Movie Recommendation System

1 Abstract

Movies have long been a staple of the entertainment industry in both the United States and across the globe. Around 19 percent of U.S. adults claim to watch or stream movies every day, while an additional 26 percent state that they watch movies several times per week. As of 2018, 54 percent of U.S.-based respondents stated that their favorite moving watching location was at home. Only around 13 percent of respondents stated that they preferred to watch movies in a theatre, while 22 percent stated that they like the theatre and home options equally. 54 percent is a very large percentage and since they prefer to watch movies in the home they need a good source of entertainment websites or applications. Some of them include Netflix and Amazon Prime Video. Netflix revenue for the year 2021 is 7.6 Billion USD and it has 219.7 million subscribers as of year 2021. The main reason behind such a successful company is of course their video content but from a technical person point of view it is their recommendation system and user interface. So the recommendation system is which we are modelling in this project using collaborative filtering.

2 Collaborative Filtering

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. It looks at the items they like and combines them to create a ranked list of suggestions. There are many ways to decide which users are similar and combine their choices to create a list of recommendations. To build a system that can automatically recommend items to users based on the preferences of other users, the first step is to find similar users or items. The second step is to predict the ratings of the items that are not yet rated by a user.

Apart from Collaborative filtering there are many different types of recommendation system like Content based filtering, Demographic based recommendation etc. The following figure shows the difference between collaborative filtering and content based filtering. From the above example,

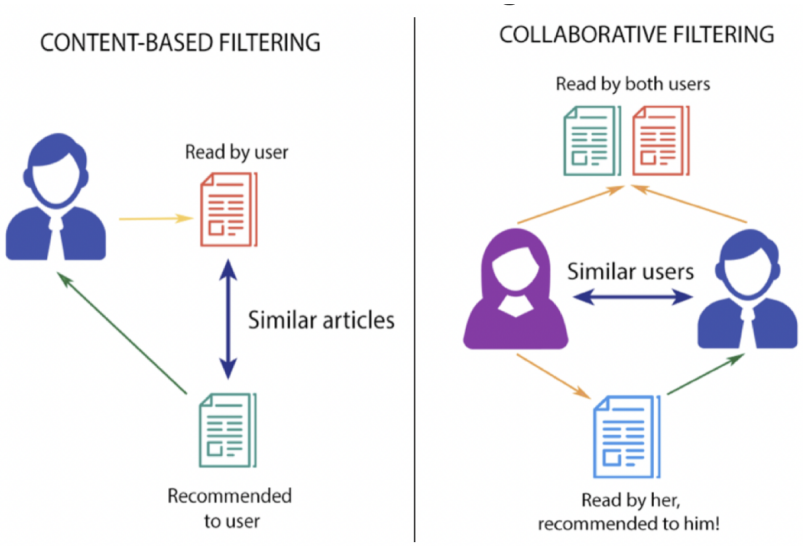

Figure 1: Difference between Content and Collaborative based filtering

Content based filtering recommends an article if the user has read a similar article. Here similarity can be based on the semantic meaning of the article or the tags in the article. Whereas in collaborative filtering , here two users read a different article before but since they are similar the second person gets the recommendation from the articles the first person read. Here similar users can be from the same geolocation or from the same workplace or persons who have the same hobbies or habits.

3 Implementing Collaborative based Recommendation System

To implement the recommendation system we need to have the data. The data used in the project is from the movielens dataset. The data contains movies and user ratings for movies. Movies has information about movie id, movie name and its genre. The user ratings contains information about userid , movieid and rating given by the user for that movie. Assume the data to be a matrix such that for every user i there is a cell for the movie j rating. This matrix is called a sparse matrix. The image shows an example of a sparse matrix which represents user ratings for the movies in

	Movie1	Movie2	Movie3	Movie4	Movie5
U1		5	4	2	1
U2	1			5	3
U3	1	4	4	1	
U4			2		2
U5	3	1	1		

