**IDS 561 – Big Data Analytics**

**Movie Recommendation System using ALS**

**Team**

**Charu Yadav - 653762591**

**Deepak Singhal - 672190946**

**Introduction**

Isn’t it amazing to find the products which are very similar to your choice without putting any efforts? Almost everyone has noticed and experienced this thing while using Netflix. We watch one movie and based on that Netflix recommends many similar movies. A recommendation system is very important in today’s world. This type of system can help us to find appropriate movies based on customers’ interests and the popularity of the movies. The recommendation system is mainly useful in commercial applications like Netflix.

There are many methods that can be used in recommendation systems such as collaborative filtering, content-based filtering, and hybrid system. Collaborative filtering uses a customer’s history along with the other customers’ history and using all this information, it builds a model. Through this, a customer will get all the movies recommendation which was watched by similar customers. Content-based filtering focuses on content instead of the customers. It compares the movies which are very similar in content and based on that it recommends movies to people. Both systems have pros as well as cons. For the collaborative filtering method, a huge dataset is required which is challenging sometimes. But content-based filtering can produce results with a very small dataset, but its scope is very limited as it can only recommend movies that are like the original movie.

**Literature**

Recommender systems were first mentioned in a technical report as a "digital bookshelf" in 1990 by Jussi Karlgren at Columbia University and implemented at scale and worked through in technical reports and publications from 1994 onwards by Jussi Karlgren, then at SICS and research groups led by Pattie Maes at MIT Will Hill at Bellcore and Paul Resnick, also at MIT whose, work with Group Lens was awarded the 2010 ACM Software Systems Award.

Many people have worked in this field utilizing big data techniques such as spark to develop a model using huge datasets. Many fields like social networks, e-commerce, education, etc. are using recommendation systems to grow their business. The hybrid system was first designed by Wei et al. [2]. This approach focuses on a mixture of collaborative and content-based filtering. Using a collaborative filtering algorithm based on items a recommendation system was developed by Kupisz and Unold [3]. This paper used Apache spark to develop this model along with MapReduce to handle the large datasets. Zhou et al. [4] developed a collaborative filtering-based ALS Algorithm for the Netflix Prize. This method handles the scalability issue of extensive datasets.

**Data**

This dataset describes 5-star rating from **MovieLens**, a movie recommendation service. It contains 25,000,095 ratings across 62423 movies. These data were created by 162,541 users between January 09, 1995, and November 21, 2019. This dataset was generated on November 21, 2019. Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided. The data are contained in the files “movies.csv”, and “ratings.csv”.

Data is taken from: <https://grouplens.org/datasets/movielens/25m/>

More details about the contents and use of all these files follow.

First Data file: Ratings Data File Structure (ratings.csv)

All ratings are contained in the file ratings.csv. Each line of this file after the header row represents one rating of one movie by one user, and has the following format:

Table 1: Columns of ratings data file

|  |  |  |  |
| --- | --- | --- | --- |
| userId | movieId | rating | timestamp |

The lines within this file are ordered first by userId, then, within the user, by movieId. Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars). Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

Second Data file: Movies Data File Structure (movies.csv)

Movie information is contained in the file movies.csv. Each line of this file after the header row represents one movie, and has the following format:

Table 2: Columns of movies data file

|  |  |  |
| --- | --- | --- |
| movieId | title | genres |

Movie titles are entered manually or imported from <https://www.themoviedb.org/>, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.

Genres are a pipe-separated list, and are selected from the following:

Table 3: Different genres present in movies data file

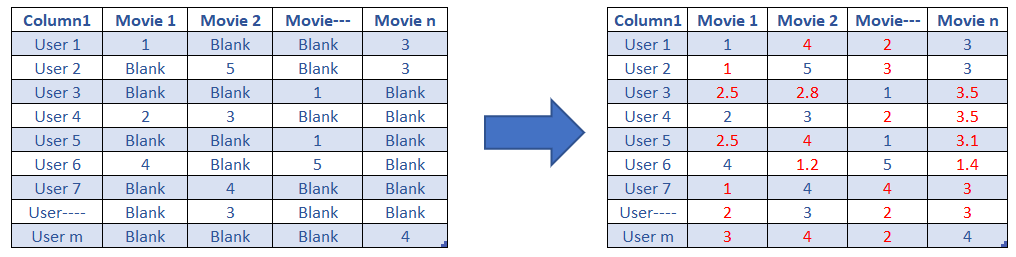
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Animation | Crime | Fantasy | Horror | Sci-fi | Thriller |
| Adventure | Comedy | Drama | Musical | Mystery | War |
| Action | Children | Documentary | Film-Noir | Romance | Western |

# Since it’s a lot of data to process (25,000,095 ratings and 62423 movies), we decided to take a subset of the data and we are considering 1,000 users and 5 reviews per user, hence in total 5,000 rows.

