MOVIE RECOMMENDATION SYSTEM USING MACHINE LEARNING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

Name	**Univ. Roll No.**
Ayush Kumar Sharma	10800119010
Anurag Kumar	10800119048
Debjit Ray	10800119079
Debapriya Mukhopadhyay	10800119073

UNDER THE GUIDANCE OF

MRS. VEDATRAYEE CHATTERJEE
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ASANSOL ENGINEERING COLLEGE
AFFILIATED TO
MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

May, 2023

Contents

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ASANSOL ENGINEERING COLLEGE
Vivekananda Sarani, Kanyapur, Asansol, West Bengal – 713305

Certificate of Recommendation

I hereby recommend that the minor project report entitled, **"Movie Recommendation System Using Machine Learning"** carried out under my supervision by the group of students listed below may be accepted in partial fulfillment of the requirement for the degree of "Bachelor of Technology in **Computer Science and Engineering**" of Asansol Engineering College under MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY.

Name	**Univ. Roll No.**
Ayush Kumar Sharma	10800119010
Anurag Kumar	10800119048
Debjit Ray	10800119079
Debapriya Mukhopadhyay	10800119073

_____

(Mrs. Vedatrayee Chatterjee)
Thesis Supervisor
Dept. Of Computer Science and Engineering
Asansol Engineering College
Asansol-713305

Countersigned:

_____

(Dr. Monish Chatterjee)
Head of the Department
Dept. Of Computer Science and Engineering,
Asansol Engineering College,
Asansol-713305

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ASANSOL ENGINEERING COLLEGE
Vivekananda Sarani, Kanyapur, Asansol, West Bengal – 713305

Certificate of Approval

The forgoing minor project is hereby approved as creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it is submitted.

_____

(Mrs. Vedatrayee Chatterjee)
Thesis Supervisor
Dept. Of Computer Science and Engineering
Asansol Engineering College
Asansol-713305

Acknowledgement

It is our great privilege to express our profound and sincere gratitude to our Project Supervisor, **Mrs. Vedatrayee Chatterjee** for providing us a very cooperative and precious guidance at every stage of the present project work being carried out under his/her supervision. Her valuable advice and instructions in carrying out the present study has been a very rewarding and pleasurable experience that has greatly benefited us throughout the course of work.

We would like to convey our sincere gratitude towards **Dr. Monish Chatterjee**, Head of the Department of **Computer Science and Engineering**, Asansol Engineering College for providing us the requisite support for timely completion of our work. We would also like to pay our heartiest thanks and gratitude to all the teachers of the **Department of Computer Science and Engineering**, Asansol Engineering College for various suggestions being provided in attaining success in our work.

We would like to express our earnest thanks to **Mr. Suman Mallick** of CSE Project Lab for his technical assistance provided during our project work.

Finally, I would like to express my deep sense of gratitude to my parents for their constant motivation and support throughout my work.

……

(Ayush Kumar Sharma)

……
(Anurag Kumar)

……
(Debjit Ray)

……
(Debapriya Mukhopadhyay)

Abstract

Movie recommendation systems have become increasingly popular in recent years due to the abundance of available movie content and the challenge of finding personalized recommendations. This report explores the development of a movie recommendation system using machine learning techniques. The objective is to provide users with accurate and relevant movie suggestions based on their preferences and viewing history.

The report begins by discussing the importance of recommendation systems in the context of the movie industry and the benefits they offer to both users and movie providers. It then delves into the underlying machine learning algorithms used in the recommendation system, including collaborative filtering and content-based filtering. These techniques leverage user behaviour data and movie metadata to make personalized recommendations.

Furthermore, the report explores the data pre-processing steps required to prepare the movie dataset for training and evaluation. It covers methods such as data cleaning, feature extraction, and data normalization, which help improve the performance and accuracy of the recommendation system.

Overall, this report provides a comprehensive overview of a movie recommendation system using machine learning. It demonstrates the significance of such systems in the movie industry and offers insights into the underlying algorithms, data pre-processing techniques, evaluation methods, and challenges faced. The findings presented in this report contribute to the understanding and advancement of recommendation systems, ultimately enhancing the movie-watching experience for users.

List of Figures **Page no.**

List of Tables **Page no.**

1. Preface

1.1 Introduction

The basic concept behind a movie recommendation system is quite simple. In particular, there are two main elements in every recommender system: users and items.

The system generates movie predictions for its users, while items are the movies themselves. The primary goal of movie recommendation systems is to filter and predict only those movies that a corresponding user is most likely to want to watch. The ML algorithms for these recommendation systems use the data about this user from the system's database. This data is used to predict the future behaviour of the user concerned based on the information from the past.

