Movie Recommendation System

Omkar Sandeep Vedak
(1217359169)
Arizona State University
Tempe, USA
ovedak@asu.edu

Vedant Milind Salvi
(1217678657)
Arizona State University
Tempe, USA
vsalvi@asu.edu

Anoop Reddy Vutukuri
(1217182122)
Arizona State University
Tempe, USA
avutuku2@asu.edu

Arun Vignesh Malakkan
(1215098183)
Arizona State University
Tempe, USA
amalarkk@asu.edu

Dia Dutta
(1215140186)
Arizona State University
Tempe, USA
ddutta8@asu.edu

Shrimangal Rewagad
(1217439366)
Arizona State University
Tempe, USA
srewaga1@asu.edu

***Abstract*—Recommender systems have become an integral part of most websites like Amazon, Netflix, and YouTube. Nowadays users are flooded with choices so providing recommendations to match the needs of the user can help to improve the user experience immensely. The recommendation system tries to learn inherent patterns and relations from the data to recommend user what else they might like. In this project, we would be designing a movie recommendation system by implementing some popular content-based filtering, collaborative filtering, and a hybrid model. We would then evaluate these three types of algorithms using appropriate metrics to see which algorithm gives the most accurate result. Additionally, we will also be addressing common problems that the recommender system faces such as cold start problems and data-sparsity problems.**

Keywords- Recommendation System, Collaborative Filtering, Restricted Boltzmann Machine, Matrix Factorization, Nearest Neighbor, MovieLens Dataset.

I. Introduction

The entertainment industry has been booming over the past few years due to the widespread use of the internet. Movie streaming has become a part and parcel of life. There are a wide variety of movies available on these streaming services such as comedy, thriller, horror, animation, action, etc. Due to the ever-increasing media availability, modern users are getting overwhelmed with choices. There are a plethora of movies available on the internet, making it impossible for a human to parse through all of them to find the movies of their liking. Hence came the necessity for recommender systems that takes user's preferences into account and match it with similar users, the movie descriptions, etc., to suggest a movie as per his/her liking.

Streaming services such as Netflix are providing a huge selection of content that could meet the different tastes and needs of its users. It is important that such services provide appropriate content to the right users to enhance user satisfaction. Therefore, the recommender system plays a critical role in the success of these services which provide a personalized recommendation to users that suits the user's taste.

Recommender systems are beneficial for both the customers as well as the service providers. It reduces the search time required by users to find the content they like and the products which the users would have missed to buy otherwise, thus decreasing user anxiety. This also improves user satisfaction and thus increases the purchase probability of the user and user retention rates. Recommender systems have come a long way and are currently very complex and well developed that people claim that the recommender systems can suggest better than their friends and some systems claim that they know you better than yourself.

II. Problem Statement

In this project, our goal is to implement various algorithms for the movie recommendation system and analyze the results. We have used 'the MovieLens dataset' [1] and 'The Movies Dataset' [15] in this project. We have chosen to implement various collaborative filtering models such as Restricted Boltzmann Machine, User-based filtering, Item-based filtering, Stochastic Gradient Descent, and Alternating Least Square methods. We have also implemented a hybrid model which is a combination of Content-based filtering and Restricted Boltzmann Machine. By evaluating the results, we can understand the advantages and disadvantages of each method hence giving a better understanding of the use cases of the respective algorithms. Some of the notable websites using collaborative filtering are Hulu, Amazon, and Youtube. Content-based filtering is used by Amazon and Pandora. Hybrid models are used by Netflix, Linkedin, and Youtube, this one of the reasons for having one of the best recommendation systems in the market. We will be establishing and explaining that hybrid models are better than the Content-based filtering and Collaborative filtering models considered individually with results and reasoning. We will also be addressing the common issues faced by recommender systems such as the cold start problem, Data Sparsity, etc and also which algorithms to use or avoid dealing with these issues efficiently.

III. Related Works

There are mainly two types of recommendation systems that are used widely, the first one is content-based filtering and the other one is collaborative filtering. There are also various hybrid approaches available. In collaborative filtering users similar to that user are found and recommendations are made according to these users. Collaborative filtering can also be divided into two types Memory based and Model-based. The memory-based recommendation system is simpler, and they also can be divided into two types the first one is user-based collaborative filtering and the second one is item-based [6]. In user-based Collaborative filtering, similar users are found, and recommendations are made based on that. In item-based collaborative filtering is based on finding similar items based on the ratings of users [18].

