MovieRec: A Movie Recommendation System

Amit Kumar
University of Southern California
Los Angeles CA 90081
kuma310@usc.edu

Aarushi Gupta
University of Southern California
Los Angeles CA 90081
aarushi@usc.edu

Abstract
Recommender systems can be defined as systems that aim to recommend the most suitable items to the users by predicting a user's interest based on related information about the items, the users and their interaction with each other. This helps improve customer relationship management. Recommendations have been used since decades to boost business.
They have applications in various domains: e-commerce, e-learning, e-business, e-tourism, e-government, e-resource services etc. Looking at the widespread application of recommendation systems, we decided to implement a movie recommendation system to recommend movies to users. We used the MovieLens 20M dataset from Kaggle which had 20 million ratings given by users to different movies as our training data. Our movie recommendation system used collaborative filtering which we implemented using the Alternating Least Squares (ALS) Algorithm. This is the most popular algorithm used by recommendation systems and has been widely used by various well-known softwares. Since this algorithm involves computation which does not depend on previous iterations, it can be executed in parallel. Hence, we could use pyspark to parallelize the processing on the huge dataset.

1. Problem

These days there are tons of data available over the internet. When a user searches for information about a topic, he gets billions of results related to that topic but all of them might not be of interest to him.

Though state-of-the-art web browsers try to provide relevant results to the user, they fail to provide customised results that will differ from user-to-user.

These days there is more to "results" than just results pertaining to a search query. When a user buys a product through an e-commerce website or watches a video on YouTube or a movie on Netflix, he expects the software to provide him with suggestions of products/ videos/ movies that he may like and hence, that might be worth watching without having to undergo the tedious process of finding an item he might like.

Hence, here we are faced with the modern day problem of providing customised suggestions/ results that would be relevant to the user and displaying it as the top most result.

2. Introduction

Recommender systems can be defined as systems that aim to recommend the most suitable items to the users by predicting a user's interest based on related information about the items, the users and their interaction with each other. This helps improve customer relationship management.

In this project, we focus on recommending movies of interest to the user, that is, we aim to build a movie recommendation system.

A movie may encompass tons of genres under it, but the user might be interested in just a few of them. What genres or groups of genres the user might be interested in, can be guessed by considering his past history of activities, that is, by considering the movies he has watched and the ratings he bestowed upon those movies. By doing so, we would be taking into account what he liked and disliked in the past.

This will enable us to make intelligent guesses about his preferences and provide him results or suggestions that are more likely to be relevant to him and that are more likely to satisfy his interests.

3. Recommender Systems

Recommender systems aim to predict the "rating" a user would give to an item. These systems produce recommendations by one of the two methods - Collaborative Filtering or Content-based Filtering.

Collaborative Filtering uses the user's past behaviour and decisions made by similar users to build a model to predict items the user might be interested in. User's past behaviour includes items he previously purchased or movies he previously watched or the ratings given by him to the various items. The basic assumption of collaborative filtering is that if user X has the same opinion about most items as that of user Y, user X is more likely to have the same opinion as user Y about a different item compared to a randomly picked user. For example, if user X and user Y have similar ratings for most of the movies they have watched in common, user X is more likely to like other movies than user Y has liked, and dislike other movies that user Y has disliked compared to some other random user.

In this project, we have implemented the Alternating Least Squares algorithm, which is the most popular algorithm used to perform collaborative filtering based recommendation.

Content-based filtering systems typically use the various characteristics of an item to recommend items with similar properties. Based on what the user likes, the algorithm recommends other items with similar properties to the user. These systems do not depend on the choices made by the other users and rely solely on the properties of the items. Though there will be less diversity in the recommendations , this method does not require the user to rate items. Similarity metrics like cosine similarity can be used to find items with similar properties.

Although both methods are used by recommendation systems, collaborative filtering does have some advantages over content-based.
Some of the advantages are:
1. There is no need to know about the content of items.
2. The "item cold-start" problem can be avoided, that is, when information about an item is not available, it is not required as long as the item has been used by some user.
3. User interests may change over time and by only considering the properties of items as in content-based filtering, we lose flexibility on the user's side.
These and many more advantages of collaborative filtering over content-based filtering motivated us to choose this method for our movie recommender system. We use the Alternating Least Squares algorithm to implement Collaborative filtering in this project.

4. Dataset

We used the MovieLens 20M Dataset from Kaggle for our Movie Recommendation System. This dataset has 20 million ratings given by users to various movies. Besides this, it has attributes like movie name, genres, release date, movie tags etc. It has 27,256 movies, 138493 unique users. The size of ratings.csv is 658MB. As the data was too huge, we used 75% of the data to train our model.