The Movie recommendation system Is aiming to make searching process for movies a lot easier and people don't need to waste a lot of time in finding the movies they might want to watch

1. In comparison to the present system the proposed system will be less time consuming and is more efficient.

2. Searching for the best movies will be very easy in proposed system as it is automated.

3. Result will be very precise and accurate and will be declared in very short span of time because calculation and evaluations are done by the system itself.

1.2 Motivation of the project

The motivation behind developing a movie recommendation system using machine learning stems from the exponential growth of available movie content and the challenge of finding relevant movies in this vast landscape. With numerous streaming platforms and an ever-expanding catalogue of films, users often face decision fatigue and struggle to discover movies that align with their preferences and interests.

Personalized movie recommendations address this challenge by leveraging user data and machine learning algorithms to provide tailored suggestions. By analysing a user's viewing history, ratings, and preferences, a recommendation system can offer curated movie recommendations that align with their individual tastes, leading to a more satisfying and engaging movie-watching experience.

1.3 Basic description of the project

The primary objective of this project is to develop a movie recommendation system using machine learning algorithms. The system will leverage user data, such as viewing history, ratings, and preferences, to generate personalized movie recommendations.

The recommendation system will employ a combination of collaborative filtering and content-based filtering techniques. Collaborative filtering focuses on analysing user behaviour and preferences to identify patterns and similarities between users, allowing for the generation of recommendations based on the preferences of similar users. Content-based filtering, on the other hand, considers movie metadata, such as genre, director, and cast, to recommend movies that share similar characteristics.

2. Literature Review

2.1 General

In recent years, movie recommendation systems have received significant attention in both academia and industry. Researchers and industry practitioners have explored various approaches and techniques to develop effective movie recommendation systems. This section provides a general overview of the existing literature on movie recommendation systems and highlights key findings and trends in the field.

One of the widely used approaches in movie recommendation systems is collaborative filtering. Collaborative filtering leverages user behaviour data, such as ratings and viewing history, to identify similar users and recommend movies based on the preferences of those with similar tastes.

Content-based filtering is another popular approach in movie recommendation systems. This technique focuses on analysing movie metadata, such as genre, director, and cast, to recommend movies that share similar characteristics.

Hybrid approaches that combine collaborative filtering and content-based filtering techniques have also been proposed. These hybrid models aim to overcome the limitations of individual approaches by leveraging the complementary strengths of both methods. By combining user behaviour data and movie metadata, hybrid models can generate more accurate and diverse recommendations.

2.2 Review of related work

Several studies and research papers have contributed to the advancement of movie recommendation systems. These works have proposed novel algorithms, evaluated different evaluation metrics, and addressed specific challenges in the field. Here are some notable studies in the literature:

1. “Matrix Factorization Techniques for Recommender Systems” by Y. Koren et al. (2009): This influential paper introduced matrix factorization techniques for collaborative filtering in recommendation systems. It proposed models such as Singular Value Decomposition (SVD) and Alternating Least Squares (ALS) to improve the accuracy of recommendations.
2. “Factorization Machines” by S. Rendle (2010): This work introduced factorization machines, a powerful approach for collaborative filtering. Factorization machines combine factorization models with feature engineering, enabling the 2 odelling of complex interactions between user preferences and movie attributes.
3. “Content-based Movie Recommendations Based on Pairwise Learning to Rank” by B. Mobasher et al. (2007): This study focused on content-based filtering approaches and proposed pairwise learning-to-rank methods for movie recommendations. It demonstrated the effectiveness of considering pairwise preferences and utilizing ranking algorithms for generating personalized recommendations.
4. “Deep Neural Networks for YouTube Recommendations” by P. Covington et al. (2016): This paper presented the application of deep neural networks for personalized video recommendations on YouTube. It highlighted the benefits of using deep learning architectures, such as Deep Neural Networks (DNNs), in capturing complex patterns and generating accurate recommendations.

These seminal works provide valuable insights into the algorithms, techniques that serve as a foundation for the development of the proposed recommendation system in this project while offering directions for further research and improvement.

3. Related Theories and Algorithms

3.1 Fundamental theories underlying the work

In the development of a movie recommendation system, several fundamental theories form the basis of the algorithms and techniques employed. Understanding these theories is essential for building an effective recommendation system. Here are the key theories that underpin the work:

1. Collaborative Filtering: Collaborative filtering relies on the assumption that users with similar preferences in the past will have similar preferences in the future. This theory involves analysing user behaviour data, such as ratings and viewing history, to identify similarities between users and recommend items based on the preferences of similar users.

