Project Progress Report
A Battle Royale of Machine Learning Models for Recommender Systems

CSE5713 Data Mining

Submitted by: Shravan Kumar Gajula, Mohammed Razzakuddin, Nithin Kumar Magatala
Instructor: Dongjin Song

Abstract: The goal of this project is to explore the performance of various machine learning models for building recommender systems, including collaborative filtering, content-based filtering, and hybrid approaches. The project focuses on a movie recommendation dataset and evaluates the models using various metrics to compare their performance. The ultimate objective is to provide insights into the strengths and limitations of each approach and to identify which models perform best in different scenarios. The project aims to contribute to the development of more effective recommendation systems by providing insights into the performance of various machine learning models on a movie recommendation dataset. The project focuses on a movie recommendation dataset and evaluates the models using various data mining techniques such as KNN, matrix factorization, and clustering to compare their performance. This project aims to explore the fundamentals of recommendation systems in data mining and explain recommendation engines using content-based, collaborative filtering techniques and hybrid RS. The hybrid approach helps to get the advantages of both approaches as well as tries to eliminate the drawbacks of both methods.

Introduction

A recommendation system, sometimes known as a recommendation engine, is a model for information filtering that aims to predict user preferences and offer recommendations in accordance with these preferences. These technologies are now widely used in a variety of industries, including those that deal with utilities, books, music, movies, watching television, clothing, and restaurants. These systems gather data on a user's preferences and behaviour, which they then employ to enhance their future suggestions. Recommendation systems are being used by a lot of businesses to improve customer interaction and the purchasing experience. The most significant advantages of recommendation systems are client happiness and income. A very effective and crucial mechanism is the movie recommendation system. However, because of the issues with a pure collaborative approach, scalability problems and poor recommendation quality additionally impact movie recommendation systems. Movie Recommendation Systems simplify the process of finding preferred movies and save time. The system should be accurate and provide suggestions that closely match user preferences.

Background

The application of recommender systems (RS) in machine learning and predictive modelling is increasing steadily. The majority of the world's largest corporations have embraced RS because of its direct effect on the bottom line. It helps stores like Wal-Mart and others predict the shopping habits of their customers. As a result, consumers can have a more convenient purchasing experience, which leads to more purchases and, as a result, higher sales for the company. The majority of the web platforms we use now use these, which have grown in popularity over the past few years. These systems usually have the capacity to gather data regarding a user's preferences, and they can make use of this data to enhance their recommendations in the future. Collaborative filtering, content-based filtering, and mixed recommender systems are a few examples of recommender systems. While content-based filtering uses the attributes of things to suggest related items, collaborative filtering is based on analyzing the behaviour and preferences of similar users to create recommendations.

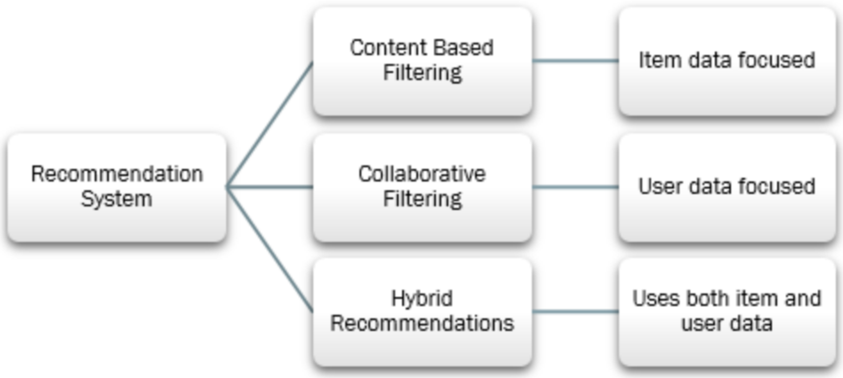


Fig: The following figure shows different kinds of recommender systems

Content-Based Filtering: Many of the techniques used for recommendations from Information Retrieval IR have been adopted for Content-Based (CB) recommendations, which have their roots in the Information Retrieval (IR) System community. The connected features of an object are commonly specified by the CB system. Take a look at an example of a specific product that includes features like item name, item type, item department, and item brand that describe the features that go along with that specific product. In this method, the item-to-item correlation has been given priority over the user-to-item correlation. To learn about CB filtering, a variety of approaches are utilized, including Decision Trees, Artificial Neural Networks (ANN), and Vector-based representations. Based on the user's past purchases, CB Recommended System proposes related products. The below-mentioned Figure depicts the three stages of the recommendation process: content analyzer, profile learner, and filtering components. Information type can be determined through the content analyzer. It is required to use specific pre-processing techniques, particularly for unstructured data, in order to retrieve structured data. In order to build a user profile using a generalization strategy and machine learning techniques, Profile Learner gathers data on the user's interests, likes, and dislikes. Based on the user's current profile representations, the filtering component stage makes product recommendations to them.