We also used the dataset from IMDB (Internet Movie Database) to fetch movie posters that could be displayed on the home page of our recommendation system. This database is updated daily and has entries for almost all the movies released worldwide.

5. Alternating Least Squares: The Algorithm

The Alternating Least Squares algorithm implements collaborative filtering by using matrix factorization.

Collaborative filtering, as has been stated earlier, makes predictions about a user's interest by learning his preferences based on his reaction (ratings here) on certain movies along with the reaction of other users on the same set of movies. By discovering a user's engagement pattern with items and other users of the system, the system is able to infer the user's hidden preferences and make recommendations of potentially interesting items (movies here) to them.

Clustering, as we know, is a method of grouping similar items together. Clustering approaches are divided into two types: soft clustering and hard clustering. In hard clustering, one object can belong to only one cluster whereas in soft clustering, one object could potentially belong to more than one cluster. Hence, soft clustering is also known as "fuzzy clustering".

Real world data is generally created by a joint interaction among two types of entities, that is, it is bimodal. This is referred to as the co-clustering problem.

For example, a user's rating to a movie is affected by both, the user's characteristics (his preferences towards a particular set of genres) and the movie characteristics (its association with a certain set of genres). Such signals are generally represented as matrices where each dimension represents one entity type (here, one dimension represents movies, while the other represents users). In this matrix, users are represented by rows, movies by columns and the value of each cell represents the rating given by the corresponding user to the corresponding movie. This is called the "Characteristic Matrix". Co-clustering this matrix would mean grouping both similar users and similar movies synchronously.

The characteristic matrix is sparse by nature. This is intuitive given the large number of users and movies. The number of movies watched by a user is very low compared to the total number of movies. It is also very unlikely that a lot of users have watched and rated the same movies. Hence, clustering users based on the movies they have watched, or clustering movies by overlapping viewers seems useless. It is here that the co-clustering approach finds use.

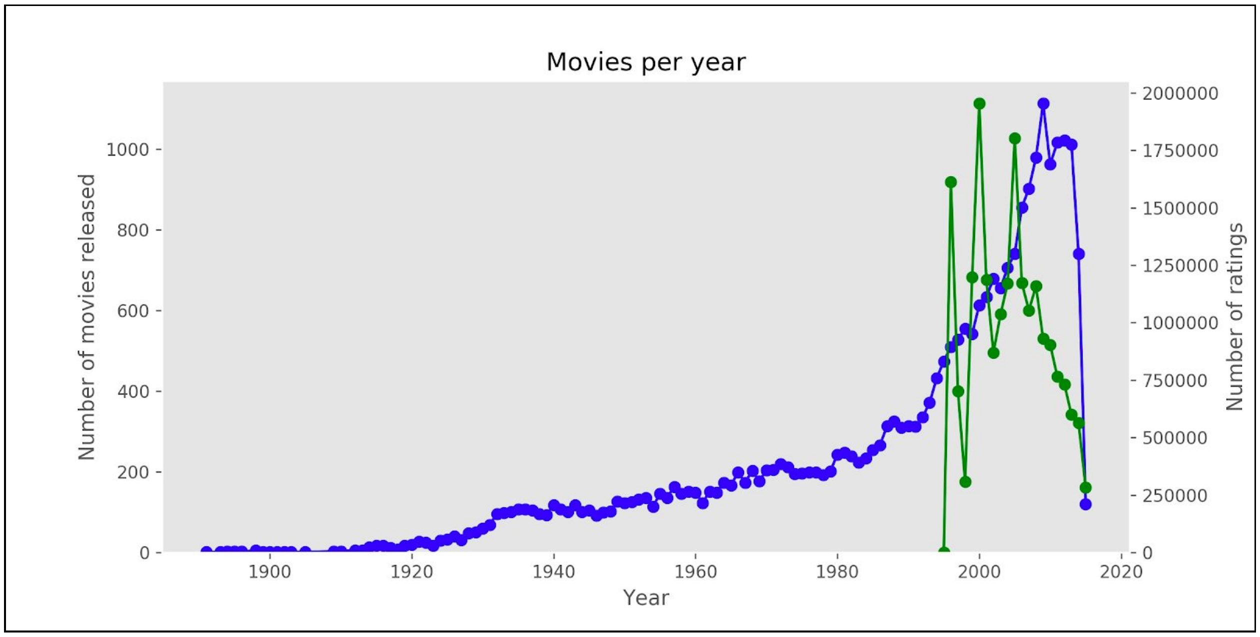


Figure 1: The above graph shows skewed data. Green = Ratings in that year
Purple = Movies released in that year