Model-based collaborative filtering works best as they handle the cold start problem and data sparsity. Here we have to use different machine learning algorithms like decision trees so that we can recommend movies that have not been rated yet also we do dimensionality reduction so that features which are not important are ignored which in turn increases the accuracy of the system. Matrix factorization techniques [4] are also very popular nowadays and the most used technique is the Singular Value Decomposition. The SVD decomposes the user-item matrix into eigenvalues and eigenvectors and the preferences of the users can be found from that. There are few disadvantages of the model they give less accurate results if there are missing value and also the complexity is high so there is also an alternative to this which is known as Simon Funk's SVD which only consider the non-missing values only. There are also deep learning models available that are more accurate and where the hidden layer is the reduced features that can be obtained from weights and biases.

In Content-based collaborative filtering Users and items have their characteristics and based on these characteristics' recommendations are provided. It is based on the content like the genre, actor and hence the recommendations are very similar to each other and do not consider the preferences of the user. A very common approach is to provide exact matching, term frequency can be used for putting more weightage to words which are more important than other words [17]. A supervised model can also be build based on where the output will be whether the movies should be recommended or not.

Association Rule Mining is used for recommending movies as well. It creates a graph where the items which are more frequently used are connected and hence, they form clusters and they should have minimum support and confidence levels. Having high confidence means they provide accurate results. But the disadvantage of this algorithm is they are not very scalable [19]. There is an item-centered Bayesian classifier that takes the user features as input instead of item features. We then compute the ratio between the probability of the user to like or not to dislike. Here it is assumed that the likelihoods are Gaussian distributed, and this can be used to create models like naïve Bayes classifier. They are also many hybrid models that take deep learning models and collaborative and content-based filtering and more accurate than the others.

IV. System Architecture & Algorithms

A. Architecture Design

Figure 1. represents the system architecture for the Movie Recommender system. There are three types of recommenders in the system namely, content-based recommender, collaborative-filtering recommenders, and a hybrid recommender. Two types of datasets were used in the system. We used 'The MovieLens' dataset [1] to fetch ratings used to compute collaborative filtering results. We also used 'The Movies Dataset' [15] to extract keywords, cast, and crew information for the movies listed in 'The MovieLens' dataset to compute content-based recommendations.

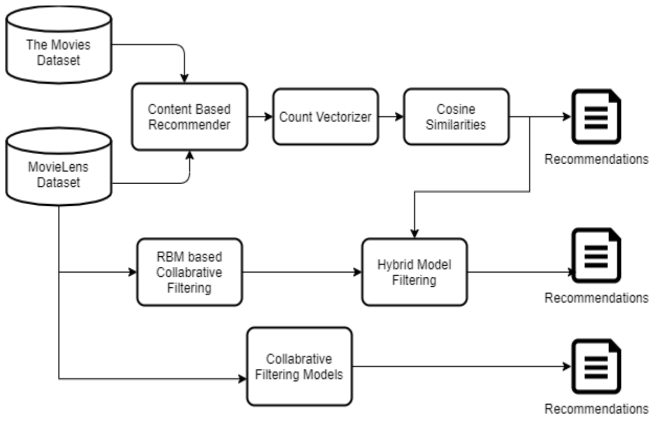


Figure 1: System Architecture

The content-based recommender uses the metadata from the 'The Movies Dataset' to create a metadata dump. This meta dump is passed on to the count vectorizer to create a count matrix. The count vectorizer uses unigrams and bigrams to the create-count matrix. In the next step, we calculate the cosine similarities and return the most similar movies. The cosine similarity is calculated as [16]:

$$\mathrm{cosine}(x, y) = \frac{x.y^T}{\| x\|.\|y\|}$$

Where x and y represent the count matrix.

The Restricted Boltzmann Machine (RBM)[13] is another model we used to implement collaborative filtering. RBMs can learn latent features from the input data that cannot be inferred directly. These give them the capability to reconstruct the input by trying to learn the probability distribution of the input with a reasonable approximation. We also build a hybrid recommender by filtering the results obtained from the RBM with those obtained from the content-based recommender. Then we sorted the recommendations based on the scores obtained from the RBM to output the top 10 recommendations.

We also implemented some other collaborative filtering models that accept ratings from the 'The MovieLens' dataset and apply various collaborative filtering algorithms described below.

B. Algorithms

1) Content-Based Filtering: Content-based methods create a profile for each movie to classify them on their nature and then suggest the items by matching with the item profile. For example, a movie profile could include attributes such as genre, actors, director, language, popularity, etc. The algorithm matches the movie profile with the profile of the movie that has been entered, if there is a suitable find then it recommends the movie to the user.