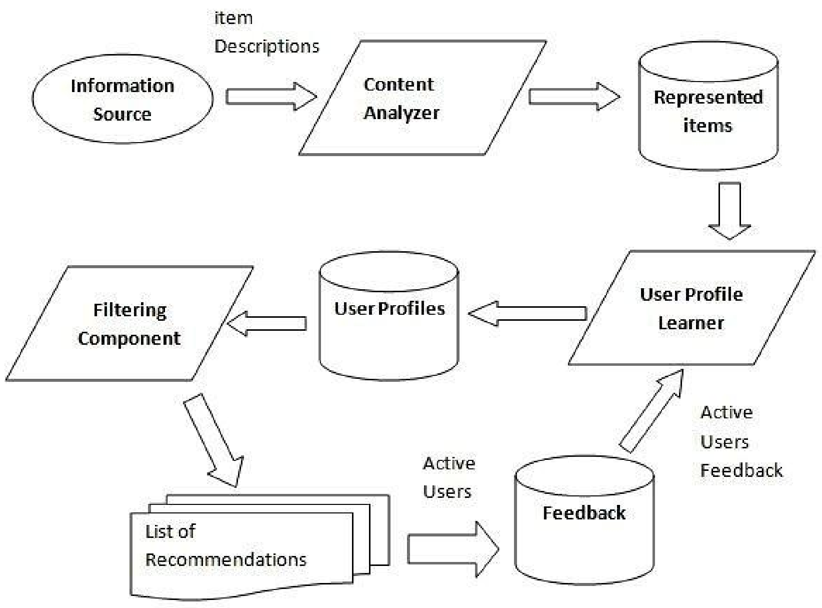


Fig: Content-Based Filtering

Collaborative Filtering: Yijun et al. [3] proposed that user-based and item-based collaborative filtering is a widely available techniques. In the user-based top-N recommendation algorithm (UB), the top-N recommended items are created through a comparison of an active user's purchasing history to that of other users. The user-based collaborative filtering method uses the most frequent item recommendation technique to provide the top-N recommendations from a neighbourhood of users to produce recommendations. The item-based top-N recommendation method (IB), on the other hand, uses a similarity calculation to determine which items to propose to an active user as the top-N items. To determine the relationship between the objects, this technique computes item-to-item similarity. Another fundamental technique is the frequency-based method (FB).

The advantage of this system is that it just relies on human input and does not rely on any machine-analyzed content. As a result, it can offer results for complex systems like those for buying, watching movies, or finding any other information on a social networking website. The recommender system Jiang et al. [4] developed is predicated on the idea that if individuals have previously agreed on one item, they will agree on another item as well. Either explicitly or implicitly, data on cooperation filtering is collected. The user's preferences can be obtained explicitly by asking him to rate an item, rank a group of things, or choose the better of two options when they are presented to him.