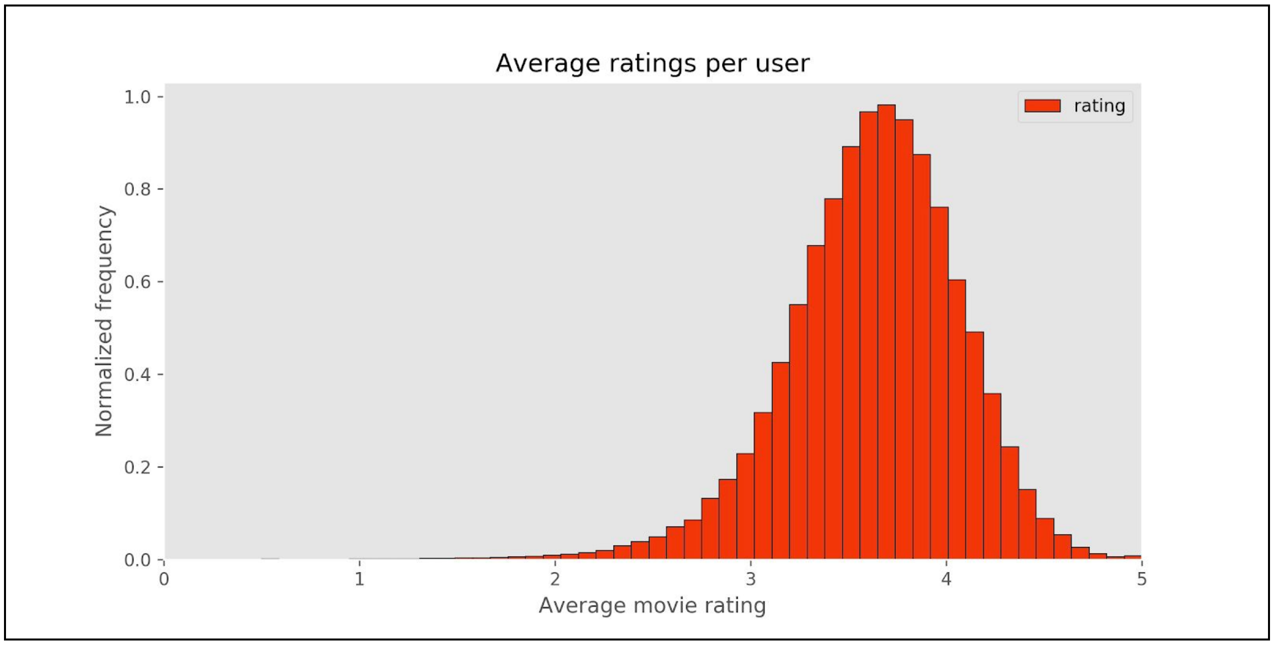
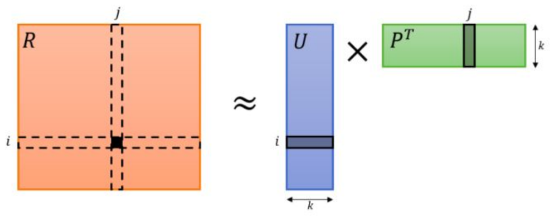


Figure 2: Distribution of Ratings.
The above graph shows the people tend to rate only they like.

5.1. Matrix Factorization

Matrix Factorization is amongst the most popular algorithms to solve the co-clustering problem. This algorithm assumes the Characteristic matrix (R), a mxn matrix, with m users and n movies. It factorizes R into two matrices, U (a mxk matrix) and V (a kxn matrix) such that R ≈ U x V (Their multiplication is approximately equal to R.



A new quantity, k, has been introduced that serves as the dimensions of both U and V. Each $R_{i,j} = U_i \,.\, V_j$, that is each $R_{i,j}$ is factorized into the dot product of $U_i \,.\, V_j$ where $U_i \,.\, V_j$ $U_i \,.\, V_j \, \varepsilon \, R^k$. This model assumes k effects affect every rating in R. Both users and movies in U and V are represented in terms of these k effects.

Therefore, we can assume, in the user movie rating matrix, R, each movie is linked with one or more categories with some strength and with different strengths, each user likes or dislikes some movie categories. The rating a user gives a movie should depend on the similarity between his characteristics and the characteristics of the movie.

Matrix factorization approximates the original rating matrix with two smaller matrices, U and V, such that the following cost function is minimized:

$$J = \| R - U \times V^T \|_2 + \lambda (\|U\|_2 + \|V\|_2)$$

The first term in the above cost function refers to the Mean Square Error (MSE) distance measure between R, the original ratings matrix and $U \times V^T$, its approximation. The second term is the regularization term and it helps to prevent overfitting. This is hence, an optimization problem.

With the cost function having introduced two new parameters, k and λ, it is imperative to determine the optimal values for these parameters. Once determined, given these values of k and λ, the Alternating Least Squares algorithm is used to have this cost function converge to a minimum.

5.2. Alternating Least Squares

We aim at minimizing the cost function obtained above from Matrix Factorization. Hence, we would need to learn the two variables - the U matrix and the V matrix. The cost function, as we have seen above is the following:

$$J = \| R - U \times V^T \|_2 + \lambda (\|U\|_2 + \|V\|_2).$$

Here the first term is the sum

$$\| R - U \times V^T \|_2 = \sum_{i,j} (R_{i,j} - U_i \times V_j).$$

Alternating Least Squares is a two-step iterative optimization process. In every iteration, the algorithm first fixes V and solves for U, and after which, it fixes U and solves for V. As in linear regression, when we solve for β by minimizing the squared error given by X and y: $\|y - X\beta\|_2$, the solution ultimately is given by the Ordinary Least Squares formula:

$$\beta = (X^T X)^{-1} X^T y.$$

Since the Ordinary Least Squares guarantees the minimal Mean Square Error and is unique, in every step the cost function either decreases or remains unchanged but never increases. Hence, alternating between the two steps until convergence, guarantees a reduction of the cost function. According to the two step function, the cost function can be broken down as follows:

$$\forall u_i : J(u_i) = \|R_i - u_i \times V^T\|_2 + \lambda\|u_i\|_2$$

$$\forall v_j : J(v_j) = \|R_i - U \times v_j^T\|_2 + \lambda\|v_j\|_2$$

The corresponding solutions for u_i and v_j are:

$$u_i = (P^T \times P + \lambda I)^{-1} \times P^T \times R_i$$

$$v_j = (U^T \times U + \lambda I)^{-1} \times U^T \times R_j$$

Since each u_i is independent of other $u_{j \neq i}$ vectors, and each v_j is independent of other $v_{i \neq j}$ vectors, each step can be performed in parallel.

6. Challenges

6.1. Implementing ALS in Pyspark

While working on this project, the first challenge that we were faced with was implementing the Alternating Least Squares (ALS) algorithm in pyspark.

We had successfully implemented the algorithm in python but trying to parallelize the algorithm to implement it in spark was a tedious task. It was tedious because we needed to access and update a global characteristic matrix across clusters and updates to the matrix were to be made within different clusters. This meant this characteristic matrix needed to be broadcasted, but again due to the huge size of the data, it was not possible to broadcast the entire data in a single matrix.

6.2. Handling the huge dataset

The greatest challenge that we were posed with was that of handling the huge dataset at hand.
We used the "MovieLens 20M Dataset" from Kaggle which has data pertaining to movies such as movie_id, title, genre, user_id, rating, timestamp, tags etc.

This dataset can be found at: https://www.kaggle.com/grouplens/movielens-20m-dataset#tag.csv

Due to the large size of that data, even though we used 75% of it, our model required very long training time and huge system memory. This meant the whole dataset could not be loaded into memory at once to run the algorithm.

6.3. Broadcasting the Characteristic Matrix

The most important data file for this project was the "ratings.csv" which has 20 million rows of data entry.
When we tried to compute the characteristic matrix using these 20 million rows, the data became too huge to fit into memory and the characteristic matrix creation failed.
This matrix needed to be broadcasted across all worker nodes so that the user matrix and movie matrix computations could be done in parallel.

Pyspark uses pickle to broadcast data, and is unable to serialize data larger than 2GB. Therefore, broadcasting the characteristic matrix across worker nodes in spark was the most challenging part.

6.4. Fetching images for Movies

The other challenge faced was in fetching images for the movies to display on the home page, as part of suggestions, movies watched, and recent movies. The movie ids that were in the ratings.csv file of the MovieLens 20M Dataset did not completely match the movie ids on the IMDB website. We had to fetch images from the IMDB website as the MovieLens 20M dataset did not contain images. To do so, we needed a way to be able to match the movies in the ratings.csv to those on the IMDB website. Since, the movie ids did not completely match, we needed to devise a way to match movies on ratings.csv with that on that IMDB website.

7. Solutions

Details on how we tried to solve the challenges we came across have been mentioned below:

7.1. Broadcasting the Characteristic Matrix

The characteristic matrix needed to be broadcasted across all the worker nodes in pyspark, and pyspark uses pickle for broadcasting. Pickle can serialize a maximum of 2GB of data but the size of our characteristic matrix was much larger than 2gb. Therefore, we divided the characteristic matrix into chunks of size less than 2 GB and broadcasted these chunks instead of trying to broadcast the entire characteristic matrix.
The chunk to be used was then decided at the worker nodes.

7.2. Fetching Images for Movies

In order to fetch images for movies from the IMDB dataset, we needed a way to match the movie in the ratings.csv file with that in the IMDB dataset. To do this, as the movie ids in the ratings.csv file did not match the movie ids
Ever since the mid-1990s, various recommender system techniques have been proposed and recently, various types of recommender system software have been developed for a variety of applications.

Recommender systems have applications in various domains: e-commerce, e-learning, e-business, e-tourism, e-government, e-resource services, e-news, e-books etc. Examples of how recommender systems are used in some of these domains is given below:

on the IMDB website, we fetched movie names from the movies.csv file of the MovieLens 20M Dataset and fetched movie names from the IMDB database. We performed text matching on names fetched from both datasets and the names that best matched each other were considered to be the same movie. We then used the IMDB dataset's movie id as the principle movie id and mapped movies from the MovieLens 20M Dataset using this id. This movie id was then used to fetch images corresponding to movies from the IMDB dataset.

8. Applications

A recommender system aims to provide personalised online products or service recommendations to users to handle the plethora of information available over the internet and make it relevant to the user. Unless personalised for each user, every user gets the same information about topics irrespective of his activity history.

Recommendations have been used since decades to boost business. The most obvious example of recommendation systems helping businesses can be seen in grocery stores where salsa sauce is kept near the chips aisle, or jams next to bread, etc.

E-commerce: Product suggestions are made to customers based on their order history and/or browsing history. These suggestions are drawn from users who have exhibited similar buying patterns.

E-tourism: A tourism website uses the places a customer visits and likes to recommend him other places that might be of interest to him. These suggestions are mostly based on places that similar people have visited and liked.

E-learning: A website that offers courses to users can suggest new courses to the user related to the ones he has undertaken, like courses about similar topics, or related topics.

E-books: Based on the books read by a user, the e-books site tries to predict books the user may like by relying on the genres of books read by the user and/or the books other users with similar preferences liked.

Hence, looking at the all the examples above, it is evident that recommendation systems have gained widespread popularity and have become a business model for various domains. It is liked by the industry as it boosts revenue and it is liked by the user as it provides him items of his interest without much effort from his side. It thereby makes the system more user-friendly and increases its usage.

9. Evaluation Results

The evaluation results of our model have been details below.

Id	Title	Rating	Genres
118696	The Hobbit: The Battle of the Five Armies	5.0	Adventure|Fantasy
106489	Hobbit: The Desolation of Smaug	5.0	Adventure|Fantasy|IMAX
98809	Hobbit: An Unexpected Journey, The (2012)	5.0	Adventure|Fantasy|IMAX
51662	300 (2007)	5.0	Action|Fantasy|War|IMAX
7438	Kill Bill: Vol. 2 (2004)	5.0	Action|Drama|Thriller
7153	Lord of the Rings: The Return of the King	5.0	Action|Adventure|Drama|Fantasy
6874	Kill Bill: Vol. 1 (2003)	5.0	Action|Crime|Thriller
5952	Lord of the Rings: The Two Towers, The..	5.0	Adventure|Fantasy
4993	Lord of the Rings: The Fellowship of the Ring.	5.0	Adventure|Fantasy
1291	Indiana Jones and the Last Crusade (1989)	5.0	Action|Adventure

Table 1.0 Actual Movies Ratings by user with id 65432

Id	Title	Rating	Genres
2571	Matrix, The (1999)	4.669288	Action|Crime|Drama
6539	Pirates of the Caribbean: The Curse of the Bla.	4.648897	Action|Adventure|Sci-Fi
58559	Dark Knight, The (2008)	4.637863	Action|Adventure|Sci-Fi
1196	Star Wars: Episode V - The Empire Strikes...	4.070639	Action|Sci-Fi|Thriller
1210	Star Wars: Episode VI - Return of the Jedi	3.870287	Adventure|Animation|Fantasy
6377	Finding Nemo (2003)	3.635052	Adventure|Animation|Comedy
318	Shawshank Redemption, The (1994)	3.583591	Action|Adventure|Comedy|Fanta
8961	Incredibles, The (2004)	3.552151	Action|Adventure|Animation
4886	Monsters, Inc. (2001)	3.535238	Action|Crime|IMAX
33794	Batman Begins (2005)	3.501225	Action|Crime|Drama|IMAX

Table 1.1 Predicted Movies Ratings for user with id 65432

Table 1.0 shows that the user is most interested in Action, Adventure and Fantasy Movies. User has not given high ratings to any movie in the Horror, Romance genre. Table 1.1 shows the predictions for this user from our Alternating Least Squares Model. This model only uses the ratings given by the user to the different movies without considering their genre. But, looking at the genres in Table 1.1, it is evident that the model has made appropriate predictions and has predicted high user ratings for movies in the Action, Adventure, Sci-Fi, fantasy genres.

Id	Title	Rating	Genres
3178	Hurricane, The (1999)	5.0	Drama
3004	Bachelor, The (1999)	5.0	Comedy|Romance
2997	Being John Malkovich (1999)	5.0	Comedy|Drama|Fantasy
2858	American Beauty (1999)	5.0	Comedy|Drama
2762	Sixth Sense, The (1999)	5.0	Drama|Horror|Mystery
2690	Ideal Husband, An (1999)	5.0	Comedy|Romance
2359	Waking Ned Devine (a.k.a. Waking Ned)	5.0	Comedy
2324	Life Is Beautiful (La Vita 8 bella) (1997)	5.0	Comedy|Drama|Romance|War
1500	Grosse Pointe Blank (1997)	5.0	Comedy|Crime|Romance
1446	Kolya (Kolja) (1996)	5.0	Comedy|Drama

Table 2.0 Actual Movies Ratings by user with id 76543

Id	Title	Rating	Genres
357	Four Weddings and a Funeral (1994)	4.579528	Drama|Romance
17	Sense and Sensibility (1995)	4.486611	Drama|Fantasy|Romance
1641	Full Monty, The (1997)	4.129497	Comedy|Crime|Drama|Thriller
2571	Matrix, The (1999)	4.011674	Comedy|Drama|Romance|War
1704	Good Will Hunting (1997)	3.963075	Comedy|Romance
296	Pulp Fiction (1994)	3.902584	Comedy|Romance
1617	L.A. Confidential (1997)	3.892601	Crime|Film-Noir|Mystery|Thriller
356	Forrest Gump (1994)	3.891726	Comedy|Drama
1307	When Harry Met Sally... (1989)	3.847840	Drama|Romance
265	Like Water for Chocolate (Como agua para..	3.847305	Action|Sci-Fi|Thriller

Table 2.1 Predicted Movies Ratings for user with id 76543

Similarly, Table 2.0 shows that the user is mostly interested in movies of the Comedy, Romance, Drama genres. Table 2.1 shows the predictions our Alternating Least Squares models has made for this user. It can be seen from this table that our model has predicted high ratings for this user for movies with the Comedy, Drama and Romance genres.

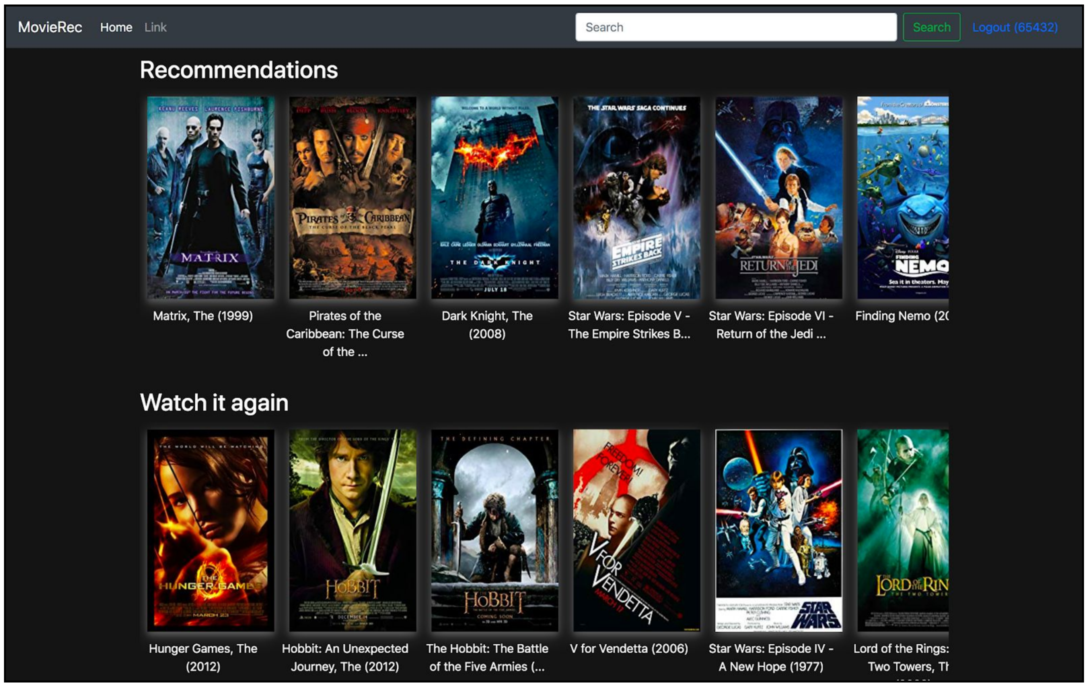

MovieRec
Home
Link
Search
Search
Logout (65432)
Recommendations
Matrix, The (1999)
Pirates of the Caribbean: The Curse of the ...
Dark Knight, The (2008)
Star Wars: Episode V - The Empire Strikes B...
Star Wars: Episode VI - Return of the Jedi ...
Finding Nemo (20
Watch it again
Hunger Games, The (2012)
Hobbit: An Unexpected Journey, The (2012)
The Hobbit: The Battle of the Five Armies (...
V for Vendetta (2006)
Star Wars: Episode IV - A New Hope (1977)
Lord of the Rings: Two Towers, Th

Figure 3: Home Page of our Movie Recommendation System

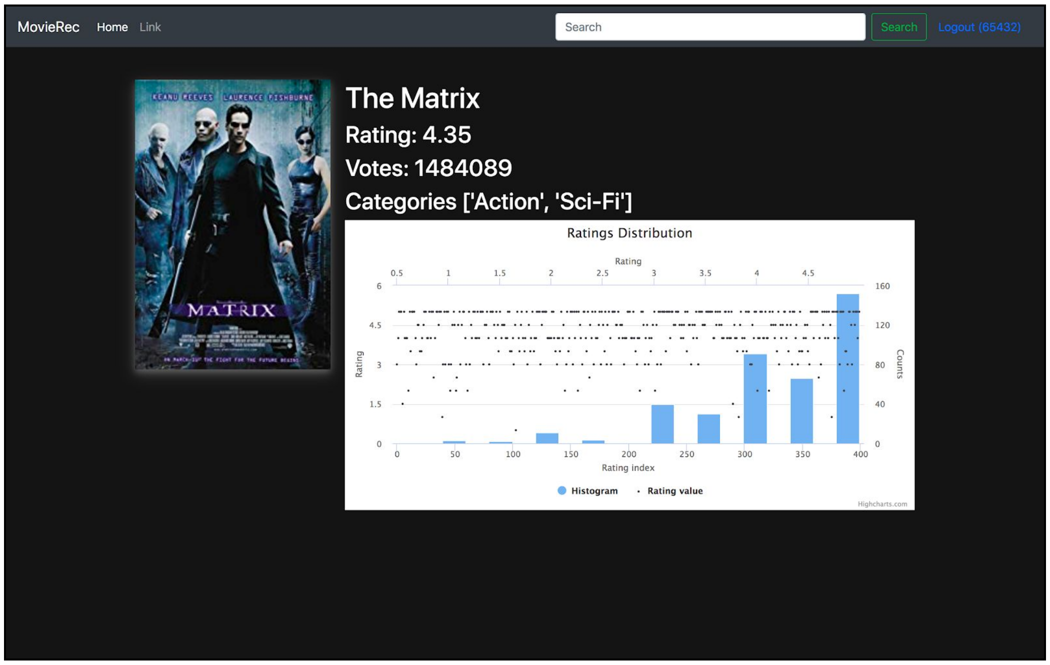


Figure 4: On clicking a movie on the home page, the user lands on this page. This page shows the distribution of ratings given by users to this movie.