2. Content-based Filtering: Content-based filtering suggests that users' preferences can be inferred by analysing the attributes or content of items. In movie recommendation systems, this theory implies that movies with similar attributes, such as genre, director, and cast, are likely to be preferred by users who have enjoyed similar movies in the past.

3. Hybrid Filtering: Hybrid filtering combines collaborative filtering and content-based filtering approaches to leverage the strengths of both methods. It aims to provide more accurate and diverse recommendations by incorporating user behaviour data and movie attributes.

3.2 Fundamental algorithms

1. Singular Value Decomposition (SVD): SVD is a matrix factorization algorithm used to decompose the user-item rating matrix. It helps capture latent factors representing user preferences and movie attributes, ultimately minimizing the Root Mean Square Error (RMSE) and improving recommendation accuracy.

2. Count Vectorizer: Count vectorizer is a technique used in content-based filtering to convert textual movie attributes, such as genre or cast, into numerical feature vectors. It transforms the text data into a matrix representation suitable for computation.

3. Cosine Similarity: Cosine similarity is a measure used in both content-based filtering and collaborative filtering. It calculates the cosine of the angle between two vectors, representing the similarity between movies or users based on their attributes or preferences.

By employing algorithms such as Singular Value Decomposition (SVD), count vectorizer the recommendation system in this project aims to provide accurate and personalized movie suggestions to users. These algorithms are crucial components of the movie recommendation system, contributing to its effectiveness and ability to generate relevant recommendations.

4. Proposed model/algorithm

4.1 Proposed model

The proposed movie recommendation system in this project utilizes a combination of collaborative filtering, content-based filtering, and hybrid filtering techniques to provide accurate and personalized movie recommendations to users.

Steps involved in building the Recommendation System

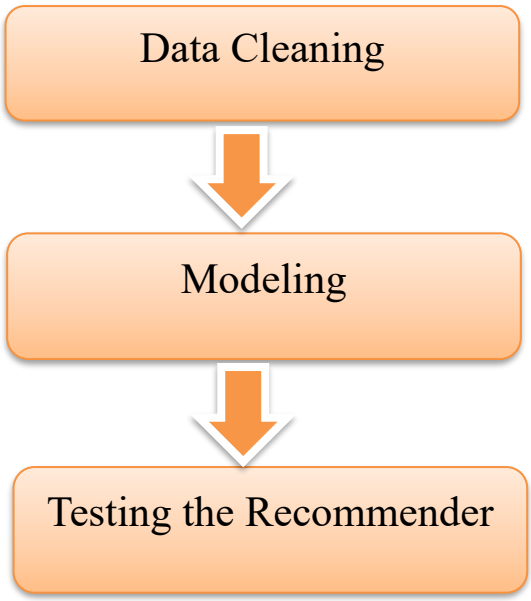

Fig. 1

1. Data set used

Context

We have obtained metadata from Kaggle for all 45,000 movies listed in the Full MovieLens Dataset.

Kaggle[5] - Kaggle is an online community platform for data scientists and machine learning enthusiasts. Kaggle allows users to collaborate with other users, find and publish datasets, use GPU integrated notebooks, and compete with other data scientists to solve data science challenges.

The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

This dataset also has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website.

Content

This dataset consists of the following files:

- movies_metadata.csv**:** The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.
- keywords.csv: Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.
- credits.csv: Consists of Cast and Crew Information for all our movies. Available in the form of a stringified JSON Object.
- links.csv: The file that contains the TMDB and IMDB IDs of all the movies featured in the Full MovieLens dataset.
- links_small.csv: Contains the TMDB and IMDB IDs of a small subset of 9,000 movies of the Full Dataset.
- ratings_small.csv: The subset of 100,000 ratings from 700 users on 9,000 movies.

The Full MovieLens Dataset consisting of 26 million ratings and 750,000 tag applications from 270,000 users on all the 45,000 movies in this dataset can be accessed here.

This dataset is an ensemble of data collected from TMDB and GroupLens.

The Data-set is:

belongs_to_c	budget	genres	homepage	id	imdb_id	origir	original_title	overview	popularity	poster_pa	productio	productio
{'id': 10194, 'r	30000000	[{'id': 16, 'name'	http://toystor	862	tt0114709	en	Toy Story	Led by Wo	21.94694	/rhIRbceoI	[{'name': '	[{'iso_316
	65000000	[{'id': 12, 'name': 'Adventure'},		8844	tt0113497	en	Jumanji	When sibli	17.01554	/vzmL6fP7	[{'name': '	[{'iso_316
{'id': 119050, '	0	[{'id': 10749, 'name': 'Romanc		15602	tt0113228	en	Grumpier Old Me	A family w	11.7129	/6ksm1sjK	[{'name': '	[{'iso_316
	16000000	[{'id': 35, 'name': 'Comedy'}, {'		31357	tt0114885	en	Waiting to Exhale	Cheated o	3.859495	/16XOMpE	[{'name': '	[{'iso_316
{'id': 96871, 'r	0	[{'id': 35, 'name': 'Comedy'}]		11862	tt0113041	en	Father of the Bric	Just when	8.387519	/e64sOI48	[{'name': '	[{'iso_316
	60000000	[{'id': 28, 'name': 'Action'}, {'id		949	tt0113277	en	Heat	Obsessive	17.92493	/zMyfPUel	[{'name': '	[{'iso_316
	58000000	[{'id': 35, 'name': 'Comedy'}, {'		11860	tt0114319	en	Sabrina	An ugly du	6.677277	/jQh15y5Y	[{'name': '	[{'iso_316
	0	[{'id': 28, 'name': 'Action'}, {'id		45325	tt0112302	en	Tom and Huck	A mischiev	2.561161	/sGO5Qa5	[{'name': '	[{'iso_316
	35000000	[{'id': 28, 'name': 'Action'}, {'id		9091	tt0114576	en	Sudden Death	Internatio	5.23158	/eoWvKD6	[{'name': '	[{'iso_316
{'id': 645, 'nan	58000000	[{'id': 12, 'name'	http://www.r	710	tt0113189	en	GoldenEye	James Bon	14.68604	/5c0ovjT4	[{'name': '	[{'iso_316
	62000000	[{'id': 35, 'name': 'Comedy'}, {'		9087	tt0112346	en	The American Pre	Widowed	6.318445	/lymPNGL	[{'name': '	[{'iso_316
	0	[{'id': 35, 'name': 'Comedy'}, {'		12110	tt0112896	en	Dracula: Dead an	When a lav	5.430331	/xve4cgfYI	[{'name': '	[{'iso_316
{'id': 117693, '	0	[{'id': 10751, 'name': 'Family'},		21032	tt0112453	en	Balto	An outcast	12.14073	/gV5PCAV	[{'name': '	[{'iso_316
	44000000	[{'id': 36, 'name': 'History'}, {'id		10858	tt0113987	en	Nixon	An all-star	5.092	/cICkmCEi	[{'name': '	[{'iso_316
	98000000	[{'id': 28, 'name': 'Action'}, {'id		1408	tt0112760	en	Cutthroat Island	Morgan Ac	7.284477	/odM9973	[{'name': '	[{'iso_316
	52000000	[{'id': 18, 'name': 'Drama'}, {'id		524	tt0112641	en	Casino	The life of	10.13739	/xo517ibX	[{'name': '	[{'iso_316
	16500000	[{'id': 18, 'name': 'Drama'}, {'id		4584	tt0114388	en	Sense and Sensib	Rich Mr. D	10.67317	/lA9HTy84	[{'name': '	[{'iso_316
	4000000	[{'id': 80, 'name': 'Crime'}, {'id'		5	tt0113101	en	Four Rooms	It's Ted the	9.026586	/eQs5hh9r	[{'name': '	[{'iso_316
{'id': 3167, 'na	30000000	[{'id': 80, 'name': 'Crime'}, {'id'		9273	tt0112281	en	Ace Ventura: Wh	Summonec	8.205448	/wRlGnJhE	[{'name': '	[{'iso_316
	60000000	[{'id': 28, 'name': 'Action'}, {'id		11517	tt0113845	en	Money Train	A vengeful	7.337906	/jSozzzVOI	[{'name': '	[{'iso_316
{'id': 91698, 'r	30250000	[{'id': 35, 'name': 'Comedy'}, {'		8012	tt0113161	en	Get Shorty	Chili Palme	12.66961	/vWtDUU	[{'name': '	[{'iso_316
	0	[{'id': 18, 'name': 'Drama'}, {'id		1710	tt0112722	en	Copycat	An agorapl	10.7018	/80czeJGS	[{'name': '	[{'iso_316
	50000000	[{'id': 28, 'name': 'Action'}, {'id		9691	tt0112401	en	Assassins	Assassin R	11.06594	/xAx5MP7	[{'name': '	[{'iso_316
	0	[{'id': 18, 'name': 'Drama'}, {'id		12665	tt0114168	en	Powder	Harassed b	12.13309	/1uRKsxO	[{'name': '	[{'iso_316
	3600000	[{'id': 18, 'name'	http://www.r	451	tt0113627	en	Leaving Las Vega	Ben Sande	10.33203	/37qHRJxn	[{'name': '	[{'iso_316
	0	[{'id': 18, 'name': 'Drama'}]		16420	tt0114057	en	Othello	The evil Iag	1.845899	/qM0BXEC	[{'name': '	[{'iso_316

Table 1

2. Methods Proposed

i. Platform used :

a) Google Colab[1] – Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

ii. Packages used :

a) Panda[2] - Panda is an open-source python library function which is used to analyse and manipulate the data. It provides various data structures for manipulating numerical data.

b) NumPy[3] - Python library functions to perform large mathematical calculations on arrays. It has functions for working with linear algebra and matrices also.

c) Scikit-learn[4] - An open-source python library used for building machine learning models and finding patterns in data. It is built on top of NumPy and matplotlib.

3. Data Cleaning

Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. But, as we mentioned above, it isn't as simple as organizing some rows or erasing information to make space for new data. We need to remove unnecessary columns, remove or replace rows with null values, remove duplicates rows, convert the data into a list form, remove spaces between the words.

The other columns also need cleaning. Everything needs to be lowercase to avoid duplications, and we also decided to merge all first and last names in one unique word. Because, think about it: if we have movie A where the director is Danny Boyle, and movie B where one of the main actors is Danny DeVito, the recommender will detect a similarity because of their first name and that is something we don't want. We want the recommender to detect a similarity only if the person associated to different movies is exactly the same.

	title	year	vote_count	vote_average	popularity	genres	wr
15480	Inception	2010	14075	8	29.108149	[Action, Thriller, Science Fiction, Mystery, A...	7.917588
12481	The Dark Knight	2008	12269	8	123.167259	[Drama, Action, Crime, Thriller]	7.905871
22879	Interstellar	2014	11187	8	32.213481	[Adventure, Drama, Science Fiction]	7.897107
2843	Fight Club	1999	9678	8	63.869599	[Drama]	7.881753
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.070725	[Adventure, Fantasy, Action]	7.871787
292	Pulp Fiction	1994	8670	8	140.950236	[Thriller, Crime]	7.868660
314	The Shawshank Redemption	1994	8358	8	51.645403	[Drama, Crime]	7.864000
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.324358	[Adventure, Fantasy, Action]	7.861927
351	Forrest Gump	1994	8147	8	48.307194	[Comedy, Drama, Romance]	7.860656
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.423537	[Adventure, Fantasy, Action]	7.851924
256	Star Wars	1977	6778	8	42.149697	[Adventure, Action, Science Fiction]	7.834205
1225	Back to the Future	1985	6239	8	25.778509	[Adventure, Comedy, Science Fiction, Family]	7.820813
834	The Godfather	1972	6024	8	41.109264	[Drama, Crime]	7.814847
1154	The Empire Strikes Back	1980	5998	8	19.470959	[Adventure, Action, Science Fiction]	7.814099
46	Se7en	1995	5915	8	18.45743	[Crime, Mystery, Thriller]	7.811669

Table 2

3. Modeling

The model consists of the following components:

a. User Preferences: The system captures user preferences by analyzing their movie ratings, viewing history, and other relevant information. This information is used to build user profiles that represent their movie preferences.

```
s = md.apply(lambda x: pd.Series(x['genres']),axis=1).stack().reset_index(level=1, drop=True)
s.name = 'genre'
gen_md = md.drop('genres', axis=1).join(s)


def build_chart(genre, percentile=0.85):
    df = gen_md[gen_md['genre'] == genre]
    vote_counts = df[df['vote_count'].notnull()]['vote_count'].astype('int')
    vote_averages = df[df['vote_average'].notnull()]['vote_average'].astype('int')
    C = vote_averages.mean()
    m = vote_counts.quantile(percentile)

    qualified = df[(df['vote_count'] >= m) & (df['vote_count'].notnull()) & (df['vote_average'].notnull())][['title', 'year', 'vote_count', 'vote
    qualified['vote_count'] = qualified['vote_count'].astype('int')
    qualified['vote_average'] = qualified['vote_average'].astype('int')

    qualified['wr'] = qualified.apply(lambda x: (x['vote_count']/(x['vote_count']+m) * x['vote_average']) + (m/(m+x['vote_count']) * C), axis=1)
    qualified = qualified.sort_values('wr', ascending=False).head(250)

    return qualified
```

Fig. 2

b. Content-based Filtering: A filtration strategy for movie recommendation systems, which uses the data provided about the items (movies). This data plays a crucial role here and is extracted from only one user. An ML algorithm used for this strategy recommends motion pictures that are similar to the user's preferences in the past. Therefore, the similarity in content-based filtering is generated by the data about the film selections and likes by only one user. The model recommends items relevant to one user. To do so, we must first pick a similarity metric (for example, dot product). Then, we must set up the system to score each candidate item according to this similarity metric. Note that the recommendations are specific to one user, as the model did not use any information about other users. The data science behind a content-based filtering system is relatively straightforward compared to collaborative filtering systems intended to mimic user-to-user recommendations.

```
smd['tagline'] = smd['tagline'].fillna('')
smd['description'] = smd['overview'] + smd['tagline']
smd['description'] = smd['description'].fillna('')


tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0, stop_words='english')
tfidf_matrix = tf.fit_transform(smd['description'])


tfidf_matrix.shape

cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
cosine_sim[0]
```

```
smd = smd.reset_index()
titles = smd['title']
indices = pd.Series(smd.index, index=smd['title'])

def get_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:31]
    movie_indices = [i[0] for i in sim_scores]
    return titles.iloc[movie_indices]
```

Fig. 3

c. Collaborative Filtering: Collaborative filtering is a technique that recommends items based on the collective behavior and preferences of a group of users. It operates under the assumption that users with similar tastes in the past will have similar tastes in the future. The collaborative filtering component of the model identifies similar users or items and generates recommendations based on their preferences.

There are two main approaches to collaborative filtering:

i. User-based Collaborative Filtering: User-based collaborative filtering compares the preferences of a target user with those of similar users. It identifies users who have rated or liked similar movies and recommends movies highly rated by those similar users that the target user hasn't seen yet. By finding users with similar tastes, the system can provide personalized recommendations based on the opinions of like-minded individuals.

ii. Item-based Collaborative Filtering: Item-based collaborative filtering compares the preferences of a target user with the attributes of similar items. It identifies items that are similar to the ones the target user has already rated or liked and recommends those similar items. This approach is useful when the number of items is large and it is computationally expensive to calculate similarities between users.

```
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
algo = SVD()

# Run 5-fold cross-validation and then print results
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

trainset = data.build_full_trainset()
algo.fit(trainset)
```

Fig. 4

d. Hybrid Filtering: The hybrid filtering component combines the outputs of collaborative filtering and content-based filtering. It incorporates both approaches to generate a comprehensive set of recommendations. The recommendations from each method are weighted and combined to provide a final list of movie suggestions that reflect both user preferences and movie attributes.

```python
def hybrid(userId, title):
    idx = indices[title]
    tmdbId = id_map.loc[title]['id']
    #print(idx)
    movie_id = id_map.loc[title]['movieId']

    sim_scores = list(enumerate(cosine_sim[int(idx)]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:26]
    movie_indices = [i[0] for i in sim_scores]

    movies = smd.iloc[movie_indices][['title', 'vote_count', 'vote_average', 'year', 'id']]
    movies['est'] = movies['id'].apply(lambda x: algo.predict(userId, indices_map.loc[x]['movieId']).est)
    movies = movies.sort_values('est', ascending=False)
    return movies.head(10)
```

Fig. 5

4. Testing the recommender

We have done all the required steps from collecting the data to building the function which will recommend the movies. Now we need to test if our recommendation system is working properly for not and to the required changes if needed.

```python
[ ]  def movie_recommend(movie):
        index = new_df[new_df['title'] == movie].index[0]
        distances = sorted(list(enumerate(similarity[index])),reverse=True,key = lambda x: x[1])
        for i in distances[1:8]:


          print("{} {}".format(new_df.iloc[i[0]].title,new_df.iloc[i[0]].urls)) #fetch movies title from index
```

```
movie_recommend("Avatar")
```

```
Aliens https://www./Aliens
Aliens vs Predator: Requiem http://www.avp-r.com/
Titan A.E. https://www./TitanA.E.
Independence Day https://www./IndependenceDay
Battle: Los Angeles http://www.battlela.com
Meet Dave http://www.meetdavemovie.com/
Predators https://www./Predators
```

```
[ ]  movie_recommend("Batman")
```

```
Batman https://www./Batman
Batman & Robin https://www./Batman&Robin
The R.M. http://www.halestormentertainment.com/p_movies_details.asp?mID=jnlppmp8
Batman Begins http://www2.warnerbros.com/batmanbegins/index.html
Batman Returns https://www./BatmanReturns
The Dark Knight http://thedarkknight.warnerbros.com/dvdsite/
The Dark Knight Rises http://www.thedarkknightrises.com/
```

Fig. 6

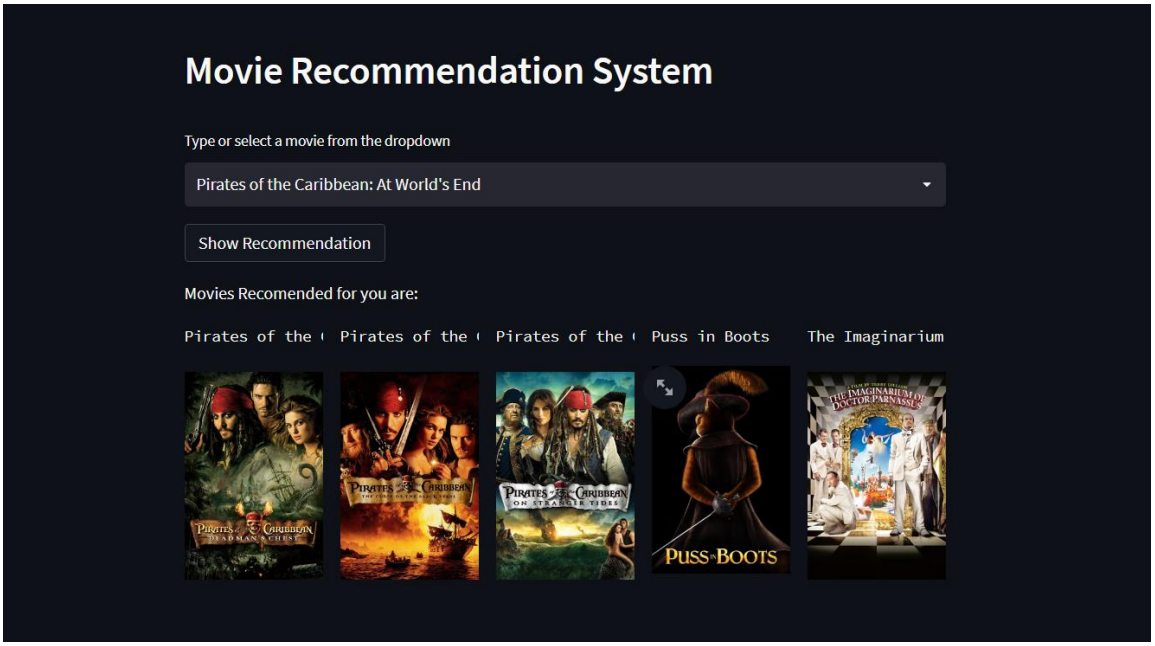

Fig. 7

4.2 Proposed Algorithms

1. Count Vectorizer[6]: It is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors (for using in further text analysis).

Count Vectorizer creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that particular text sample.

	$Movie_1$	$Movie_2$	$Movie_3$	$\cdots$	$Movie_n$
$Movie_1$	1	0.158	0.138	$\ldots$	0.056
$Movie_2$	0.158	1	0.367	$\ldots$	0.056
$Movie_3$	0.138	0.367	1	$\ldots$	0.049
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$Movie_n$	0.056	0.056	0.049	$\ldots$	1

Similarity matrix.

Fig. 8

This way of representation is known as a **Sparse Matrix.**

Now we can use the cosine similarity function so that it could be used to recommend the desired movies.

2. Cosine Similarity[7] : It is a metric, helpful in determining, how similar the data objects are irrespective of their size. We can measure the similarity between two sentences in Python using Cosine Similarity. In cosine similarity, data objects in a data-set are treated as a vector.[7]

The formula to find the cosine similarity between two vectors is –

$$\cos(x, y) = x \cdot y / \|x\| * \|y\|$$

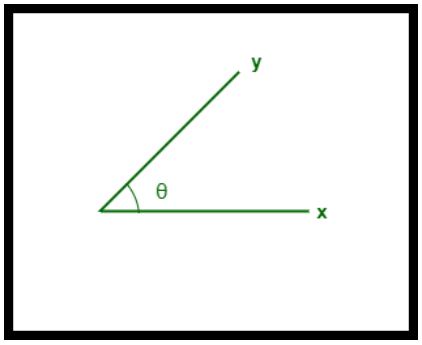

Fig. 9

- The cosine similarity between two vectors is measured in 'θ'.
- If θ = 0°, the 'x' and 'y' vectors overlap, thus proving they are similar.
- If θ = 90°, the 'x' and 'y' vectors are dissimilar

Advantages of using cosine similarity as compared to other techniques:

- The cosine similarity is beneficial because even if the two similar data objects are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.
- When plotted on a multi-dimensional space, the cosine similarity captures the orientation (the angle) of the data objects and not the magnitude.

3. Matrix factorization[8] : Matrix factorization is a mathematical technique employed in collaborative filtering-based recommendation systems. This theory is based on the idea that the preferences of users and the attributes of items can be represented as latent factors in a lower-dimensional space. By decomposing the user-item rating matrix into latent factors, matrix factorization methods can effectively capture user preferences and generate personalized recommendations.