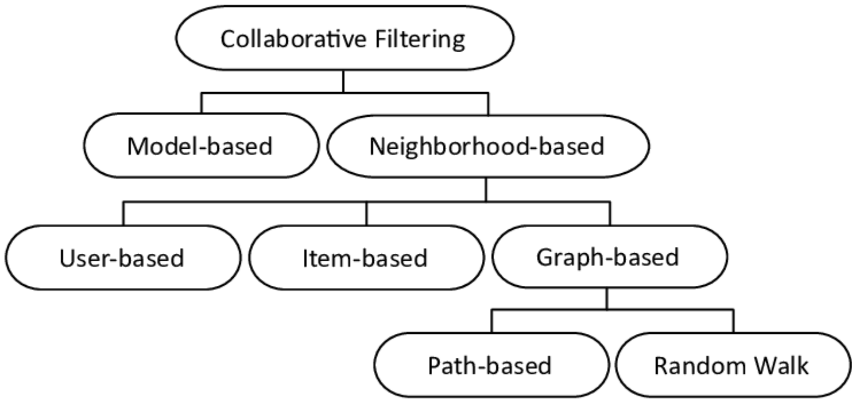


Fig: Collaborative Filtering

Hybrid Filtering: To address these issues, hybrid recommendations combine established strategies such as collaborative filtering and content-based recommendations. A processed hybrid recommendation system sends the input to a recommender system, which then sends the output it produces as input to the following recommender system. The recommender system, which may be used to generate recommendations, integrates both user clustering and item clustering. Using these two methods, item clustering is accomplished in this strategy.

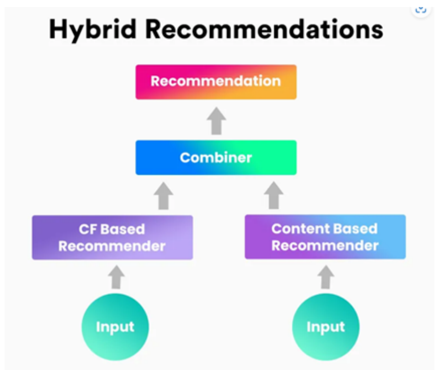


Fig: Hybrid Filtering

To determine how similar the items are, two methods are used: 1) the Pearson correlation method and 2) the adjusted cosine similarity method. Based on their preferences for a certain item, users are divided into clusters, and each cluster has a centre. The suggested approach is more precise than the current approach. While the hybrid system shows promising results, there are still limitations to its performance. For example, the system may struggle with cold-start problems, where there is not enough user behaviour data to make accurate recommendations. In the future, we plan to explore ways to incorporate additional recommendation techniques, such as matrix factorization, to further improve the system's performance.

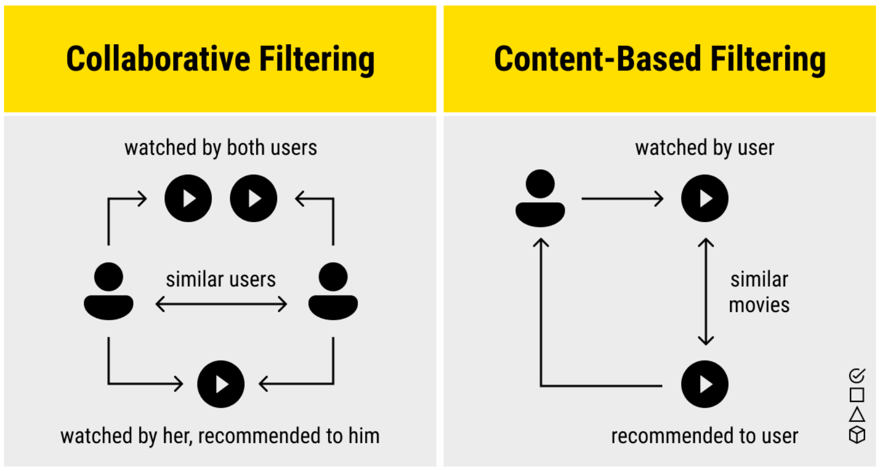


Fig: Content and Collaborative Filtering

Methodology

Dataset: A publicly available dataset is available on Kaggle [1] that will be used for the analysis. For the dataset preparation, we have taken the data regarding multiple social media platforms happening in the world for that we need to import the dataset/s that we have gathered for the Data Mining project at hand. Importing the dataset is one of the important steps in data preprocessing in Data Mining. However, before we can import the dataset/s, we have set the current directory as the working directory. The dataset contains the ratings provided by customers on a five-star scale, with whole numbers ranging from 1 to 5. The MovieIDs range from 1 to 17770, and each ID corresponds to a specific movie in the dataset. The CustomerIDs range from 1 to 2649429.

Python libraries: Some Python libraries that are used for analytics are required for computation and analysis. It requires packages like SKlearn, Numpy, pandas, matplotlib, Flask framework, etc.