2) Collaborative Filtering: Restricted Boltzmann Machines (RBM) [11] is one of the types of deep learning models that we used for collaborative filtering due to its input reconstruction property. RBM is a special type of two-layered generative artificial neural network. It is also a specific type of Boltzmann machine because it is restricted in terms of connections between hidden and visible nodes. It has two layers of nodes visible and hidden which are connected by a fully bipartite graph. No two nodes in the same layer are connected.

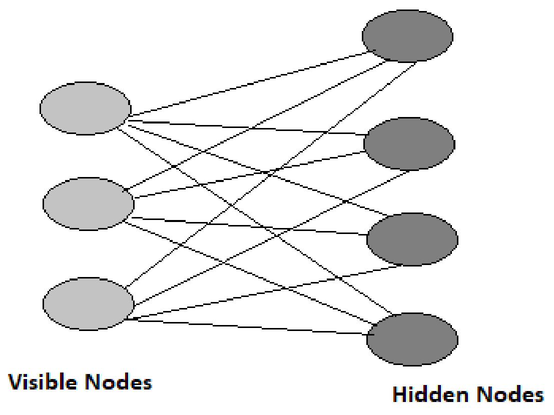


Figure 2: Restricted Boltzmann Machine [14]

The main reason we used RBMs as one of the methods in our recommendation is because of its ability to reconstruct back user input from the hidden layer output. RBMs learn patterns in input data to do this [13]. This helps it to create a lower-dimensional representation of the pattern which is further used to recreate approximations of the original user input. As in a recommendation system, we need the algorithm to find a pattern like (the movie preference of the user) and reconstruct this in a form of rating for each movie, this makes RBMs ideal solution to our problem. This will help us in recommending the movies to the user based on the reconstructed movie ratings.

We also implemented memory-based collaborative filtering algorithms that build a model that can predict items that a user might be interested in based on the user's past behavior such as numerical rating given to an item as well as behaviors of other similar users. There are two main types of collaborative filtering models which are Memory based and Model-based. Furthermore, the model-based approach is divided into two types:

a) User-based: This approach assumes that similar users will be interested in similar items. The idea is to find users who are similar to target users and recommend movies based on the likes of similar users. A similarity measure between users needs to be defined before we cluster similar users.

b) Item-based: This approach assumes that the two movies are similar if they receive similar ratings from a user. It will predict movies to the target user by calculating the weighted average of rating on most similar items. The similarity between items defines based on cosine similarity, correlation-based similarity, or Jaccard distance.

Most of the memory-based techniques collaborative filtering produce very good results but have the problem with the cold start problem [3] which is when the system is created or restarted. Another alternative is to use the singular value Decomposition technique but it does not work very well when the matrix is sparse and as we know that the user-item matrix will be very sparse as not every user may have seen all the movies present in the database. So, matrix factorization techniques which tend to reduce the regularized square error have become increasingly popular. Two such techniques exist which are Stochastic Gradient Descent and Alternating least squares. We implemented both the methods to see which produces more accurate results.

a) Gradient Descent Algorithm: This approach is used to find the lowest point of the cost function but that leads to a lot of randomness and it is very slow when we are using large amounts of data. Stochastic gradient descent reduces this randomness by choosing one data point from the whole dataset instead of the whole dataset at each iteration and then finding the parameter values that can minimize the cost function [2].

b) Alternating Least Square: We choose one of the features either the user or the item constant. For example, if we take the user to be constant, then we take the derivative of the loss function concerning the item vector, and then we set it to zero in order to find the minimum of the cost function. Now make the new item vectors constant and find the derivative with respect to the user which is the previously constant vector. We go on alternating till it converges [9].

3) Hybrid Model: While both the Content-based methods and Collaborative filtering provide good recommendations, they have drawbacks that can be eliminated by combining them and developing a hybrid model. [16] describes a method using which we can create a hybrid model. We combined a content-based model with RBM observed improved results. The recommendations seemed more relevant and diverse, unlike the individual algorithms which were actor or director centric.