Figure 2: Sparse Matrix

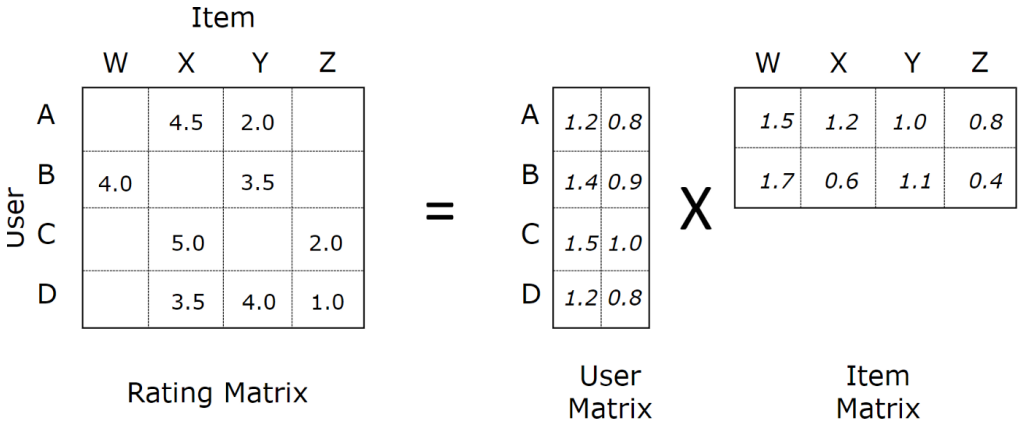

Figure 3: Sparse Matrix

Factorization is the method of expressing something big as a product of smaller factors. Similarly, Matrix Factorization finds two rectangular matrices with smaller dimensions to represent a big rating matrix (RM). These factors retain the dependencies and properties of the rating matrix. One matrix can be seen as the user matrix (UM) where rows represent users and columns are k latent factors. The other matrix is the item matrix (IM)where rows are k latent factors and columns represent items. Here k < number of items and k < number of users. Now the rating matrix is represented as a dot product of user and item matrix.

In this case, the model still learns to factorize the sparse matrix, but RMSE will be calculated only for the ratings that are actually present in the matrix. After obtaining the best factors to approximate the ratings we have, we then perform dot product of the factor matrices to fill in the missing entries in the rating matrix. These filled in ratings are then used to provide recommendations to the users. The image in figure 3 shows an example of matrix factorization and decomposition into two rectangular matrices of lower dimension.

How does matrix factorization solve our problems? Model learns to factorize rating matrix into user and movie representations, which allows model to predict better personalized movie ratings for users With matrix factorization, less-known movies can have rich latent representations as much as popular movies have, which improves recommender's ability to recommend less-known movies In the sparse user-item interaction matrix, the predicted user rating u for given item i is computed as:

$$(r)_{ui} = \sum_{f=0}^{nfactors} (H)_{uf} * (W)_{fi} \tag{1}$$

Here H is the user matrix and W is the item matrix. Rating of item i given by user u can be expressed as a dot product of the user latent vector and the item latent vector. The idea behind matrix factorization is to use latent factors to represent user preferences or movie topics in a much lower dimension space. Matrix factorization is one of very effective dimension reduction techniques in machine learning.

4 Architecture Diagram

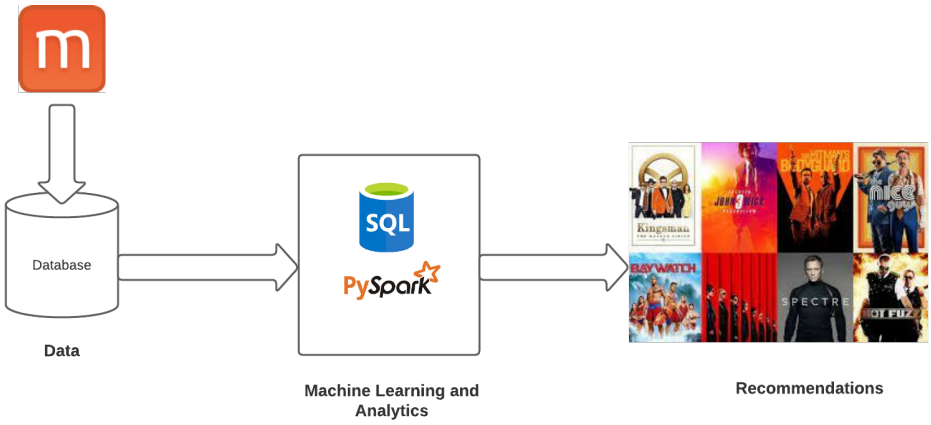

Figure 4: Architecture Diagram

Figure 4 shows the architecture used in the project.The data was loaded from MovieLens dataset and the data is preprocessed usinfg PySpark. Data Analysis was made using PySpark SQL library. The model is trained using SparkML libraries and tested through the same. The recommmendations were drawn using Spark framework.

5 Alternating Least Squares Model

The model we use to generate movie recommendations is Alternating Least Squares. Alternating Least Square (ALS) is also a matrix factorization algorithm and it runs itself in a parallel fashion. ALS is implemented in Apache Spark ML and built for a larges-scale collaborative filtering problems. ALS is doing a pretty good job at solving scalability and sparseness of the Ratings data, and it's simple and scales well to very large datasets.

Its training routine is different: ALS minimizes two loss functions alternatively; It first holds user matrix fixed and runs gradient descent with item matrix; then it holds item matrix fixed and runs gradient descent with user matrix. Its scalability: ALS runs its gradient descent in parallel across multiple partitions of the underlying training data from a cluster of machines. The following is the algorithm used for Stochastic Gradient Descent.

SGD Algorithm for MF

Input: training matrix V, the number of features K, regularization parameter λ, learning rate ϵ
Output: row related model matrix W and column related model matrix H

Initialize W, H to $UniformReal(0, \frac{1}{\sqrt{K}})$
repeat
  for random $V_{ij} \in V$ **do**
    $error = W_{i*}H_{*j} - V_{ij}$
    $W_{i*} = W_{i*} - \epsilon(error \cdot H_{*j}^{\intercal} + \lambda W_{i*})$
    $H_{*j} = H_{*j} - \epsilon(error \cdot W_{i*}^{\intercal} + \lambda H_{*j})$
  end for
until convergence

6 Model Hyperparameters and Tuning

Just like other machine learning algorithms, ALS has its own set of hyper-parameters. We probably want to tune its hyper-parameters via hold-out validation or cross-validation. Most important hyper-params in Alternating Least Square (ALS):

maxIter: the maximum number of iterations to run (defaults to 10)

rank: the number of latent factors in the model (defaults to 10)

regParam: the regularization parameter in ALS (defaults to 1.0)

Hyper-parameter tuning is a highly recurring task in many machine learning projects. We can code it up in a function to speed up the tuning iterations. In this project we used a set of hyper-parameters for each model and found that the least RMSE(0.83) is observed for the model with maxIter 15, rank 10 and regParam 0.01.

7 Workflow of the model

Our work flow is following:

A new user inputs his/her favorite movies, then system create new user-movie interaction samples for the model

System retrains ALS model on data with the new inputs System creates movie data for inference (in my case, I sample all movies from the data)

System make rating predictions on all movies for that user

System outputs top N movie recommendations for that user based on the ranking of movie rating predictions.

8 Model Training, Testing and Recommendations

The model is trained on the test data which is the user ratings for movies and after training the model is used to transform the test data which gives the predicted user ratings for the movies. Also we test the model using Root Mean Square Error metric and RMSE was 0.83 for this model. Recommendations are generated from the predicted user ratings and the flow of generating recommendations is given in the workflow of the model. Some of the recommendations on the test data include:
User who watched Toy Story has got recommendation to watch Balto
User who watched Jumanji has got recommendation to watch Zathura.

These are pretty good results obtained from the model and the RMSE of the model on the testing data is 0.83.

9 Exploratory Data Analysis

The following image shows the movies data.

```
movies.show()
```

```
+-------+--------------------+--------------------+
|movieId|               title|              genres|
+-------+--------------------+--------------------+
|      1|    Toy Story (1995)|Adventure|Animati...|
|      2|      Jumanji (1995)|Adventure|Childre...|
|      3|Grumpier Old Men ...|      Comedy|Romance|
|      4|Waiting to Exhale...|Comedy|Drama|Romance|
|      5|Father of the Bri...|              Comedy|
|      6|         Heat (1995)|Action|Crime|Thri...|
|      7|      Sabrina (1995)|      Comedy|Romance|
|      8| Tom and Huck (1995)|  Adventure|Children|
|      9| Sudden Death (1995)|              Action|
|     10|    GoldenEye (1995)|Action|Adventure|...|
|     11|American Presiden...|Comedy|Drama|Romance|
|     12|Dracula: Dead and...|       Comedy|Horror|
|     13|        Balto (1995)|Adventure|Animati...|
|     14|        Nixon (1995)|               Drama|
|     15|Cutthroat Island ...|Action|Adventure|...|
|     16|       Casino (1995)|         Crime|Drama|
|     17|Sense and Sensibi...|       Drama|Romance|
|     18|   Four Rooms (1995)|              Comedy|
|     19|Ace Ventura: When...|              Comedy|
|     20|  Money Train (1995)|Action|Comedy|Cri...|
+-------+--------------------+--------------------+
```

Figure 5: Movies Data

The following image shows the rating data.

```
ratings.show()
```

```
+------+-------+------+----------+
|userId|movieId|rating| timestamp|
+------+-------+------+----------+
|     1|    307|   3.5|1256677221|
|     1|    481|   3.5|1256677456|
|     1|   1091|   1.5|1256677471|
|     1|   1257|   4.5|1256677460|
|     1|   1449|   4.5|1256677264|
|     1|   1590|   2.5|1256677236|
|     1|   1591|   1.5|1256677475|
|     1|   2134|   4.5|1256677464|
|     1|   2478|   4.0|1256677239|
|     1|   2840|   3.0|1256677500|
|     1|   2986|   2.5|1256677496|
|     1|   3020|   4.0|1256677260|
|     1|   3424|   4.5|1256677444|
|     1|   3698|   3.5|1256677243|
|     1|   3826|   2.0|1256677210|
|     1|   3893|   3.5|1256677486|
|     2|    170|   3.5|1192913581|
|     2|    849|   3.5|1192913537|
|     2|   1186|   3.5|1192913611|
|     2|   1235|   3.0|1192913585|
+------+-------+------+----------+
```

Figure 6: Ratings Data

The following image shows the number of users in the data.

```
#Number of Users in the data
q1=spark.sql("""select count(distinct userId) as Number_of_users from ratings_data""")
q1.show()
```

```
+---------------+
|Number_of_users|
+---------------+
|         283228|
+---------------+
```

Figure 7: Number of Users

The following image shows the number of movies in the data.

```
#Number of Movies in the data
q2=spark.sql("""select count(*) as Number_of_movies from movies_data""")
q2.show()
```

```
+-----------------+
|Number_of_movies|
+-----------------+
|            58098|
+-----------------+
```

Figure 8: Number of Movies

The following image shows the range of ratings in the data.

```
#User Rating Range
q3=spark.sql("""select min(rating) as Min_Rating , max(rating) as Max_Rating from ratings_data""")
q3.show()
```

```
+----------+----------+
|Min_Rating|Max_Rating|
+----------+----------+
|       0.5|       5.0|
+----------+----------+
```

Figure 9: Rating Range

The following image shows the number of movies which have no ratings.

```
#Movies which have no rating
q4=spark.sql("""select 58098-count(distinct movieId)  as Num_movies_with_no_rating from ratings_data""")
q4.show()
```

```
+-------------------------+
|Num_movies_with_no_rating|
+-------------------------+
|                     4209|
+-------------------------+
```

Figure 10: Number of Movies with no rating

The following image shows the movies which have no ratings.

```
#List of such movies which don't have a rating
q5=spark.sql("""select movies_data.title, movies_data.genres ,ratings_data.rating
    from movies_data left JOIN ratings_data
    on ratings_data.movieId = movies_data.movieID
    where ratings_data.rating IS null LIMIT 10""")
q5.show()
```

```
+--------------------+--------------------+------+
|               title|              genres|rating|
+--------------------+--------------------+------+
|Baby Blue Marine ...|               Drama|  null|
|Wide Open Spaces ...|   Animation|Comedy|  null|
|Ashes and Blood (...|               Drama|  null|
|Music in the Air ...|Comedy|Musical|Ro...|  null|
|      Muggers (2000)|              Comedy|  null|
|Tarzan's Magic Fo...|Action|Adventure|...|  null|
```

Figure 11: Movies with no rating

The following image shows popular movies in the data which have most of the ratings as 5.

```
#List of movies which have a rating 5 by most users
#Popular Movies
q6=spark.sql("""select ratings_data.movieId ,movies_data.title, count(*) as Num_users_rated_5
           from ratings_data , movies_data
           where ratings_data.movieId=movies_data.movieId
           and rating=5
           group by ratings_data.movieId , movies_data.title
           order by Num_users_rated_5 desc""")
q6.show(truncate=False)
```

```
+-------+-----------------------------------------------------------------------------+-----------------+
|movieId|title                                                                        |Num_users_rated_5|
+-------+-----------------------------------------------------------------------------+-----------------+
|318    |Shawshank Redemption, The (1994)                                             |48762            |
|296    |Pulp Fiction (1994)                                                          |37458            |
|356    |Forrest Gump (1994)                                                          |32009            |
|260    |Star Wars: Episode IV - A New Hope (1977)                                    |31385            |
|593    |Silence of the Lambs, The (1991)                                             |30280            |
```

10 Future Work

The collaborative filtering works on a similar user metric. For the future improvements of the project we can define the similar user metric in such a way that it can give better recommendations than comparing the users only on the rating. These can be defined by using the geographical location, type of work people do, demographics and religion.

11 This project falls under Big Data, Why or How?

This is big data because firstly the size of data is large around 1GB(Volume), it has new data obtained at different speeds for different models(Velocity), and there is different type of data like movies and users (Variety), data is from real world and accurate(Veracity) and it is useful since we saw in the first slide how Netflix generates a huge amount(Value). All these 5 V's define this project as Big Data.