**Methodology**

The recommendation system uses ratings of all the users to produce personalized items for a customer. The recommendation system majorly has three categories – content-based, collaborative, and hybrid. In our project, we have decided to use collaborative filtering. This information can be presented in an m\*n matrix where m is the number of users and n is the number of movies. Here in the below picture, on the left side, we have a sparse matrix as each user is not rating all the movies and this produces a lot of missing information. When this much information is not available, is it possible to develop reliable ML models? For the model development, data is the key, this solution of data sparsity is crucial. This problem can be solved using matrix factorization.

Figure 1: Collaborative filtering functioning



Matrix factorization is basically the factorization of a matrix into a product of multiple products.

Matrix factorization can be rewritten as an optimization problem –

argmin Σ (rui – (Xu^T) Yi)^ 2 + λ( Σ|| Xu ||^ 2 + Σ||yi||^ 2)

Here, Xu vector represents all the users and Yi represents all the movies.

The optimization of this equation can produce all the ratings. This method is called **ALS**.

**Results**

Before data modeling, we did some EDA (Exploratory data analysis) on our dataset.

1. We found out the top 5 most rated movies by user as mentioned in figure 2:

Figure 2: Top rated movies

Text

Description automatically generated

1. We also filtered out the top-rated genres in the dataset as shown in figure 3

Figure 3: Top rated genres

Table

Description automatically generated

We have divided the dataset into an **80-20** ratio for training and testing purposes. When we implemented the ALS approach of Collaborative filtering, we obtained the Root mean square error of **1.239** and strongly believe when we will run our model on a complete dataset on multiple servers, this evaluation parameter (RMSE) will be much lesser.

We have used multiple **hyperparameters** to tune our model as per our dataset.

1. **Rank**: It is the number of latent factors in the model, its default value is 10, but we have used multiple values (10, 50, 100, and 150) to obtain the optimal one.
2. **regParam** specifies the regularization parameter in ALS, its default value is 1.0, but we have used multiple values (.01, .05, .1, .15, 1.0) to obtain the optimal one.
3. **maxIter**: the maximum number of iterations to run (defaults to 10)

Table 4: Tuned Hyperparameter

|  |  |  |
| --- | --- | --- |
| **Rank** | **Max Iterations** | **Regularization Parameter** |
| 150 | 10 | 0.05 |

**Top rated movies VS Recommended Movies for UserId 471:**

Figure 4: Top rated movies by UserId: 471**Graphical user interface, application

Description automatically generated**

Figure 5: Recommended movies by UserId: 471

**Graphical user interface, application

Description automatically generated**

When we observe Figure 4 and 5, we can see that the movies rated by the user and recommended to the user by our model are very much similar, hence our model has done a decent role.

Figure 6: Top 10 movies to each of the 1000 users

Text

Description automatically generated

In the end, we recommended top **10 movies** to each user (1000) as per their and others previously rated movies according to ALS algorithm under Collaborative filtering.

**Role of each team member**

Tasks in the project:

1. Data Cleaning – Both (Charu, Deepak)
2. Exploratory Data Analysis – Both (Charu, Deepak)
3. Data Modeling – Both (Charu, Deepak)
4. Conclusion and Results – Both (Charu, Deepak)

We tried to keep our group small in the beginning, so that both of us can get a chance of working on all departments. We both used to implement each step on our own and then merge the code.

**Bibliography**

1. <https://ijesc.org/upload/f0d1e3f5683da81c9018ff3308495420.A%20Movie%20Recommender%20System%20MOVREC%20using%20Machine%20Learning%20Techniques.pdf>
2. J. Wei, J. He, K. Chen, Y. Zhou, Z. Tang, “Collaborative filtering and deep learning-based hybrid recommendation for cold start problem,” In 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech) (pp.874-877). IEEE,2016, August.
3. J.P Verma, B. Patel, A. Patel, “Big data analysis: recommendation system with Hadoop framework,” In 2015 IEEE International Conference on Computational Intelligence Communication Technology (pp. 92-97), 2015, February, IEEE.
4. 4. Alzogbi, P. Koleva, G. Lausen, “Towards Distributed Multi-Model Learning on Apache Spark for Model-Based Recommender,” In 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW) (pp. 193-200). IEEE,2019, April.