V. Datasets

A. *Descriptions*

We used the following two datasets to train our models:

1) The MovieLens Dataset: [1] We obtained the dataset from MovieLens. MovieLens is a web-based recommender system and community that recommends movies to its users based on their movie preferences. It uses collaborative filtering techniques based on movie ratings and reviews. We are using MovieLens 20M Dataset which is a combination of 20 million ratings on 27,000 movies by 138,000 users. There are three primary files in this dataset namely:

a) ‘ratings.csv’: This file contains four columns namely, UserID, MovieID, Rating, TimeStamp.
b) movies.csv’: This file contains three columns: MovieID, Title, Genre.
c) ‘tags.csv’: This file contains three columns UserID, MovieID, Tag, TimeStamp

2) The Movies Dataset: [15] This dataset contains meta-data like keywords, credits, cast, budget, etc. about the movies listed in the MovieLens dataset. There are four primary files in this dataset namely:

a) ‘credits.csv’: This file contains three columns namely, cast, crew, id.

b) ‘keywords.csv’: This file contains two columns: id, keywords.

c) ‘links_small.csv’: This file contains three columns movieId, imdbId, tmbId

d) ‘ratings_small.csv’: This file contains four columns userId, movieId, rating and timestamp

B. *Data Insights*

From the initial analysis of the data, we found out that it is very sparse. In figure 3, we can see that the distribution of ratings follows the long-tail property. According to this property, there are only a small number of movies that are rated frequently. These movies are the most popular. Most of the movies are rated rarely. This causes the skewed distribution of ratings.

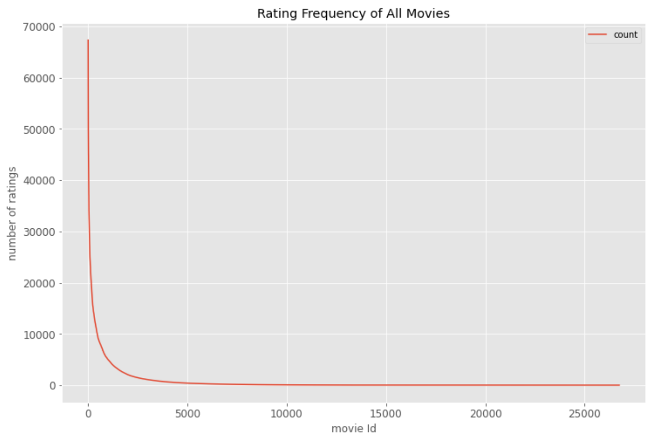


Figure 3: Rating Frequency of all movies

In figure 4, we can see that the distribution of the ratings by users is very similar to the ratings for the movies. It also follows the long-tail property. There are only a few users who are actively rating movies that they watched. Most of the users are not interested in rating the movies.

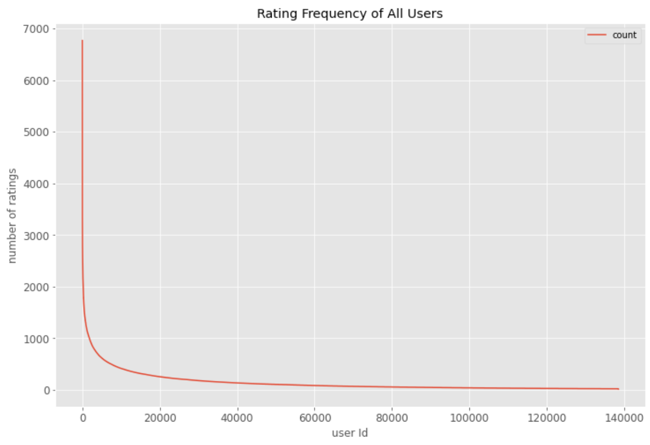


Figure 4: Rating Frequency of all users

C. *Preprocessing Steps*

Depending on the type of recommendation system we used different preprocessing steps:

1) *Collaborative Filtering*
   a. Filter out a set of ratings due to processing limitations.
   b. In the case of Item-based filtering, select the top 25% movies which are highly rated and limit the number of users to the top 40%.
   c. Normalize the rating values in the range of [0,1]
   d. Create a user movie matrix for the ratings given by each user.

2) *Content-Based Filtering*
   a. Create a meta-data dump using the director name, top 3 cast members, keywords from the description.
   b. Remove the spaces between the names and converted them to the lower case.
   c. Apply count vectorizer to create count matrix which extracts unigrams and bigrams

VI. Evaluations

A. *Metrics*

For evaluating the matrix factorization using Stochastic Gradient Descent, Alternating Least Square, and Restricted Boltzman Machine we use the Root Mean Square Error (RMSE) method. RMSE measures the error of a model. In RMSE we determine the difference between the actual values and predicted values and take the average of the squared difference. RMSE expresses average model prediction error in units of the variable of interest.

