highly effective and valuable tool that enhances the movie-watching experience for users. Through advanced algorithms and data analysis, the system provides personalized movie suggestions based on individual preferences, past viewing history, and user feedback. This not only saves users time in searching for suitable movies but also exposes them to a broader range of films they might not have discovered otherwise.

The Movie Recommendation System's success lies in its ability to continuously learn and adapt to users' changing tastes, making the recommendations more accurate and relevant over time. Additionally, the system contributes to increased user engagement and satisfaction, as it fosters a deeper connection between viewers and the movie platform.

Moreover, this technology holds significant potential for businesses in the entertainment industry, as it can lead to increased user retention, higher conversion rates, and better customer loyalty. By tailoring the movie recommendations to each user, companies can create a more personalized and engaging experience, ultimately leading to a competitive advantage in the market.

However, it is essential to address certain challenges, such as privacy concerns and the risk of creating echo chambers, where users' preferences are reinforced rather than diversified. Striking a balance between personalization and serendipity is crucial to ensure that users are exposed to a variety of films that

align with their interests while still exploring new genres and styles.

As technology advances and more data becomes available, the Movie Recommendation System is poised to evolve further, incorporating cutting-edge AI techniques and expanding its capabilities. As a result, it has the potential to revolutionize how audiences discover and enjoy movies in the future.

Overall, the "Movie Recommendation System" is a significant advancement in the realm of movie consumption, revolutionizing the way users find and enjoy films. Its impact on the entertainment industry and the potential it holds for enhancing user experiences make it a promising and exciting field of research and development.