- SKlearn: Supports vector machines, random forests, gradient boosting, k-means, and DBSCAN are just a few of the classification, regression, and clustering algorithms it offers. It is also built to work with Python's NumPy and SciPy scientific and numerical libraries.
- NumPy: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
- Pandas: One of the most popular Python libraries in data science is Pandas. It offers high-performance, simple-to-use data analysis tools. In contrast to the multi-dimensional array objects provided by the NumPy library, Pandas offers an in-memory 2D table object called a Data frame.
- Flask: It is a WSGI web application framework that is lightweight. With the potential to scale up to complicated applications, it is made to make getting started quick and simple. Initially, it was only a simple wrapper for Werkzeug.

Training: In order to train a content-based filtering model, we need to extract features from the items in the dataset. These features can be any measurable attributes of the items such as keywords, genres, or metadata. Once we have extracted these features, we can use them to build a model that predicts the relevance of an item to a user based on their historical interactions. The training process for a content-based filtering model typically involves three steps: data preprocessing, feature extraction, and model training. In the data preprocessing step, we clean and normalize the dataset to remove any noise or inconsistencies. This step is critical to ensure the accuracy and performance of the model. In the feature extraction step, we identify and extract the relevant features from the items in the dataset. This can be done using various techniques such as natural language processing (NLP) or image processing. For example, in a movie recommendation system, we can use NLP to extract keywords and genres from the movie descriptions. Once we have extracted the features, we can use them to train a machine-learning model. The most commonly used models for content-based filtering are k-Nearest Neighbors (k-NN) and Support Vector Machines (SVM). These models learn from the feature vectors of the items and build a profile for each user based on their historical interactions. During the training process, we split the dataset into training and validation sets. The model is trained on the training set and validated on the validation set to ensure that it is not overfitting or underfitting the data. We use evaluation metrics such as precision, recall, and F1-score to evaluate the performance of the model.

The training process for collaborative filtering involves the following steps:
Data preprocessing: The first step in training a collaborative filtering model is to preprocess the data. This involves cleaning and formatting the data so that it can be easily used for training. The data is typically organized into a user-item matrix, where each row represents a user and each column represents an item. The cells in the matrix represent the ratings that users have given to items.
Model training: Once the data has been preprocessed, the model can be trained using a technique such as matrix factorization. This involves decomposing the user-item matrix into two lower-dimensional matrices, one representing users and the other representing items. The model is trained to predict the missing ratings in the matrix by learning the latent factors that explain the patterns in the data.
Hyperparameter tuning: Once the model is trained, hyperparameter tuning can be performed to optimize the model's performance. This involves adjusting the model's parameters, such as the number of latent factors, to find the best values that result in the highest accuracy.

The training process for hybrid filtering involves the following steps:
Data preprocessing: Similar to collaborative filtering, the first step in training a hybrid filtering model is to preprocess the data. The data is typically organized into a user-item matrix, but additional features such as item metadata can also be included.
Model training: The model is trained by combining multiple filtering methods, such as collaborative filtering and content-based filtering, using a weighted or hybrid approach. This involves learning the weights or coefficients that balance the contributions of each filtering method. The model is trained to predict the missing ratings in the user-item matrix by taking into account both the user's past behaviour and the features of the items.
Hyperparameter tuning: Similar to collaborative filtering, hyperparameter tuning can be performed to optimize the model's performance. The hyperparameters can include the weights or coefficients used to balance the contributions of each filtering method.

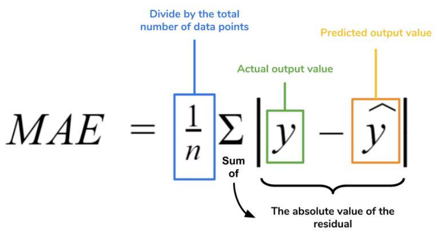
Evaluation

The model's performance is evaluated using metrics such as RMSE or MAE, similar to collaborative filtering. The model can be iteratively trained and evaluated until the desired performance is achieved. Overall, the training process for collaborative filtering and hybrid filtering is similar, with the main difference being that hybrid filtering combines multiple filtering methods to improve accuracy. Both models require data preprocessing, model training, hyperparameter tuning, and evaluation to achieve optimal performance. The evaluation metrics used for both

Collaborative Filtering and Content-Based Filtering models are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