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

The table below shows the RMSE values for the algorithms mentioned above:

Algorithm	RMSE
Matrix Factorization – SGD	4.899511084129212
Matrix Factorization – ALS	0.9009418
Restricted Boltzmann Machine	0.009865913

For User-item based matching, we follow the similarity matching between two random users according to the rating matrix.

B. *Experiments*

- We created a hybrid model by combining Content-Based Model with RBM based Collaborative Filtering model
- The hybrid model filters the RBM results with the results obtained from content-based recommendations.
- The Alternating Least Squares algorithm reduces the popularity bias, cold start problems up to a threshold in the recommender systems.

- The ALS algorithm is implemented in PySpark and so it runs parallelly over a cluster of machines processing a huge dataset and thereby solving the scalability issue.
- A list of recommendations is given based on the new user's favorite movie.

C. Findings

1) Matrix Factorization – SGD

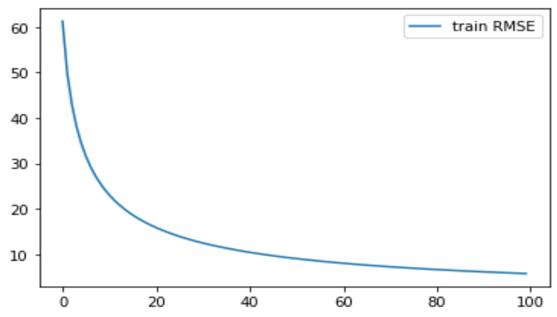
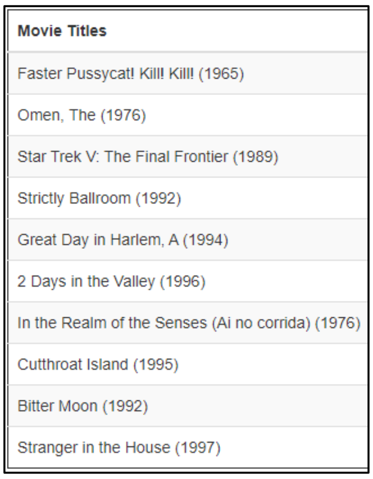


Figure 5: RMSE for Matrix Factorization using SGD

In every epoch weights and biases are getting re-calculated, so prediction is changing and RMSE is getting decreased with the number of an epoch.

Epochs	RMSE
10	25.430141013921105
20	16.976909447280637
30	13.273099137634212
100	6.919882648000294
200	4.899511084129212



Movie Titles
Faster Pussycat! Kill! Kill! (1965)
Omen, The (1976)
Star Trek V: The Final Frontier (1989)
Strictly Ballroom (1992)
Great Day in Harlem, A (1994)
2 Days in the Valley (1996)
In the Realm of the Senses (Ai no corrida) (1976)
Cutthroat Island (1995)
Bitter Moon (1992)
Stranger in the House (1997)

Figure 6: Recommendations using Matrix Factorization (SGD) for user Id: 21

2) Matrix Factorization – ALS

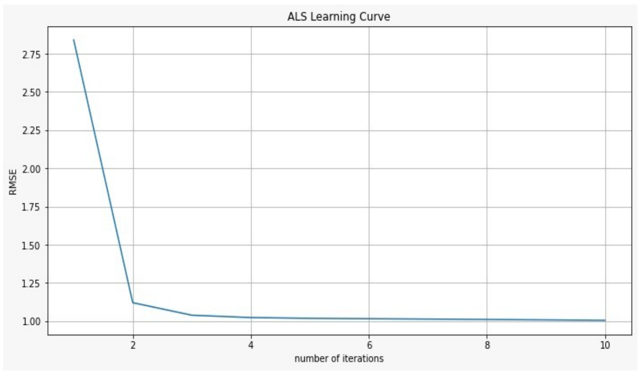
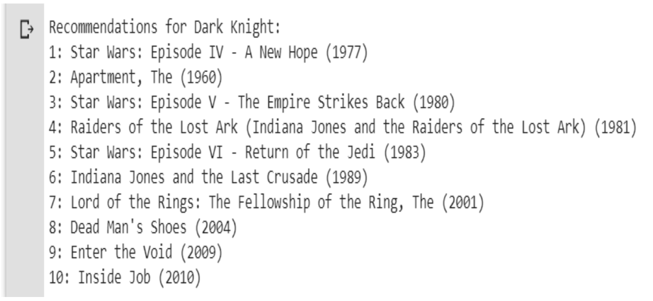


Figure 7: RMSE for Matrix Factorization using ALS

In the ALS method, we can notice that the RMSE value falls drastically with a slight increase in the number of Iterations.