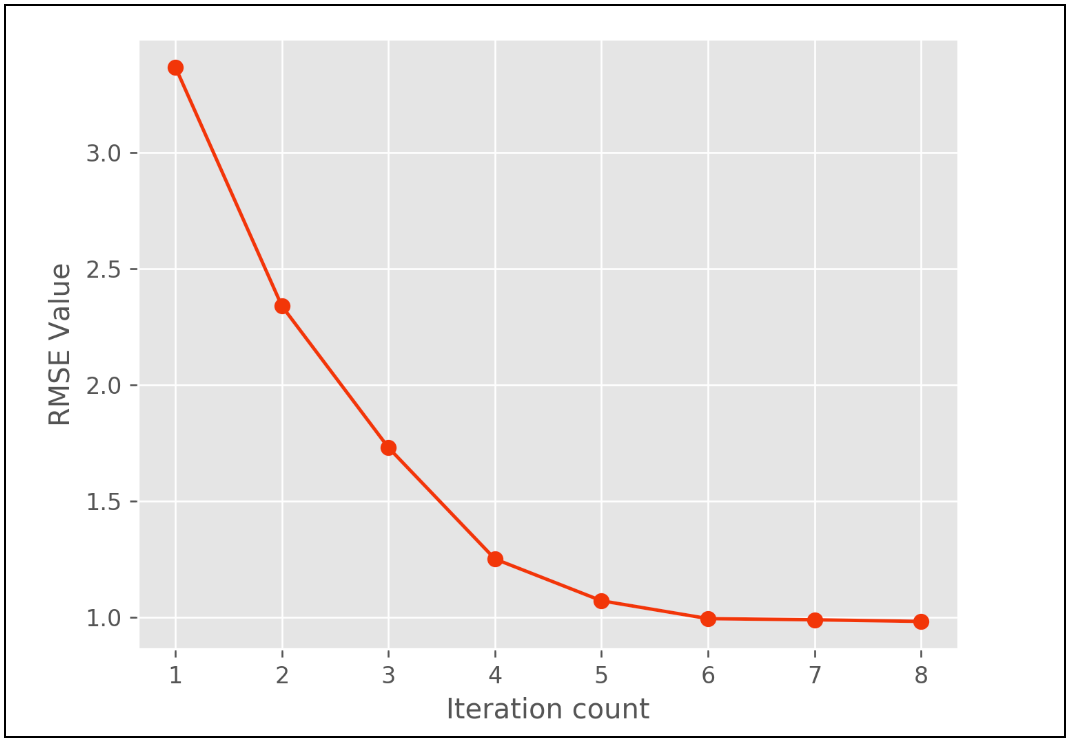


Figure 5: The above curve shows the RMSE values for different iterations of our ALS model. It can be seen from the graph that we got a minimum RMSE of 0.981.

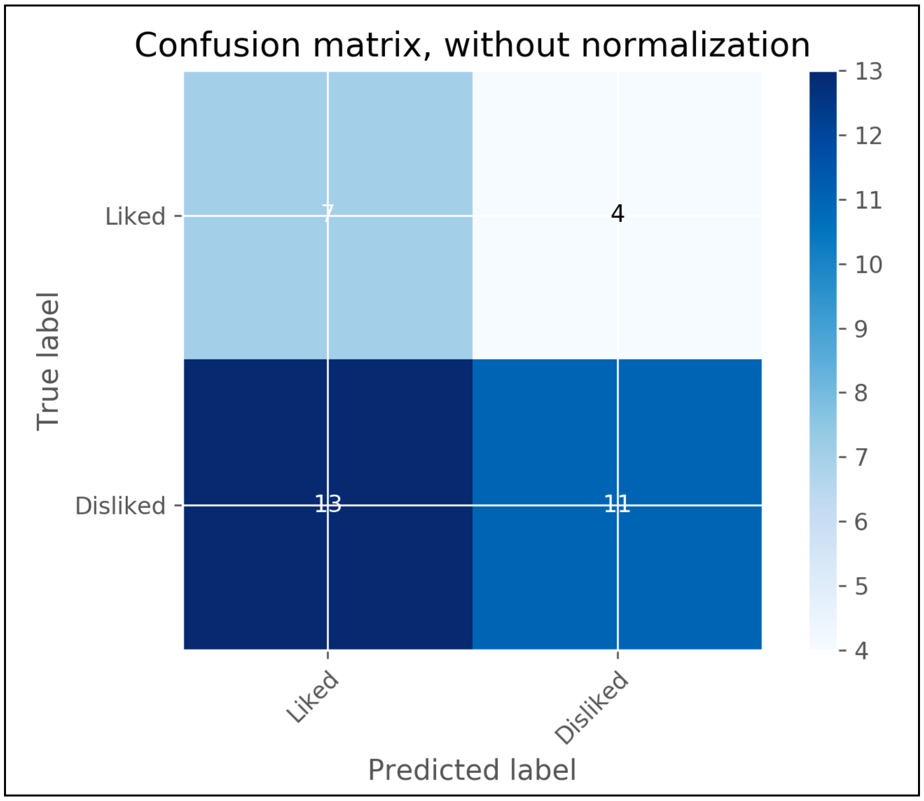


Figure 6: Confusion Matrix

Concluding Remarks

By analysing the movie recommendations provided by our model to users, we can conclude that the biased version of Alternating Least Squares is performing well as the recommendations were pretty appropriate to the users' interests. Also, since our data was very large, in order to parallelize the computations of U-V matrix factorizations, we had to split the characteristic matrix and broadcast this matrix in chunks to all the worker nodes.

Acknowledgements

We would like to thank Professor Wensheng Wu for his encouragement and valuable comments during the Project Proposal Presentation and Final Project Presentation. We would also like to thank our Teaching Assistants, Abhishek Bhatt, Qianmu Yu and Yiwen Tang for the support, guidance and clarifications they provided to us during the entire duration of our project.

References

[1] Yifan Hu, Yehuda Koren and Chris Volinsky. "Collaborative Filtering for Implicit Feedback Datasets"

[2] Christopher R. Aberger "Recommender: An Analysis of Collaborative Filtering Techniques"

[3] Kaggle.com "MovieLens 20M Dataset"

[4] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu and Rajan Lukose. "One-Class Collaborative Filtering"