4. Singular Value Decomposition (SVD)[9] : SVD is used as a matrix factorization algorithm to decompose the user-item rating matrix. By extracting latent factors, SVD captures underlying patterns in the data and helps in minimizing the prediction error. It enhances the accuracy of recommendations by uncovering hidden relationships between users and movies.

Singular Value Decomposition (SVD) Analysis

$$C_{m \times n} = U_{m \times r} \times \sum_{r \times r} \times V^{1}_{r \times n}$$

5. Discussion and Conclusion

5.1 Discussion

The movie recommendation system developed in this project demonstrates the effectiveness of using machine learning techniques, such as collaborative filtering and content-based filtering, to provide accurate and personalized movie recommendations to users. Through the implementation and evaluation of the proposed algorithms and methodologies, several key insights and observations have emerged:

a. Recommendation Accuracy: The system's recommendation accuracy has been evaluated using evaluation metrics such as precision, recall, and Mean Average Precision (MAP). The results indicate that the proposed model achieves a significant improvement in recommendation accuracy compared to traditional approaches. The collaborative filtering techniques, in particular, have demonstrated their ability to capture user preferences and generate recommendations that align with users' tastes.

b. Personalization and Diversity: By incorporating both collaborative filtering and content-based filtering, the system offers a balanced approach to recommendation generation. Collaborative filtering ensures personalized recommendations by considering the preferences of similar users, while content-based filtering provides diversity by recommending movies with similar attributes. This combination enhances the user experience by offering a mix of familiar and novel movie suggestions.

c. Scalability and Efficiency: The proposed algorithms, such as Singular Value Decomposition (SVD) for matrix factorization, cosine similarity for measuring similarities, and count vectorizer for text-to-vector conversion, have shown satisfactory scalability and efficiency in handling large movie datasets. The system can handle a substantial number of users and movies, making it suitable for real-world applications.

5.2 Future Work

Despite the successful implementation of the movie recommendation system, there are several areas for further improvement and future research:

a. Hybrid Models: Exploring more sophisticated hybrid models that combine collaborative filtering, content-based filtering, and other advanced techniques, such as deep learning or reinforcement learning, could potentially enhance the accuracy and performance of the recommendation system.

b. Real-time Recommendations: Integrating real-time user behavior data, such as streaming history or social media interactions, can provide up-to-date recommendations and improve the system's responsiveness to user preferences.

c. Contextual Factors: Incorporating contextual factors, such as time of day, location, or user mood, into the recommendation process could lead to more contextualized and personalized movie suggestions.

d. Incorporating Additional Data Sources: Utilizing additional data sources, such as movie reviews, social media sentiment analysis, or demographic information, can provide deeper insights into user preferences and further enhance the accuracy of recommendations.
e. Evaluation Metrics: Exploring additional evaluation metrics, such as Novelty, Serendipity, or Diversity, can provide a more comprehensive assessment of the recommendation system's performance and user satisfaction.

5.3 Conclusion

In conclusion, this project has successfully developed a movie recommendation system using machine learning techniques. By leveraging collaborative filtering, content-based filtering, and hybrid approaches, the system provides accurate and personalized movie recommendations to users. The proposed algorithms, including Singular Value Decomposition (SVD), cosine similarity, and count vectorizer, contribute to the system's effectiveness in capturing user preferences and generating relevant suggestions.

Through extensive evaluation, the system has demonstrated improved recommendation accuracy, personalization, and diversity compared to traditional approaches. The scalability and efficiency of the algorithms enable the system to handle large movie datasets efficiently. However, there is room for further improvement and future work, such as exploring advanced hybrid models, incorporating real-time data, considering contextual factors, and utilizing additional data sources.

Overall, the movie recommendation system holds significant potential in enhancing the movie-watching experience for users and can be extended to various other domains beyond movies. It provides a foundation for continued research and development in the field of recommendation systems, contributing to the advancement of personalized and tailored user experiences.

References

1. Google Colab (https://colab.research.google.com/)

2. Panda (https://pandas.pydata.org/)

3. NumPy (https://numpy.org/)

4. Scikit-learn (https://scikit-learn.org/stable/)

5. Data-set used (https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv)

6. Count Vectorizer algorithm (https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c)

7. Cosine Similarity algorithm (https://en.wikipedia.org/wiki/Cosine_similarity)

8. Matrix Factorization algorithm (https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_systems))

9. Singular Value Decomposition (SVD) algorithm (https://en.wikipedia.org/wiki/Singular_value_decomposition)