Epoch	RMSE
10	1.1997655
20	0.9138942
30	0.9009418



```
Recommendations for Dark Knight:
1: Star Wars: Episode IV - A New Hope (1977)
2: Apartment, The (1960)
3: Star Wars: Episode V - The Empire Strikes Back (1980)
4: Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
5: Star Wars: Episode VI - Return of the Jedi (1983)
6: Indiana Jones and the Last Crusade (1989)
7: Lord of the Rings: The Fellowship of the Ring, The (2001)
8: Dead Man's Shoes (2004)
9: Enter the Void (2009)
10: Inside Job (2010)
```

Figure 8: Recommendations using Matrix Factorization (ALS) for a new user.

3) Restricted Boltzmann Machine

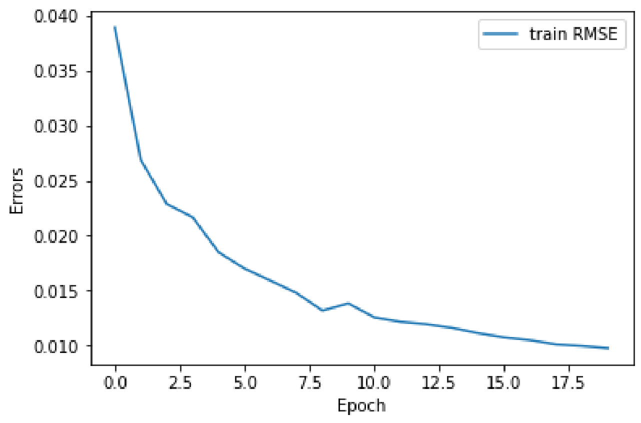
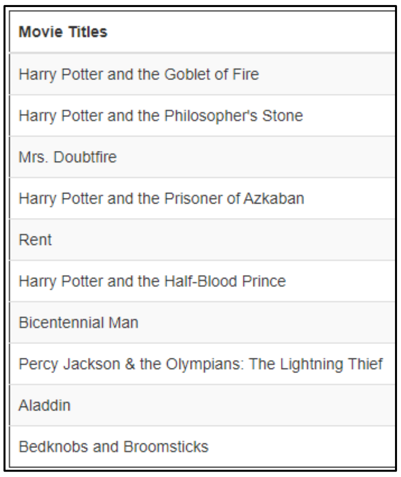


Figure 9: RMSE for Restricted Boltzmann Machine

Here, the algorithm starts to converge after the first 10 epochs.

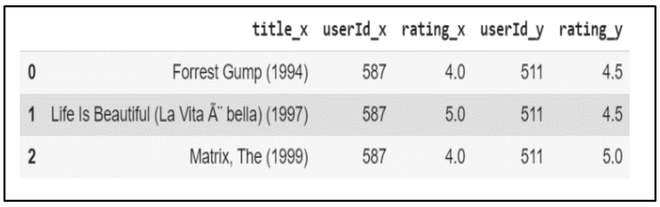


Movie Titles
Harry Potter and the Goblet of Fire
Harry Potter and the Philosopher's Stone
Mrs. Doubtfire
Harry Potter and the Prisoner of Azkaban
Rent
Harry Potter and the Half-Blood Prince
Bicentennial Man
Percy Jackson & the Olympians: The Lightning Thief
Aladdin
Bedknobs and Broomsticks

Figure 10: Recommendations using Restricted Boltzmann Machines for user Id : 4 and movie Name: Harry Potter and the Chamber of Secrets

4) *User-based matching*

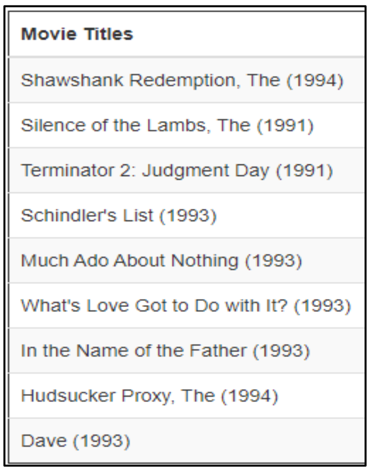
Here, we match the movies between 2 users and detect which movies are the same according to the ratings of each movie.



	title_x	userId_x	rating_x	userId_y	rating_y
0	Forrest Gump (1994)	587	4.0	511	4.5
1	Life Is Beautiful (La Vita Ã¨ bella) (1997)	587	5.0	511	4.5
2	Matrix, The (1999)	587	4.0	511	5.0

Figure 11: User Similarity Matrix

From the above Fig 11, we can notice that when we compare two users, we get common movies that are approximately rated the same by each user.

