MOVIE RECOMMENDER SYSTEM USING MOVIELENS DATA

Sharmin Kantharia, M.S.
The George Washington University

May 5, 2021

ABSTRACT

Personalized recommendations are of key importance when it comes to increasing business value and sales of products and customer retention and satisfaction. Companies generate higher revenues and save resources, by simply understanding individual customer behaviors. In recent times, a large portion of the information we see online comes from recommendations catered to us based on content or products we have previously liked or invested in. Grouping similar users or items can allow for vast exposure of products to multiple users. It also solves the issue of the cold-start problem for new users. Recommender systems are mainly of 2 types – Content-based Filtering and Collaborative Filtering (Memory-based and Model-based). Using both Implicit and Explicit ratings, companies can recommend relevant content to their users. This paper aims to describe a movie recommender system that utilizes explicit ratings and movie genres to make personalized recommendations. Machine Learning methods including Term Frequency and Inverse Document Frequency (TF-IDF), Similarity Metrics – Cosine similarity, Matrix Factorization methods – Singular Value Decomposition and Clustering are implemented showcase how different the different types of recommender systems answer various questions and the differences between the approaches. The methods are evaluated using Root Mean Squared Errors (RMSE scores) and hit rates against recommended movies.

Keywords: recommender systems, machine learning, content-based filtering, collaborative filtering, TF-IDF, matrix factorization, clustering

INTRODUCTION

A large number of applications we use today are in a way customized to our need. From music to food to social media posts, recommender systems are responsible to cater to our preferences and allow us to find the best options during our search. Harvard Business Review describes recommender systems to be the single most algorithmic distinction between "born digital" enterprises and legacy companies.

Recommender Systems are a subclass of an information filtering system, aimed to predict either a "rating" or "preference" a user would give to an item, or a "recommendation" that the user may be given based on their past preferences. They are considered to be a renewable resource, that allow companies to follow a virtuous business cycle comprising of increased number of users, increased amount of data and better insights. Integrating personalization and recommendations, has help companies such as Netflix, Amazon and Spotify increase their profits and maintain customer satisfaction and intake. There are multiple use cases of recommending the most relevant content to users, including, relevant songs/podcasts (Spotify), relevant home listings (Airbnb, Zillow), relevant videos/movies/media (YouTube, Netflix, Amazon Prime Video, Instagram), relevant restaurants (Uber Eats) and even relevant users/jobs/posts (LinkedIn).

There are 4 main approaches to building a Recommender System:

1. Content-Based Filtering
2. Collaborative Filtering
    a. Memory-based approach – Item-based filtering, User-based filtering
    b. Model-based approach
3. Knowledge-Based System
4. Hybrid approach

At present recommender systems today, utilize the hybrid approach and implement Matrix Factorization algorithms. However, with recent research, the future of recommender systems is through the implementation of Deep Neural Networks, which not only derive latent features, but also can include time as a feature in the model, which is lack in a standard matrix factorization algorithm. These systems can be evaluated using Online and Offline evaluation methods. Offline evaluation consists of low-level metrics, such as RMSE scores used in the Netflix Prize Challenge. YouTube makes use of high precision and high recall for the various stages in their recommender system, as part of offline evaluation system. Online evaluations are generally high-level business metrics used to evaluate models after deployment, such as customer retention, click-through rate, and user engagement.

LITERATURE REVIEW

Collaborative filtering techniques are widely used for generating good quality recommendations, and methods can range from simple similarity metrics to advanced Deep Neural Networks. *Paper [2]* aims to compare the various methods and provide the best algorithms, for different situations. It can be concluded that Matrix Factorization methods, especially the widely used SVD, outperforms all other methods. This is in parallel to the observations made in this project when

implementing model-based approaches. Another interesting observation made in the paper suggests than to choose an appropriate model for the recommender system, it is crucial to find the best balance amongst the complex relationship between predictive accuracy, its variance, computation time and it is memory consumption. It was also observed that user-count and item-count are highly correlated to dependency on density.

Microsoft Recommenders is an open-source repository that provide Python utilities and Jupyter notebooks that help to accelerate the process of designing, evaluating, and deploying recommender systems. *Paper [3]* describes 4 main components of the repository starting with the reco_utils package, which consists of dataset, evaluation, and tuning utilities. It further lists some of the algorithms that can help implement both classical and deep learning-based recommender systems. A general workflow showing the implementation of building a system is laid-out as well.

A comparison of the 3 different collaborative filtering methods – memory-based, model-based and hybrid methods are explored in *Paper [4]*. Algorithms from each category are analyzed based on their predictive performance and their ability to address challenges such as data sparsity, scalability, privacy, etc. This project implements item-based and user-based top N recommendations along with clustering and dimensionality reduction algorithms such as SVD. In accordance with the paper, memory-based filtering allows for good scalability for co-rated items, however, cannot scale well for larger datasets. This method does not solve the cold start problem as well. Clustering algorithms allow for better scalability while dimensionality reduction reduces performance in case of sparsity and model-building can be expensive. Hybrid methods increase performance and complexity of the system.

Item-based collaborative filtering is a memory-based approach under collaborative filtering. The various implementations of item-based methods and their results are compared to a basic k-nearest neighbor approach in *Paper [5]* for the E-Commerce domain. The paper observes that item-based recommendations using adjusted cosine similarity were of better quality as it had the least Mean Average Error (MAE) score. The method was evaluated to have higher performance, due to the static nature of the items neighborhood. Predictions were more effective initially using regression than weighted-sum, however, due to overfitting regression has decreased performance. This project reflects these observations using cosine similarity and showcases that the item-based methods are faster as well.

The implementation of deep learning technology can help build richer recommender systems. According to *Paper [6]*, collaborative filtering is sequence predication problem, which projects as a Recurrent Neural Network (RNN). Using this approach, Long Short-Term Memory (LSTM) is implemented as a collaborative filtering method and compared to the standard clustering (user-based k-nearest neighbors) and matrix factorization (Markov Chain) methods. A vanilla LSTM model is compared to the standard static methods based on Short-term Prediction, recall, user coverage and Item coverage. Results indicate that the RNN outperforms the static methods for the MovieLens and Netflix datasets, by providing a higher short-term prediction and item coverage. It is suggested that modifying the objective function of the RNN can provide better results.

While collaborative filtering methods prevail current research, new algorithms and a hybrid approach to content-based methods can prove to be highly relevant in the future. *Paper [7]* explores

the various trends in content-based recommender systems. This approach mainly considers the past preferences of users and uses a feature-based representation of the items to learn a preference model. Compared to collaborative filtering, content-based approached were disadvantaged as the quality of a recommendation was not taken into consideration, however, they were beneficial in solving the cold-start problem. Recent trends suggest the use of Linked Open Data (external metadata), user-generated data (e.g., tags), meta-path-based approaches (meta-path connection between objects by a sequence of labeled features), meta encodings and deep learning.

Paper [8] helps to understand a deeper implementation of content-based filtering systems in the domain literature. It aimed to create a fast-acting recommender system that allowed to quickly scan scientific literature and find relevant and non-relevant documents. An open-source Python library called Science Concierge is created that implements this recommender system using Latent semantic Analysis to create scalable, vectorized documents. It also uses Rocchio Algorithm to find nearest neighbors using ball trees. The system was tested on 15K posters presented at the Society for Neuroscience 2015 conference. A complete set of similar papers to visit were recommended in 100ms. This recommender system outperformed the prior existing systems by making use of more complex data like abstracts rather than simple keywords, to generate recommendations. The performance of this method was evaluated using cross validation. Human-curated topic distances were used as performance metric.

Hybrid recommender systems combine individual approaches and help eliminate their drawbacks if any. *Paper [9]* describes a hybrid recommender system that combines rating, feature, and demographic information of the items. Some common disadvantages of individual systems include scalability, quality, sparsity, and cold-start problems. This paper proposes a hybrid approach called BoostedRDF, that generates accurate predictions and tackles the cold-start problem while outperforming 7 different individual systems. The BoostedRDF with regression gave the least MAE scores under the cold-start problem.

Evaluating recommender systems is a means to numerically understand the quality of predictions made by an algorithm. With new approaches to building recommender systems, it can be observed that evaluation is generally a comparison with baseline models. *Paper [10]* proposes that while prior research suggests that baselines underperform new approaches, when baselines are implemented correctly, they can outperform these new methods. It is concluded that a careful set of a standard matrix factorization method, e.g., SVD outperforms most recent approaches. SVD++ has shown to provide higher gain as well. Overall, standardized benchmarks and extensive tuning of hyperparameters can help baselines work more effectively.

DATA AND ANALYSIS

The MovieLens website utilizes collaborative filtering to provide movie recommendations to its users based on their preferences. It mainly considers movie ratings by its members and their reviews. GroupLens is a research lab formed for Social Computing Research at the University of Minnesota, specializing in recommender systems, online communities, digital libraries, and other such technologies. They are responsible for gathering and publishing rating data from the MovieLens website.

As recommended for education and development purposes, this project uses the MovieLens Latest Dataset which consists of 4 dataset files written in comma-separated values. Prior official permission to use the dataset for this project was obtained from GroupLens. The datasets, 'movies.csv' and ratings.csv' are merged and utilized for creating the recommender system. The final dataset explores 9742 unique movies, with over 100,000 ratings provided by 610 users. The minimum and maximum ratings given to a movie are 0.5 and 5.0, respectively. The average rating for the movies is 3.5. The movies span across 18 genres, which are used in both the filtering methods to generate recommendations.

RESEARCH METHODOLOGY

This project aims to create a recommender system and answer the following SMART questions:

1. Generate movie recommendations based on a particular Genre
2. Generate movie recommendations based on Movie Similarity
3. Generate movie recommendations based on User Similarity
4. Generate movie recommendations based on a user's Past Preferences

Data Pre-processing was implemented to understand the data, and merge the two main datasets, 'movies.csv' and 'ratings.csv'. The 'timestamp' column was removed from the final dataframe since it was not required to build the recommender system. The main columns used were 'movieId', 'userId', 'title', 'genres' and 'rating'. Exploratory Data Analysis (EDA) was performed to further understand the patterns in the main dataframe, by exploring rating distribution, movie titles and genres. The EDA also included a custom function that takes as input a particular genre and displays the Top 20 movies in that genre, which have the highest average rating and the maximum rating count. The recommender system was built in using the following Machine Learning (ML) techniques:

1. Content-based Filtering – Using TF_IDF and Cosine Similarity Metric
2. Collaborative Filtering
    a. Memory-based – Using Cosine Similarity
        - Item-based
        - User-based
    b. Model-based
        - Singular Value Decomposition
        - KNNBasic
        - KNNWithMeans

Recommendations generated using similarity metrics are evaluated using Hit Rate. The goal is to find the total number of 'hits' or movies previously rated by the user and check if it appears in the list of movies recommended. A higher hit rate implies there is enough data surrounding a user, allowing for better recommendations. The model-based system is evaluated using RMSE scores.

Python 3.8.5 and Jupyter Notebook (Anaconda) are used to build the main recommender system. Libraries including NumPy, Pandas, Matplotlib, Sklearn and Surprise are used for data cleaning, EDA and modeling and evaluation. The full code for this project can be found on GitHub.

KEY FINDINGS

EDA

The following plots help understand the data better. The line plot in *Figure 1* describes the rating distribution of each user. It can be observed that some users have given over 2000. *Figure 2* showcases the Top 15 users with the highest number of ratings given.

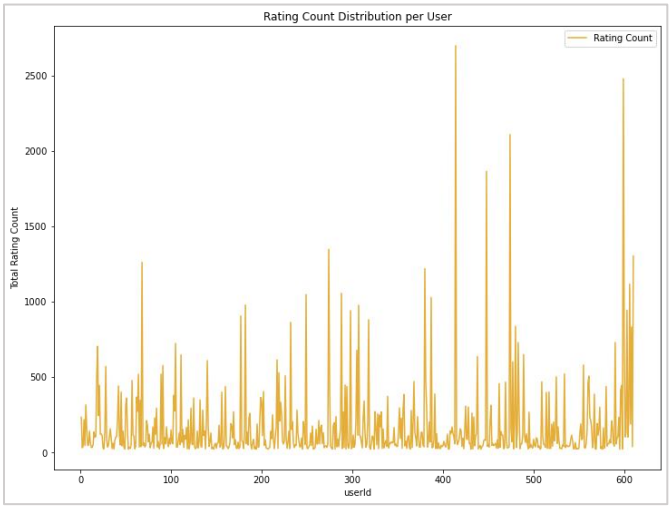

Figure 1 – User-Rating Count Distribution

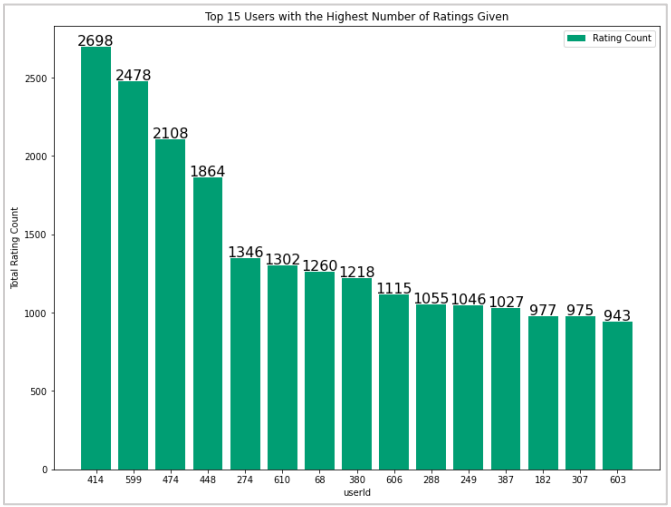

Figure 2 – Highest Rating Counts

The WordCloud in *Figure 3* shows the 18 different genres for this dataset. Drama, Comedy, Thriller, Action and Romance are 5 of the most popular genres. *Figure 4* displays a WordCloud for the words most used in the movie titles in this dataset.

Figure 3 – WordCloud of Genres

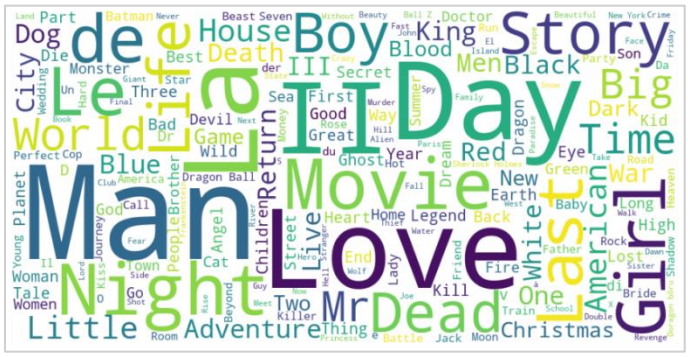

Figure 4 – WordCloud of Movie Titles

The bar plot in *Figure 5* displays the Top 20 movies which have received the highest average ratings and over 150 ratings. From the *Table 1*, Forrest Gump (1994) has the highest rating count of 329, with an average rating of 4.16. The Shawshank Redemption (1994) has the highest average rating of 4.42 with a rating count of 317.

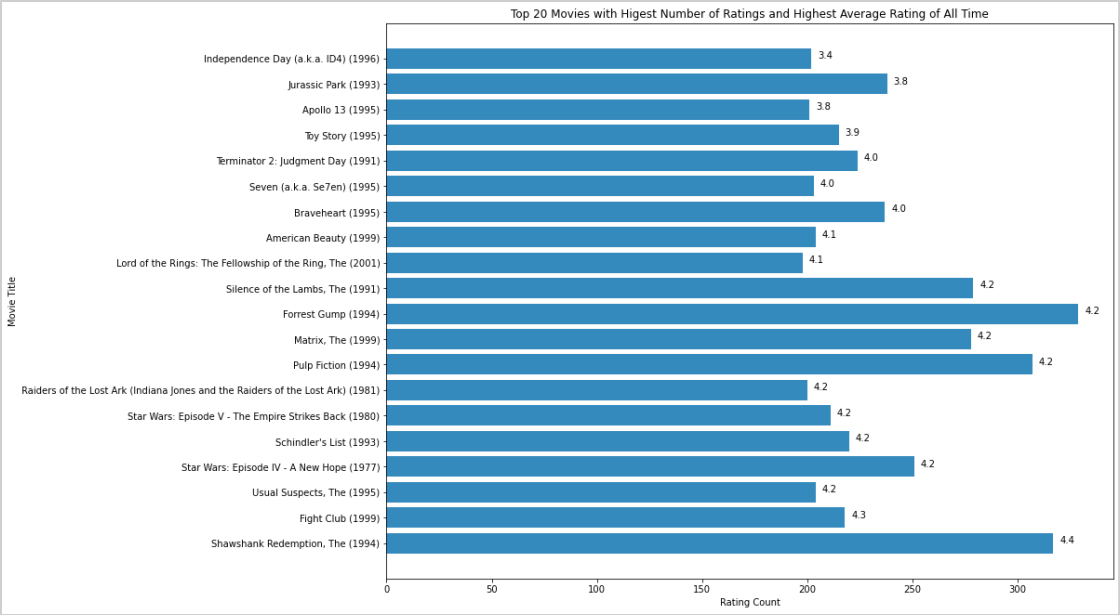

Figure 5 – Top 15 Movies with Highest Average Ratings with over 150 Ratings

	movieId	title	genres	rating count	average rating
0	318	Shawshank Redemption, The (1994)	Crime|Drama	317	4.429022
1	2959	Fight Club (1999)	Action|Crime|Drama|Thriller	218	4.272936
2	50	Usual Suspects, The (1995)	Crime|Mystery|Thriller	204	4.237745
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Sci-Fi	251	4.231076
4	527	Schindler's List (1993)	Drama|War	220	4.225000
5	1196	Star Wars: Episode V - The Empire Strikes Back...	Action|Adventure|Sci-Fi	211	4.215640
6	1198	Raiders of the Lost Ark (Indiana Jones and the...	Action|Adventure	200	4.207500
7	296	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller	307	4.197068
8	2571	Matrix, The (1999)	Action|Sci-Fi|Thriller	278	4.192446
9	356	Forrest Gump (1994)	Comedy|Drama|Romance|War	329	4.164134
10	593	Silence of the Lambs, The (1991)	Crime|Horror|Thriller	279	4.161290
11	4993	Lord of the Rings: The Fellowship of the Ring,...	Adventure|Fantasy	198	4.106061
12	2858	American Beauty (1999)	Drama|Romance	204	4.056373
13	110	Braveheart (1995)	Action|Drama|War	237	4.031646
14	47	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller	203	3.975369
15	589	Terminator 2: Judgment Day (1991)	Action|Sci-Fi	224	3.970982
16	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	215	3.920930
17	150	Apollo 13 (1995)	Adventure|Drama|IMAX	201	3.845771
18	480	Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller	238	3.750000
19	780	Independence Day (a.k.a. ID4) (1996)	Action|Adventure|Sci-Fi|Thriller	202	3.445545

Table 1 – Top 20 Movie by Genre

Content-based Filtering

This filtering method generates recommendations based on the idea of recommending similar items. We can use this method to either find items similar to a target item or find items a user would like based on their past preferences. This project aims to recommend movies that are similar to a target movie. TF-IDF *(Figure 6)* and Cosine similarity *(Figure 7)* are used to find similar movies based on the genre. The system takes as input the title of the target movie and returns a list of similar movies that

have the highest average rating. TF-IDF are used in information retrieval systems and determine the relative importance of documents (in this case movie genre) for a corpus. Term Frequency (TF) represents the frequency of each word in a document, in this case the frequency of the target genre in each remaining movie. Inverse Document Frequency represents how common or uncommon a word is in the entire corpus. Once the TF-IDF scores are calculated, a vector space model is created, where each vector represents a movie, and the axes represent genres. The cosine angles between the vectors are used to generate the similarity matrix and determine proximity. As the cosine angle decreases, the similarity value between two vectors (movies) increases indicating higher similarity.

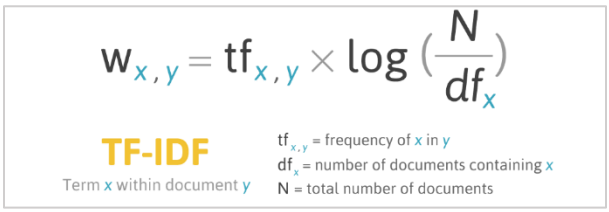

Figure 6 – TD-IDF Formula

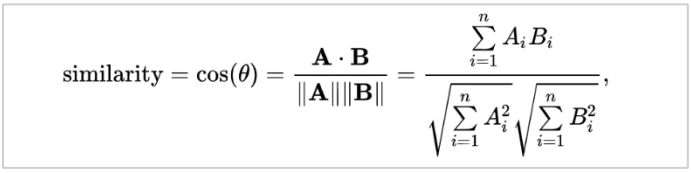

Figure 7 – Cosine Similarity Formula

TF-IDF vectorizer from the feature extraction module from Sklearn was used to find TD-IDF scores. The similarity matrix was created using linear kernel from sklearn.metrics, since TF-IDF vectorizer was used, simply calculating the dot product would give the similarity score. A custom function recommend_by_movie() uses the similarity score and manipulates existing dataframes and generates recommendations. The Top 20 recommendations *(Figure 8)* are selected based on high similarity score and sorted by highest average rating. The target movie 'Silence of the Lambs, The (1991)', had the genres Crime, Horror, Thriller. From the list of recommendations, it is observed that the movies had either all or a subset of the target genres. While some of the movies had lower average ratings, they were sorted in a descending order.

```
recommended_movies = movie_list_by_title('Silence of the Lambs, The (1991)', title, similar_movie, average_rating)
recommended_movies
```

Top 20 Movies Similar to Silence of the Lambs, The (1991) are:

	title	movieId	genres	rating
0	Usual Suspects, The (1995)	50	Crime|Mystery|Thriller	4.237745
1	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	32	Mystery|Sci-Fi|Thriller	3.983051
2	Seven (a.k.a. Se7en) (1995)	47	Mystery|Thriller	3.975369
3	Heat (1995)	6	Action|Crime|Thriller	3.946078
4	Casino (1995)	16	Crime|Drama	3.926829
5	Dead Man Walking (1995)	36	Crime|Drama	3.835821
6	Eye for an Eye (1996)	61	Drama|Thriller	3.750000
7	From Dusk Till Dawn (1996)	70	Action|Comedy|Horror|Thriller	3.509091
8	GoldenEye (1995)	10	Action|Adventure|Thriller	3.496212
9	Get Shorty (1995)	21	Comedy|Crime|Thriller	3.494382
10	Screamers (1995)	76	Action|Sci-Fi|Thriller	3.400000
11	To Die For (1995)	45	Comedy|Drama|Thriller	3.312500
12	Copycat (1995)	22	Crime|Drama|Horror|Mystery|Thriller	3.222222
13	Assassins (1995)	23	Action|Crime|Thriller	3.125000
14	Shanghai Triad (Yao a yao yao dao waipo qiao) ...	30	Crime|Drama	3.000000
15	Dead Presidents (1995)	42	Action|Crime|Drama	3.000000
16	Don't Be a Menace to South Central While Drink...	63	Comedy|Crime	2.714286
17	Money Train (1995)	20	Action|Comedy|Crime|Drama|Thriller	2.500000
18	Lawnmower Man 2: Beyond Cyberspace (1996)	66	Action|Sci-Fi|Thriller	2.500000
19	Dracula: Dead and Loving It (1995)	12	Comedy|Horror	2.421053

Figure 8 – Content-based Recommendation

The Hit Rate for content-based filtering is 0.48 *(Figure 9)*. This is a fairly decent score, since the data consists of explicit rating and hence, there is not enough information for the system to use to generate recommendations.

```
The Hit Rate for this method is: 0.4827586206896552
```

Figure 9 – Hit Rate for Content-based filtering

Collaborative Filtering

Memory-based collaborative filtering is of 2 types – Item-based and User-based Filtering. Item-based approach allows to find items (in this case movies) that are similar to each other using an Item Similarity Matrix, and the User-based approach creates a User Similarity Matrix, finds similar users, and then recommends movies to one user based on the preferences of another similar user. A rating matrix is created for each approach *(Figure 10, 11)* using the pivot_table() function from Pandas. Cosine similarity from sklearn.metrics.pairwise_distances is used to create the Item and User similarity matrix *(Figure 12, 13)*. The diagonals are filled with 0 since each item/user will have a similarity score of 1 with itself.

userId	1	2	3	4	5	6	7	8	9	10	...
0	4.0	0.0	0.0	0.0	4.0	0.0	4.5	0.0	0.0	0.0	...
1	0.0	0.0	0.0	0.0	0.0	4.0	0.0	4.0	0.0	0.0	...
2	4.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...
3	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	...
4	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...
...	...	...	...	...	...	...	...	...	...	...	...
9719	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
9720	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
9721	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
9722	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
9723	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

Figure 11 – Item-based Rating Matrix

movieId	1	2	3	4	5	6	7	8	9	10	...
0	4.0	0.0	4.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	...
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
4	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
...	...	...	...	...	...	...	...	...	...	...	...
605	2.5	0.0	0.0	0.0	0.0	0.0	2.5	0.0	0.0	0.0	...
606	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
607	2.5	2.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	...
608	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	...
609	5.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...

Figure 12 – User-based Rating Matrix

	0	1	2	3	4	5	6	7	8	9	...
0	0.000000	0.410562	0.296917	0.035573	0.308762	0.376316	0.277491	0.131629	0.232586	0.395573	...
1	0.410562	0.000000	0.282438	0.106415	0.287795	0.297009	0.228576	0.172498	0.044835	0.417693	...
2	0.296917	0.282438	0.000000	0.092406	0.417802	0.284257	0.402831	0.313434	0.304840	0.242954	...
3	0.035573	0.106415	0.092406	0.000000	0.188376	0.089685	0.275035	0.158022	0.000000	0.095598	...
4	0.308762	0.287795	0.417802	0.188376	0.000000	0.298969	0.474002	0.283523	0.335058	0.218061	...
...	...	...	...	...	...	...	...	...	...	...	...
9719	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
9720	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
9721	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
9722	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
9723	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.072542	...

9724 rows × 9724 columns

Figure 13 - Item (Movie) Similarity Matrix

	0	1	2	3	4	5	6	7	8	9	...
0	0.000000	0.027283	0.059720	0.194395	0.129080	0.128152	0.158744	0.136968	0.064263	0.016875	...
1	0.027283	0.000000	0.000000	0.003726	0.016614	0.025333	0.027585	0.027257	0.000000	0.067445	...
2	0.059720	0.000000	0.000000	0.002251	0.005020	0.003936	0.000000	0.004941	0.000000	0.000000	...
3	0.194395	0.003726	0.002251	0.000000	0.128659	0.088491	0.115120	0.062969	0.011361	0.031163	...
4	0.129080	0.016614	0.005020	0.128659	0.000000	0.300349	0.108342	0.429075	0.000000	0.030611	...
...	...	...	...	...	...	...	...	...	...	...	...
605	0.164191	0.028429	0.012993	0.200395	0.106435	0.102123	0.200035	0.099388	0.075898	0.088963	...
606	0.269389	0.012948	0.019247	0.131746	0.152866	0.162182	0.186114	0.185142	0.011844	0.010451	...
607	0.291097	0.046211	0.021128	0.149858	0.135535	0.178809	0.323541	0.187233	0.100435	0.077424	...
608	0.093572	0.027565	0.000000	0.032198	0.261232	0.214234	0.090840	0.423993	0.000000	0.021766	...
609	0.145321	0.102427	0.032119	0.107683	0.060792	0.052668	0.193219	0.078153	0.074399	0.121072	...

610 rows × 610 columns

Figure 14 - User Similarity Matrix

To generate item-based movie recommendations, the similarity scores of the movies similar to the target movie are used along with the ratings data. The target movie is excluded, and a rating threshold of 4.0 or greater is set. The top 20 movies are sorted by rating and displayed. The movieId 5 corresponds to the movie 'Father of the Bride Part II (1996)' and belongs to the Comedy genre. As shown in *Figure 15*, the recommendations have the highest ratings and mainly belong to the Comedy

genre. Item-based recommendations are evaluated using Hit Rate of genres. All genres from the target movie are checked to see if they are included in the genres of the predicted movies. A lower hit rate of 0.24 *(Figure 16)*, indicated that there is a lack of additional data for the targets (movies) and since the data included only explicit ratings, this issue can be resolved with the usage of implicit ratings as well.

```
recc1 = item_item_based_reccommendations(5, movie_similarity_matrix)
recc1
```

```
Top 20 Movies Similar to Father of the Bride Part II (1995) are:
```

	movieId	title	genres	userId	rating
0	4896	Harry Potter and the Sorcerer's Stone (a.k.a. ...	Adventure|Children|Fantasy	106	5.0
1	515	Remains of the Day, The (1993)	Drama|Romance	474	5.0
2	3814	Love and Death (1975)	Comedy	409	5.0
3	215	Before Sunrise (1995)	Drama|Romance	4	5.0
7	131610	Willy/Milly (1986)	Comedy|Fantasy	89	5.0
8	37384	Waiting... (2005)	Comedy	111	5.0
9	3022	General, The (1926)	Comedy|War	469	5.0
13	3388	Harry and the Hendersons (1987)	Children|Comedy	51	5.0
23	60126	Get Smart (2008)	Action|Comedy	98	5.0
24	53123	Once (2006)	Drama|Musical|Romance	483	5.0
26	2539	Analyze This (1999)	Comedy	453	5.0
27	4813	When Worlds Collide (1951)	Sci-Fi	186	5.0
28	4728	Rat Race (2001)	Comedy	275	5.0
29	3213	Batman: Mask of the Phantasm (1993)	Animation|Children	573	5.0
31	1131	Jean de Florette (1986)	Drama|Mystery	84	5.0
34	1753	Half Baked (1998)	Comedy	573	5.0
35	745	Wallace & Gromit: A Close Shave (1995)	Animation|Children|Comedy	376	5.0
36	6016	City of God (Cidade de Deus) (2002)	Action|Adventure|Crime|Drama|Thriller	326	5.0
37	3555	U-571 (2000)	Action|Thriller|War	115	5.0
38	714	Dead Man (1995)	Drama|Mystery|Western	246	5.0

Figure 15 - Item-based Recommendations

```
The Hit Rate for Item-based Collaborative Filtering is: 0.23809523809523808
```

Figure 16 - Hit Rate for Item-based filtering

To generate user-based movie recommendations *(Figure 18)*, the similarity scores of the users similar to the target user are used to find the most similar user *(Figure 17*). The highest-rated movies by the similar user are then sorted by ratings and the top-rated movies are displayed. According to the user similarity matrix, user 287 was the most similar to user 50, hence, past preferences of user 287 are now suggested to user 50 if they had not been previously rated by user 50. User-based recommendations are evaluated using Hit Rate of rated movies. A list of previously rated movies, by the user is compared to a list of recommended movies. Similar to the hit rate of the item-based approach, the hit rate of rated movies can be increased using implicit ratings. A hit rate of 0.0 implies that user 50 did not have any movies in common with user 287, however, they both watched movies from similar genres *(Figure 19).*

```
recc2 = user_user_based_reccommendations(50, user_user_similarity)
recc2
```

```
Top 10 Users Similar to User 50 are:
    similar user  similarity score
0            287          0.276549
1            607          0.263945
2            413          0.259135
3             63          0.257597
4            598          0.248365
5            273          0.247814
6            589          0.245804
7            468          0.244777
8             18          0.244546
9            479          0.239906
```

Figure 17 - Top 10 Similar Users

Top Movies Recommended to User 50 based on Similar Users are:

	movieId	title	genres	userId	rating
0	6711	Lost in Translation (2003)	Comedy|Drama|Romance	287	5.0
1	3949	Requiem for a Dream (2000)	Drama	287	5.0
2	608	Fargo (1996)	Comedy|Crime|Drama|Thriller	287	4.5
3	1921	Pi (1998)	Drama|Sci-Fi|Thriller	287	4.5
4	5618	Spirited Away (Sen to Chihiro no kamikakushi) ...	Adventure|Animation|Fantasy	287	4.5
5	1968	Breakfast Club, The (1985)	Comedy|Drama	287	4.5
6	2959	Fight Club (1999)	Action|Crime|Drama|Thriller	287	4.0
7	1136	Monty Python and the Holy Grail (1975)	Adventure|Comedy|Fantasy	287	4.0
8	8368	Harry Potter and the Prisoner of Azkaban (2004)	Adventure|Fantasy|IMAX	287	4.0
9	1682	Truman Show, The (1998)	Comedy|Drama|Sci-Fi	287	3.5
10	1732	Big Lebowski, The (1998)	Comedy|Crime	287	3.5
11	1208	Apocalypse Now (1979)	Action|Drama|War	287	3.5
12	1206	Clockwork Orange, A (1971)	Crime|Drama|Sci-Fi|Thriller	287	2.5
13	4306	Shrek (2001)	Adventure|Animation|Children|Comedy|Fantasy|Ro...	287	1.5
14	5816	Harry Potter and the Chamber of Secrets (2002)	Adventure|Fantasy	287	1.5
15	3034	Robin Hood (1973)	Adventure|Animation|Children|Comedy|Musical	287	1.0
16	1380	Grease (1978)	Comedy|Musical|Romance	287	0.5
17	1101	Top Gun (1986)	Action|Romance	287	0.5

Figure 18 - User-based Recommendations

```
The Hit Rate for User-based Collaborative Filtering is: 0.0
```

Figure 19 - Hit Rate for User-based filtering

Model-based collaborative filtering was performed using a Matrix Factorization algorithm, SVD and KNN Clustering algorithms, KNNBasic and KNNWithMeans, using functions from the Surprise Library in Python. SVD *(Figure 20)* is a dimensionality reduction technique that uses matrix factorization to reduce the number of features by reducing the space dimension from N-dimension to K-dimension (K<N). It is used as a collaborative filtering method and uses a rating matrix where the rows represent uses, columns represent items (movies) and the matrix is filled with ratings. From *Figure 21*, M is a high-level (mxn) user-item-rating matrix, U is a (mxn) orthogonal left singular matrix representing the relationship between users and latent features, d is a (nnx) diagonal matrix describing the strength of

each latent feature and V^T is a (nxn) diagonal right singular matrix indicating the similarity between items and latent features.

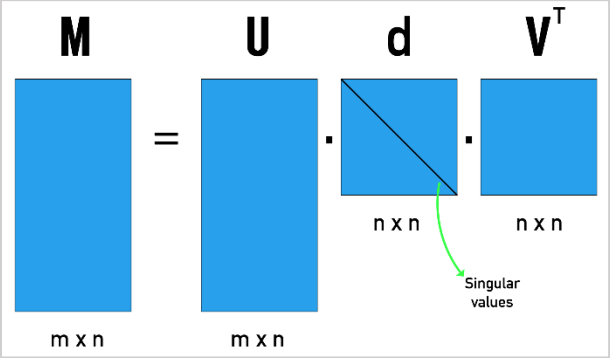

Figure 20 - SVD Representation

The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

Figure 21 - SVD using Surprise Library

KNNBasic is a basic collaborative filtering algorithm and is derived from the KNN approach. It uses the prediction formulae are shown in *Figure 22*. KNNWithMeans is similar to KNNBasic, but it also considers the mean ratings given by each user. It uses the prediction formulae shown in *Figure 23*.

The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

or

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} \text{sim}(i,j) \cdot r_{uj}}{\sum_{j \in N_u^k(i)} \text{sim}(i,j)}$$

Figure 22 - KNNBasic using Surprise Library

The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

or

$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} \text{sim}(i,j) \cdot (r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} \text{sim}(i,j)}$$

Figure 23 - KNNWithMeans using Surprise Library

Using the in-built Reader() and Dataset() functions, the ratings data was read. The Reader class is especially built to parse files containing ratings. To find the best model, a for loop was used to loop over the algorithms and a 3-fold cross validation was performed. Models were evaluated based on RMSE scores, Fit time, and Test time. From the results *(Figure 24)*, SVD was the best performing model with the least RMSE scores and least runtime on the test set. Hence, it was further used to generate the recommendations.

Algorithm	test_rmse	fit_time	test_time
SVD	0.882095	9.345310	0.598739
KNNWithMeans	0.904936	0.411218	6.296728
KNNBasic	0.957798	0.335956	5.277552

Figure 24 - 3-Fold Cross Validation Results

To generate recommendations using SVD, a userid is taken as input, estimated ratings were predicted using which recommendations were generated. The custom function initially drops all the ratings made by the user, so as to recommend new movies. Using the build_full_trainset() function, the training set was created and fit to the algorithm. The using the remaining data, estimated ratings of unseen movies by the user were predicted. A threshold of estimated ratings and original ratings greater than or equal 4.0 was set and the recommendations were displayed. In *Figure 25*, 'rating' represents the original rating of the movie by another user while 'estimated rating' represents the ratings that can be given by user 435 for each movie.

```
recommend_by_model(SVD, 435, ratings_data, ratings, data)
```

```
Recommended Movies for user 435 are:
```

	movieId	title	genres	userId	rating	estimated rating
0	858	Godfather, The (1972)	Crime|Drama	590	5.0	5.000000
1	1204	Lawrence of Arabia (1962)	Adventure|Drama|War	477	4.0	4.948049
2	1247	Graduate, The (1967)	Comedy|Drama|Romance	285	5.0	4.943887
3	1272	Patton (1970)	Drama|War	202	5.0	4.907382
4	1278	Young Frankenstein (1974)	Comedy|Fantasy	31	5.0	4.896940
5	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Sci-Fi	323	4.5	4.888398
6	1104	Streetcar Named Desire, A (1951)	Drama	414	4.0	4.873958
7	1208	Apocalypse Now (1979)	Action|Drama|War	57	4.0	4.871924
8	7153	Lord of the Rings: The Return of the King, The...	Action|Adventure|Drama|Fantasy	106	5.0	4.871781
9	750	Dr. Strangelove or: How I Learned to Stop Worr...	Comedy|War	348	4.5	4.870719
10	913	Maltese Falcon, The (1941)	Film-Noir|Mystery	23	4.0	4.865881
11	48516	Departed, The (2006)	Crime|Drama|Thriller	477	5.0	4.861472
12	1178	Paths of Glory (1957)	Drama|War	391	5.0	4.858685
13	912	Casablanca (1942)	Drama|Romance	563	4.0	4.857937
14	922	Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	Drama|Film-Noir|Romance	156	4.0	4.854643
15	71899	Mary and Max (2009)	Animation|Comedy|Drama	610	4.5	4.852360
16	31658	Howl's Moving Castle (Hauru no ugoku shiro) (2...	Adventure|Animation|Fantasy|Romance	274	4.0	4.839326
17	112552	Whiplash (2014)	Drama	212	4.5	4.836478
18	111	Taxi Driver (1976)	Crime|Drama|Thriller	532	4.0	4.834983
19	1276	Cool Hand Luke (1967)	Drama	140	4.0	4.826056

Figure 25 - Recommendations generated by SVD Model

CONCLUSION

Recommender Systems offer personalization and help in increased profits for companies and also allow users to find the most relevant products for their use. The goal of the project to achieve a high-level understanding of the various filtering methods and their subsequent technologies was accomplished. The literature survey explored the various techniques and their advantages and disadvantages which further guides in selecting better models and evaluation metrics depending on the data. Implementing content-based filtering allowed to solve the cold-start problem that posed as a restriction for collaborative filtering. However, Faster and efficient results were achieved using model-based collaborative filtering, compared to both memory-based methods and content-based filtering. The project also posed certain challenges including finding proper evaluation methods/metrics for Top-N recommendations, however, hit rate was used similar to precision metric. A hybrid approach and implementation of deep learning techniques can further improve the accuracy and efficiency of recommender systems.

FUTURE WORK

This project covers the understanding and implementation of the basic techniques used to build a simple recommender system. Recent research suggests that using hybrid models, deep learning technologies such as RNNs and extensive hyperparameter tuning of existing models can increase the scalability, performance, and efficiency of these systems. New hybrid technologies may also solve existing issues with individual methods. Furthermore, evaluation of the Top-N recommendations gave lower hit rates implying the lack of implicit ratings and meta data on movies and users. Hence, larger amount of detailed data can help provide even more personalized recommendations.

BIOGRAPHY

Sharmin Kantharia an ambitious graduate student in the Data Science Master's program at George Washington University, with a keen interest in Machine Learning. Having completed several projects using Python and R programming, spanning Machine Learning, Deep Learning, Natural Language Processing and Time Series Analysis, she is always up for a challenge to explore new technologies. Driven by her passion to benefit the community, she aims implement her knowledge and utilize her experiences in projects that enrich others. In her spare time, she enjoys practicing yoga, cooking and reading.

BIBLIOGRAPHY

[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872

[2] Lee, Joonseok & Sun, Mingxuan & Lebanon, Guy. (2012). A Comparative Study of Collaborative Filtering Algorithms. https://arxiv.org/abs/1205.3193

[3] Scott Graham, Jun-Ki Min, and Tao Wu. 2019. Microsoft recommenders: tools to accelerate developing recommender systems. In Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19). Association for Computing Machinery, New York, NY, USA, 542–543. DOI: https://doi.org/10.1145/3298689.3346967

[4] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A survey of collaborative filtering techniques. Adv. in Artif. Intell. 2009, Article 4 (January 2009), 1 pages. DOI: https://doi.org/10.1155/2009/421425

[5] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (WWW '01). Association for Computing Machinery, New York, NY, USA, 285–295. DOI: https://doi.org/10.1145/371920.372071

[6] Devooght, Robin & Bersini, Hugues. (2016). Collaborative Filtering with Recurrent Neural Networks. https://arxiv.org/abs/1608.07400v2

[7] Lops, P., Jannach, D., Musto, C. *et al.* Trends in content-based recommendation. *User Model User-Adap Inter* **29,** 239–249 (2019). https://doi.org/10.1007/s11257-019-09231-w

[8] Achakulvisut, Titipat & Acuna, Daniel & Ruangrong, Tulakan & Kording, Konrad. (2016). Science Concierge: A Fast Content-Based Recommendation System for Scientific Publications. https://arxiv.org/abs/1604.01070v2

[9] Mustansar Ali Ghazanfar and Adam Prugel-Bennett. 2010. A Scalable, Accurate Hybrid Recommender System. In Proceedings of the 2010 Third International Conference on Knowledge Discovery and Data Mining (WKDD '10). IEEE Computer Society, USA, 94–98. DOI: https://doi.org/10.1109/WKDD.2010.117

[10] Rendle, Steffen & Zhang, Li & Koren, Yehuda. (2019). On the Difficulty of Evaluating Baselines: A Study on Recommender Systems. https://arxiv.org/abs/1905.01395v1