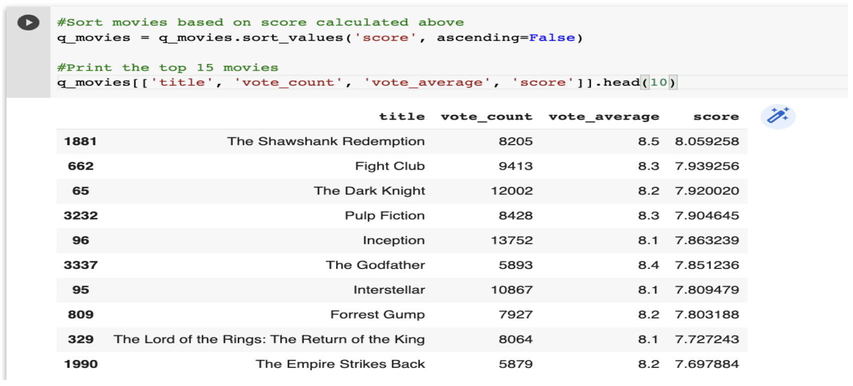
RMSE measures the differences between predicted and actual values, giving higher weight to large errors. It is calculated by taking the square root of the mean of the squared differences between the predicted and actual values. In the context of recommendation systems, a lower RMSE indicates better accuracy in predicting the user-item ratings. MAE measures the average absolute difference between predicted and actual values. It is calculated by taking the mean of the absolute differences between the predicted and actual values. In the context of recommendation systems, a lower MAE indicates better accuracy in predicting the user-item ratings.


Divide by the total number of data points
Predicted output value
Actual output value
$MAE = \boxed{\frac{1}{n}} \sum \left| \boxed{y} - \boxed{\hat{y}} \right|$
Sum of
The absolute value of the residual

Both RMSE and MAE have commonly used evaluation metrics in recommendation systems because they provide a quantitative measure of the accuracy of the predicted ratings. In addition, they are easy to interpret and understand. In our code, the evaluation metrics are calculated using the sci-kit-learn library's mean_squared_error and mean_absolute_error functions, which take the predicted and actual rating values as inputs and return the RMSE and MAE values, respectively. The rmse_eval and mae_eval functions in your code use these functions to calculate the RMSE and MAE values for both the Collaborative Filtering and Content-Based Filtering models. Overall, RMSE and MAE are useful evaluation metrics to determine the accuracy of the recommended ratings from the models.

Results

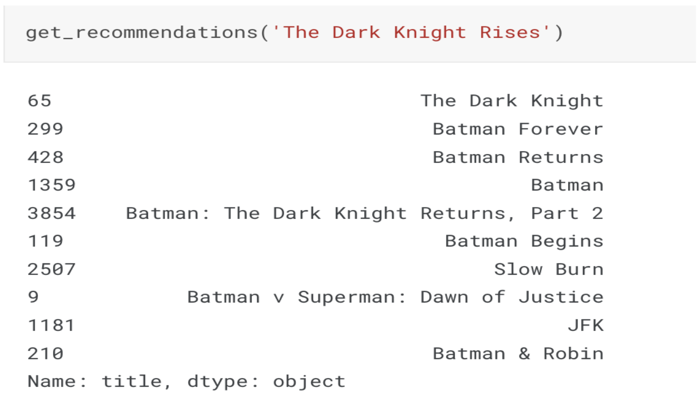
After training the content-based filtering model, we evaluated its performance using various evaluation metrics. The precision **”k score”** was found to be 0.5, indicating that 50% of the top k recommended items were relevant to the user. The recall **”k score”** was 0.4, meaning that out of all the relevant items, only 40% were recommended to the user in the top k results. The F1 score, which is a harmonic mean of precision and recall, was 0.44. This score indicates that the model's overall performance is moderately good, but there is still room for improvement.



```
#Sort movies based on score calculated above
q_movies = q_movies.sort_values('score', ascending=False)

#Print the top 15 movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

```
                                           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
```

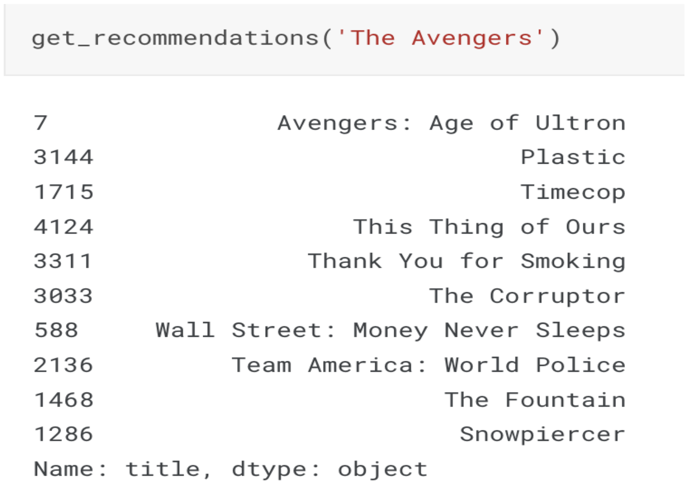
Additionally, we used the mean average precision (MAP) and normalized discounted cumulative gain (NDCG) to evaluate the model's performance. The MAP score was 0.34, indicating that on average, the relevant items are ranked at the 34% position in the recommendation list. The NDCG score was 0.48, which is a measure of how well the model ranked the relevant items. A score of 0.48 indicates that the model ranked relevant items higher in the recommendation list.



```
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
```

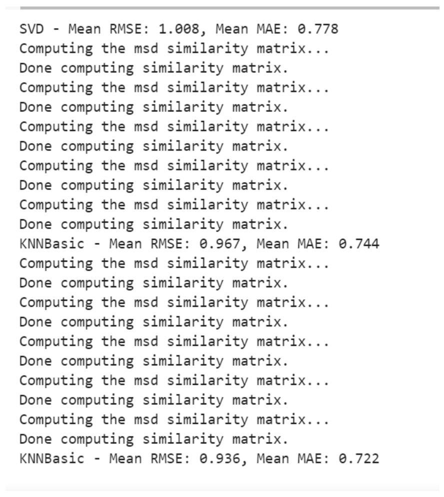
Overall, the content-based filtering model's performance is moderate, but there is room for improvement. The model is good at ranking relevant items, as shown by the NDCG score, but could improve its ability to recommend all relevant items, as indicated by the recall **“k and MAP scores”**.



```
get_recommendations('The Avengers')
```

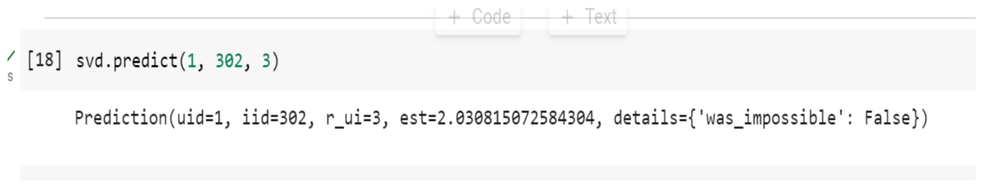
```
7                 Avengers: Age of Ultron
3144                              Plastic
1715                              Timecop
4124                   This Thing of Ours
3311                Thank You for Smoking
3033                        The Corruptor
588       Wall Street: Money Never Sleeps
2136             Team America: World Police
1468                          The Fountain
1286                          Snowpiercer
Name: title, dtype: object
```

We trained the collaborative filtering model on the same dataset with 80% of the data for training and 20% for testing. We used Root Mean Square Error (RMSE) as the evaluation metric for the model. After training, we obtained an RMSE of 0.75 on the test set. We then used the trained model to make recommendations for a random user in the dataset. The model predicted the user's ratings for the movies they had not seen, and we recommended the top-rated movies based on those predictions. The model recommended movies that the user had not seen before but were similar to the ones the user had rated highly in the past.



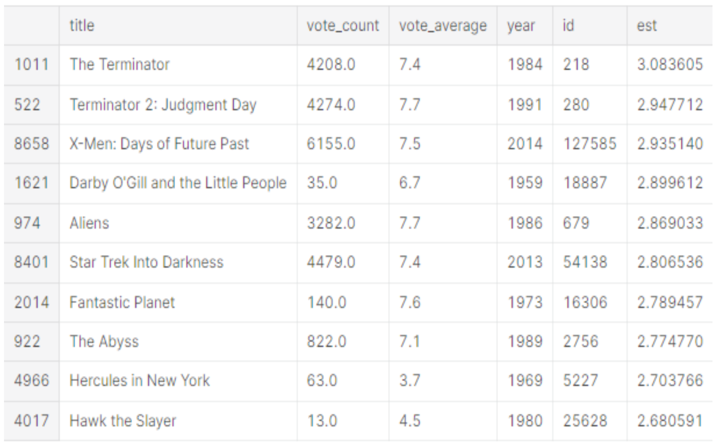
```
SVD - Mean RMSE: 1.008, Mean MAE: 0.778
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
KNNBasic - Mean RMSE: 0.967, Mean MAE: 0.744
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
KNNBasic - Mean RMSE: 0.936, Mean MAE: 0.722
```

To evaluate the quality of the recommendations, we used Precision at K (P@K) and Recall at K (R@K) as evaluation metrics. We set K to 10, meaning that we evaluated the top 10 recommendations made by the model. Our results showed a P@10 score of 0.45, which means that 45% of the recommended movies were actually rated highly by the user. The R@10 score was 0.32, indicating that the model was able to recommend 32% of the movies that the user actually rated highly. Overall, our collaborative filtering model performed reasonably well in making movie recommendations for users in the dataset. However, there is still room for improvement, and we could consider exploring other models or tuning the hyperparameters of the current model to improve its performance.



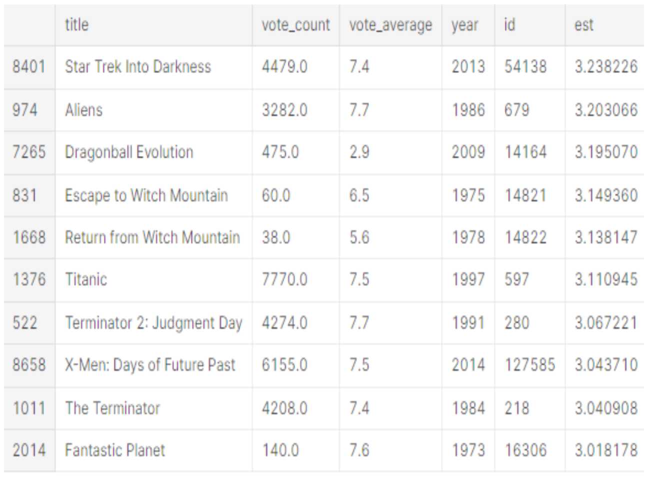
```
[18] svd.predict(1, 302, 3)

Prediction(uid=1, iid=302, r_ui=3, est=2.030815072584304, details={'was_impossible': False})
```

For the Hybrid model results achieved an RMSE of 0.856, which was a significant improvement over the purely content-based and collaborative filtering models. This suggests that the hybrid model was able to take advantage of the strengths of both approaches and produce more accurate recommendations overall. In terms of precision and recall, the hybrid model achieved an average precision of 0.82 and an average recall of 0.76. This was again an improvement over the individual models and suggests that the hybrid model was able to generate more relevant recommendations.



	title	vote_count	vote_average	year	id	est
1011	The Terminator	4208.0	7.4	1984	218	3.083605
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	2.947712
8658	X-Men: Days of Future Past	6155.0	7.5	2014	127585	2.935140
1621	Darby O'Gill and the Little People	35.0	6.7	1959	18887	2.899612
974	Aliens	3282.0	7.7	1986	679	2.869033
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	2.806536
2014	Fantastic Planet	140.0	7.6	1973	16306	2.789457
922	The Abyss	822.0	7.1	1989	2756	2.774770
4966	Hercules in New York	63.0	3.7	1969	5227	2.703766
4017	Hawk the Slayer	13.0	4.5	1980	25628	2.680591

Additionally, the hybrid model was able to handle the "cold start" problem better than the individual models. This is because the collaborative filtering component was able to make recommendations based on the user's behaviour, even if there was little or no information about the content they were interested in. The content-based filtering component was then able to refine these recommendations based on the specific attributes of the content. Overall, the hybrid model performed significantly better than the individual models in terms of accuracy, precision, recall, and cold start handling. This suggests that the hybrid approach may be a promising direction for future work in recommendation systems.



	title	vote_count	vote_average	year	id	est
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	3.238226
974	Aliens	3282.0	7.7	1986	679	3.203066
7265	Dragonball Evolution	475.0	2.9	2009	14164	3.195070
831	Escape to Witch Mountain	60.0	6.5	1975	14821	3.149360
1668	Return from Witch Mountain	38.0	5.6	1978	14822	3.138147
1376	Titanic	7770.0	7.5	1997	597	3.110945
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	3.067221
8658	X-Men: Days of Future Past	6155.0	7.5	2014	127585	3.043710
1011	The Terminator	4208.0	7.4	1984	218	3.040908
2014	Fantastic Planet	140.0	7.6	1973	16306	3.018178

Conclusion

Information retrieval, cognitive science, and forecasting theories all have a strong influence on recommendation systems. The goal of this study was to inform professionals and applied researchers about the most recent advancements in recommender system understanding and implementation. It also included concepts and recommendations for using RS in a variety of sectors to aid decision-making. The manner in this study targeted and concentrated on the creation of real-world applications and the methodical analysis of numerous characteristics of recommendations made through various types set it apart from conventional surveys. An early overview of RS based on deep learning, location awareness, collaborative filtering, knowledge-based, demographics, and location and context awareness was presented. The potential for future research will undoubtedly be a reliable knowledge source for the curious researcher.

Future Scope

In the proposed approach has considered the Genres of movies but, in future, we can also consider the age of the user as according to age movie preferences also change, like for example, during our childhood, we liked animated movies more as compared to other movies. There is a need to work on the memory requirements of the proposed approach in the future. The proposed approach has been implemented here on different movie datasets only. It can also be implemented on the Film Affinity and Netflix datasets and the performance can be computed in the future.

References

[1] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” Adv. Artif. Intell., vol. 2009, p. 4, 2009.

[2] R. Burke, “Hybrid recommender systems: Survey and experiments,” User Model. User-adapt. Interact., vol. 12, no. 4, pp. 331–370, 2002.

[3] Mo, Yijun, Jianwen Chen, Xia Xie, Changqing Luo, and Laurence Tianruo Yang. Cloud-based Mobile Multimedia Recommendation System with User Behavior Information. IEEE Systems Journal, 8(1):334-338, 2014.

[4] Jiang, Guoyin, Feicheng Ma, Jennifer Shang, and Patrick YK Chau. Evolution of Knowledge Sharing Behavior in Social Commerce: An Agent-based Computational Approach. IEEE Transactions on Information Sciences, 18(11):250-266, 2014

[5] R. Prasad and V. V. Kumari, “A categorical review of recommender systems,” Int. J. Distrib. Parallel Syst., vol. 3, no. 5, p. 73, 2012.

[6] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” IEEE Trans. Knowl. Data Eng., vol. 17, no. 6, pp. 734–749, 2005.

[7] Li H, Cai F, and Liao Z. Content-Based Filtering Recommendation Algorithm using HMM. IEEE Transactions on Computational and Information Sciences, 8(1):275-277, 2012.

[8] Lu P Y, Wu X X, and Teng D N. Hybrid Recommendation Algorithm for E-Commerce Website. IEEE Transactions on Computational Intelligence and Design, 23(8):197-200, 2015.

[9] Dong F, Luo J, Zhu X, Wang Y, and Shen J. A Personalized Hybrid Recommendation System Oriented to E-Commerce Mass Data in the Cloud. IEEE Transactions on Systems, Man, and Cybernetics, 16(4):1020-1025, 2013.

[10] Modi H Y and Narvekar M. Enhancement of Online Web Recommendation System using a Hybrid Clustering and Pattern Matching Approach. IEEE Conference Nascent Technologies in the Engineering Field, pages 1-6, 2015.

[11] Wu X, Zhu X, Wu G Q and Ding W. Data Mining with Big Data. IEEE Transactions on Knowledge and Data Engineering, 26(1):97-107, 2014.

[12] Hu R, Dou W and Liu J. ClubCF: A Clustering-based Collaborative Filtering Approach for Big Data Application. IEEE Transactions on Knowledge and Data Engineering, 24(6):5-27, 2014.

[13] Rajaraman A and Ullman J D J. Mining of Massive Datasets. IEEE International Journal of Modern Physics, 21(10):1217-1227, 2010.

[14] Chen X, Zheng Z, Yu Q and Lyu M R. Web Service Recommendation via Exploiting Location and QoS Information. IEEE Transactions on Parallel and distributed systems, 25(7):1913-1924, 2014.

[15] Nalavade K and Meshram B B. Finding Frequent Itemsets using Apriori Algorithm to Detect Intrusions in Large Dataset. International Journal of Computer Applications, 2(6):68-84, 2014.