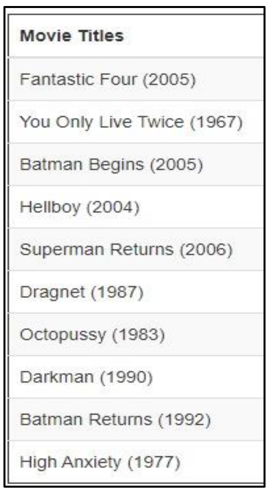


Movie Titles
Shawshank Redemption, The (1994)
Silence of the Lambs, The (1991)
Terminator 2: Judgment Day (1991)
Schindler's List (1993)
Much Ado About Nothing (1993)
What's Love Got to Do with It? (1993)
In the Name of the Father (1993)
Hudsucker Proxy, The (1994)
Dave (1993)

Figure 12: Recommendations using User-Based Matching for user Id: 21

5) *Item Based Collaborative filtering*

This algorithm uses cosine similarity to find a list of movies that are similar to the given movie based on the user-movie matrix. The recommendations for the movie "Batman" gives the following results.

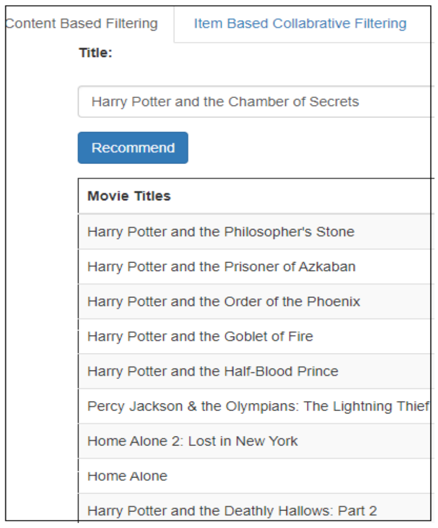


Movie Titles
Fantastic Four (2005)
You Only Live Twice (1967)
Batman Begins (2005)
Hellboy (2004)
Superman Returns (2006)
Dragnet (1987)
Octopussy (1983)
Darkman (1990)
Batman Returns (1992)
High Anxiety (1977)

Figure 13: Recommendations using Item Based filtering

VII. UI INTERFACE DESIGNS

We developed REST APIs using flask which is a python library. In the next step, we integrated these back-end APIs with UI by making asynchronous HTTP requests to the REST APIs using JavaScript. We designed UI using HTML and Bootstrap library. Figure and illustrate preliminary UI for the RBM and matrix factorization using stochastic gradient descent.



Title:

Harry Potter and the Chamber of Secrets

Movie Titles
Harry Potter and the Philosopher's Stone
Harry Potter and the Prisoner of Azkaban
Harry Potter and the Order of the Phoenix
Harry Potter and the Goblet of Fire
Harry Potter and the Half-Blood Prince
Percy Jackson & the Olympians: The Lightning Thief
Home Alone 2: Lost in New York
Home Alone
Harry Potter and the Deathly Hallows: Part 2

Figure 14: User Interface

VIII. DIVISION OF WORK

TASK	**Description**	**Contributors**
Literature Survey	Finding and reading all the relevant resources and existing methods	All
Dataset Collection	Collecting the Movie Lens dataset and preprocessing it	All

Memory-based Collaborative Filtering	Implementing item-based and user-based collaborative filtering techniques	Shrimangal Rewagad & Omkar Vedak
Restricted Boltzmann Machine	Implementing Neural Network for Restricted Boltzmann Machine	Vedant Milind Salvi & Anoop Reddy Vutukuri
Model-Based Collaborative Filtering	Implementing Stochastic Gradient Descent and Alternating Least Square methods	Dia Dutta & Arun Vignesh Malarkkan
Evaluation	Implementing methods to evaluate different recommendation strategies	All
UI	Implementing User Interface for user-system interaction	Vedant Milind Salvi
Demo	Project class demo session	All
Report	Final report with insights, evaluation, and analysis	All

IX. Conclusion

The Movie Recommender System has given interesting insights with respect to the various recommendation algorithms or models used. A detailed evaluation of each of the models was done using the Root Mean Squared Error metric and the insights are discussed as follows.

- The content-based recommendation model gave a very similar recommendation to the users' choice/profile of movie interests, i.e. the recommendations were highly specific for a particular movie search and the recommendations were very similar. For example, if the user watched a Harry Potter movie, all the recommendations were just based on the Harry Potter movie. It doesn't take into account any other factors or what other users think of a movie and so it leads to low-quality monotonous recommendations.
- Collaborative filtering methods recommend items based on how other users with similar interests rated them. We used two methods here. One which computes distance metrics to recommend to new users and one uses matrix factorization. The KNN approach which uses distance metrics gave reasonably good recommendations, but they were limited by cold start problems which are so evident in collaborative filtering algorithms since it uses a user's history of ratings or interests to recommend movies.
- Restricted Boltzman Machine model in contrast to the content-based recommendation model, was able to give good quality of recommendations with respect to the movie genres. This is because the user has similar interests in multiple genres and so the output recommendations involved all of them.
- The Matrix factorization methods which were implemented here, i.e. the Stochastic Gradient Descent and the Alternating Least Squares algorithms deal with the problems of cold start and popular bias quite well by assigning more latent factors towards the least used movies or data with least interest so that it improves the recommender's ability to recommend less known movies. However, theoretically, SGD performs faster than ALS but here while working with the Movie Lens dataset which is very sparse ALS performs really well in this case
- Also, the ALS algorithm is implemented over PySpark which makes the system scalable to a huge dataset, and as a result, top recommendation systems like Netflix run on this algorithm.

References

[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

[2] Srinivasan, A. V. (2019, September 7). Stochastic Gradient Descent - Clearly Explained !! Retrieved from https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31

[3] Yuan Yao, Hanghang Tong, Guo Yan, Feng Xu, Xiang Zhang, Boleslaw K. Szymanski, Jian Lu: Dual-Regularized One-Class Collaborative Filtering. CIKM 2014: 759-768

[4] Luo, S. (2019, February 6). Intro to Recommender System: Collaborative Filtering. Retrieved from https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26

[5] Valkov, V. (2019, June 30). Music artist Recommender System using Stochastic Gradient Descent: Machine Learning from Scratch... Retrieved from https://towardsdatascience.com/music-artist-recommender-system-using-stochastic-gradient-descent-machine-learning-from-scratch-5f2f1aae972c

[6] C. M. Wu, D. Garg, and U. Bhandary, "Movie Recommendation System Using Collaborative Filtering," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 11-15.

[7] N. Yi, C. Li, X. Feng and M. Shi, "Design and Implementation of Movie Recommender System Based on Graph Database," 2017 14th Web Information Systems and Applications Conference (WISA), Liuzhou, 2017, pp. 132-135.

[9] Insight. (2016, August 11). Explicit Matrix Factorization: ALS, SGD, and All That Jazz. Retrieved from https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea

[10] Grover, P. (2020, March 31). Various Implementations of Collaborative Filtering. Retrieved from https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

[11] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In Proceedings of the 24th international conference on Machine learning

(ICML ’07). Association for Computing Machinery, New York, NY, USA, 791–798. DOI:https://doi.org/10.1145/1273496.1273596

[13] Nayak, M. (2019, April 19). Recommender Systems Using RBM. Retrieved from https://medium.com/datadriveninvestor/recommender-systems-using-rbm-79d65fcadf8f

[14] Adityashrm21. (2018, November 17). Building a Book Recommender System using Restricted Boltzmann Machines. Retrieved from https://adityashrm21.github.io/Book-Recommender-System-RBM/

[15] Banik, R. (2017, November 10). The Movies Dataset. Retrieved from https://www.kaggle.com/rounakbanik/the-movies-dataset

[16] Banik, R. (2017, November 6). Movie Recommender Systems. Retrieved from https://www.kaggle.com/rounakbanik/movie-recommender-systems/execution

[17] Deng, H. (2019, December 5). Recommender Systems in Practice. Retrieved from https://towardsdatascience.com/recommender-systems-in-practice-cef9033bb23a

[18] Mwiti, D. (2019, December 3). How to build a Simple Recommender System in Python. Retrieved from https://towardsdatascience.com/how-to-build-a-simple-recommender-system-in-python-375093c3fb7d

[19] Kordík, P. (2019, December 15). Machine Learning for Recommender systems - Part 1 (algorithms, evaluation and cold start). Retrieved from https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed