Movie Recommender System Based on Collaborative Filtering Using Apache Spark

Mohammed Fadhel Aljunid and D. H. Manjaiah

Abstract Recently, the building of recommender systems becomes a significant research area that attractive several scientists and researchers across the world. The recommender systems are used in a variety of areas including music, movies, books, news, search queries, and commercial products. Collaborative Filtering algorithm is one of the popular successful techniques of RS, which aims to find users closely similar to the active one in order to recommend items. Collaborative filtering (CF) with alternating least squares (ALS) algorithm is the most imperative techniques which are used for building a movie recommendation engine. The ALS algorithm is one of the models of matrix factorization related CF which is considered as the values in the item list of user matrix. As there is a need to perform analysis on the ALS algorithm by selecting different parameters which can eventually help in building efficient movie recommender engine. In this paper, we propose a movie recommender system based on ALS using Apache Spark. This research focuses on the selection of parameters of ALS algorithms that can affect the performance of a building robust RS. From the results, a conclusion is drawn according to the selection of parameters of ALS algorithms which can affect the performance of building of a movie recommender engine. The model evaluation is done using different metrics such as execution time, root mean squared error (RMSE) of rating prediction, and rank in which the best model was trained. Two best cases are chosen based on best parameters selection from experimental results which can lead to building good prediction rating for a movie recommender.

Keywords Recommender systems · Collaborative filtering · Alternating Least Squares · Apache Spark · Big data · MovieLens dataset

M. F. Aljunid (✉)
Mangalore University, Mangalore, Karnataka, India
e-mail: Ngm505@yahoo.com

D. H. Manjaiah (✉)
Department of Computer Science, Mangalore University, Mangalore
Karnataka, India
e-mail: drmdh2014@gmail.com

V. E. Balas et al. (eds.), *Data Management, Analytics and Innovation*,
Advances in Intelligent Systems and Computing 839,
https://doi.org/10.1007/978-981-13-1274-8_22

1 Introduction

In recent times, big data is becoming one of the newest research interests in the areas of computer science and other related areas. With the possibility of a radical change in companies and organizations that use the information for improving the customer experience and transform their business models. Big data has several features which are volume, velocity, variety, value, and veracity. Big data is facing difficulties in managing using conventional tools, techniques, and procedures. Big data analytics is used for handling bulk quantities of data. It is used to mine and extract patterns, information, and knowledge from the data in an effective way. Big data analytics become an important trend for organizations and enterprises that are interesting in providing innovative ideas for enhancing and increasing their business performance and decision-making. RS are a group of techniques that allow filtering through large samples and information space in order to give suggestion to users when needed. Currently, RS are becoming highly popular and utilized in different areas such as movies, research articles, search queries, news, books, social tags, and music. Furthermore, there are other essential RS basically applicable for specialist, collaborators, funny story, restaurant and hotels, dresses, monetary services, life insurance, passion associates which give online dating services and several other social media such as Twitter, LinkedIn, and Facebook.

RS use a number of different technologies to filter out best suit results and provide to users to satisfy their information need. RS are classified into three broad groups which are content-based systems, collaborative filtering systems, and hybrid recommender system [1]. Content-based systems which try to test the behavior of the item which is labeled as recommended one. It works by learning the behavior of the new users based on their information need presented in objects whereby the user has rated. It is a keyword-specific RS where the keywords are used to illustrate the items. Thus, in a content-based RS, models work in such a way that they recommend users' comparable items that have been liked in the past or is browsing currently. For instance, if a MovieLen user has to browse several comedies movies, then, the RS will classify those movies into the database as getting the most ratings on the comedy varieties. Collaborative filtering system is based on similarity measures between user's information need and the items. The items recommended to a new user are those which were liked by other similar users in previous browsing history. Collaborative filtering algorithm uses an average rating of objects, recognizes similarities between the users on the basis of their ratings, and generates new recommendations based on inter-user comparisons. However, it faces many challenges and limitation such as data sparsity whose role is to the evaluation of large item set. Another limitation is hard to make prediction based on nearest neighbor algorithm, third is scalability in which number of users and number of items both increases, and the last one is cold start where poor relationship among like-minded people. To solve encounters, above mentioned, we moved to other approaches of collaborative filtering, and we landed up on model-based collaborative filtering [2]. Hybrid RS performs their tasks by

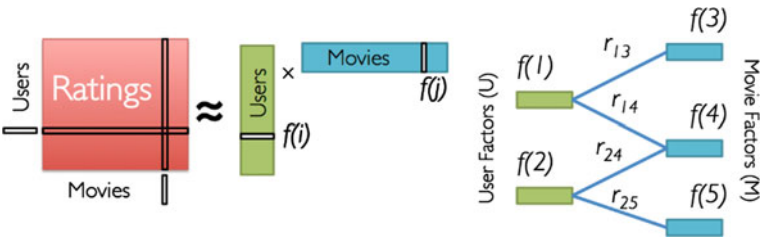

Fig. 1 Low rank factorization matrix [3]

considering the combining behavior of content-based and collaborative filtering techniques in such a way that it suits a particular item. Hybrid recommended system is regarded as the most frequently used RS system considered by many companies due to its ability to eliminate any weakness that might have arose when one RS is employed and in addition, its strength is the composite of more than two RS.

The main focus of this work is collaborative filtering system. It is well known that collaborative filtering could be described as a procedure whereby automatic prediction (i.e., filtering) about the interests of a user is made by gathering taste or preferences information from many users. The unexpressed assumption of the collaborative filtering approach can be best explained, viz., supposing a person A has similar opinion with person B on a particular issue, the assumption is that person A will be more likely to have the same opinion as person B on a different issue X did the opinion on X of a person chosen randomly [3]. Take for an instance the movie "RS" depicted in Fig. 1 which started with a matrix whose entries are movies rated by users. Both user (shown in green) and a particular movie (shown in blue) are represented each by column and rows respectively. Owing to the fact that not all users have rated all movies, all the entries in the matrix are unknown, which necessitate the need for collaborative filtering. There are ratings for only a subset of the movies for each user. With collaborative filtering, the idea is to approximate the rating matrix by factorizing it as the product of two matrices. That is the one that describes properties of each user (shown in green), and the other describing properties of each movie.

The minimization of the error for the users/movies pairs was chosen as the basis for the selection of the two matrices. The alternating least squares algorithm (ALS) which achieves this by randomly filling the user's matrix with values before optimizing the value of the movies was used for this purpose. The value of the user's matrix is optimized with the movie's matrix being kept constant (Fig. 1). Owing to a fixed set of user factors (i.e., values in the user's matrix), known ratings are employed to find the best values by optimizing the movie factors, written on top of the figure. The best user factor with the fixed movie factors is sleeted. This paper, reports for the first time, a movie recommendation system based on collaborative filtering using apache spark. The performance analysis and evaluation of proposed approach are performed on a MovieLens dataset. From the results obtained, it is concluded that the selection of parameters of ALS algorithms can affect the performance of recommender engine to be used.

The remainder of this paper is organized as follows: related work is provided in Sect. 2. Section 3 introduces the proposed movie recommender system using collaborative filtering with ALS algorithm while the experimental study is introduced in Sect. 4. Finally, the paper conclusion is presented in Sect. 5.

2 Related Work

So far, several researchers introduced and presented research in the area of building recommendation systems. Wei et al. [4] proposed a hybrid recommender model to address the cold start problem, which explores the item content features, learned from a deep learning neural network and applies them to the timeSVD++ CF model. A hybrid recommendation model is proposed which combines a time-aware model timeSVD++ with a deep learning architecture SDAE to address the cold start problem of collaborative filtering recommendation models. Kupisz and Unold [5] developed and compared item-based collaborative filtering algorithm using two cluster computing frameworks normally Hadoop's disk-based MapReduce paradigm and Spark's in-memory based RDD paradigm. In order to enhance the reliability, scalability, and to improve processing ability of large-scale data, Zeng et al. [6] proposed PLGM. In their work, two matrix factorization algorithms were considered, which are ALS and SGD. The parallel matrix factorization based on SGD was implemented on spark and was compared with ALS in MLib for its performance. The advantage and disadvantage of each model based on test results were analyzed. A variety of profile aggregation approaches were studied and the model which gives the best result was adopted. Models such as PLGM and LGM were studied in terms of efficiency and accuracy. Dianping, Lakshmi et al. [7] used item-based collaborative filtering techniques. In this method, they first inspect the user item rating matrix and they categorize the relationships among different items, and they utilize these relationships so as to figure out the recommendations for the user. A new concept namely movie swarm mining was proposed by Halder et al. [8] using format frequent item mining and two pruning rules. It addresses the problem of item recommendation and thus gives an idea about the user interests and famous movies trend. This technique can be very helpful for movie producers to manage their new movies. In addition to this, a new algorithm was proposed to recommend movies to a new user. A scalable method for building recommender systems based on similarity join has been proposed by Dev et al. [9]. MapReduce framework was used to design the system in order to work with big data applications. The unnecessary computation overhead such as redundant comparisons in the similarity computing phase can significantly be reduced by the system using a method called extended prefix filtering (REF). Chen et al. [10] used co-clustering with augmented matrices (CCAM) to design several methods including a heuristic scoring, traditional classifier, and machine learning to build a recommendation system and integrate content-based collaborative filtering for a hybrid recommendation system. Similarly, a collaborative filtering algorithm based on the ALS, as a powerful

matrix decomposition algorithm, has been proposed by Wilkinson and Schreiber [11]. They found out that it can be awesome to extend to the distributed computing and solve the data sparse problem.

3 Proposed Movie Recommender System

This section provides the idea of the proposed system. The proposed system is a movie recommender system based on ALS using Apache Spark. The novelty of this work is based on the selection of parameters of ALS algorithms that can affect the performance of building of a movie recommender system.

3.1 Proposed System Block Diagram

In this work, we apply user's ratings from the datasets the popular website like IMDB, Rotten Tomatoes, MovieLen, and Time Movie Ratings. This dataset is available in many formats such as CSV file, text file, and databases. We can either stream the data live from the websites or download and store them on our local file system or HDFS. Spark streaming is used to stream real-time data from the various source like Twitter, the stock market, and geographical system and perform powerful analytics to businesses. It used for processing real-time streaming data. We use collaborative filtering (CF) to predict the ratings of users for particular movies based on their ratings for other movies. Then collaborate this with another user's rating for that particular movie. We train the ALS algorithm using MovieLen data and get the results from the machine learning model. We use spark SQL's data frame, dataset, and SQL service to store the data. The result of the machine learning model is stored in RDBMS so that the web application can display the recommendation to a particular use. The results of the movie recommendation system are stored in our local drive. We store the recommendation movies along with the ratings in a text file and CSV file formats. We prefer storing the result into an RDBMS system so as to access it directly from the web application and display recommendation and top movies as shown in Fig. 2.

3.2 Proposed System Steps

This subsection provides the steps of applying the ALS algorithm on MovieLens datasets for train and test the selection of best parameter when building a movie recommendation system.

Fig. 2 Proposed movie recommendation system using CF with ALS

Movie Recommendation System using CF with ALS

Input: MovieLens Dataset
Output: Top Recommended Movies.
Procedures:
Procedure 1:Parsing and loading datasets
Procedure 2: Recognize the user as new or regular.
If new user goto **Procedure** 5
Procedure 3: Load training and test data into the table (userId, movieId, rating)
def parse_the_rating(line):
x = line.split()
return (int (x [0]), int (x [1]), float (x [2]))
training = sc.TrainingFile("__").map(parse_the_Rating).cache()
test = sc.Testfile("__").map(parse_the_Rating)
Procedure 4: Train the recommender model.
New_model= ALS.train (rank, train, iteration)
Procedure 5:Create predictions on (user, movie) pairs from the test data
Predict = New_model.predictAll (test.map(lambda x: (x[0], x[1]))
Procedure6: Adding new user ratings
Procedure 7: Display top N recommended movies.
Procedure 8: Save the New_model

4 Experimental Study

This section presents the experimental setup and results in discussion and analysis.

4.1 Apache Spark

Apache Spark [12] is a rapid and general-purpose cluster computing system. It introduces high-level application programming interfaces (APIs) using

programming languages such as Java, Python, Scala, and R, and has an engine that supports general execution graphs. It also supports a good set of higher level tools involving Spark SQL for structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming for real-time applications. It was built on top of Hadoop and MapReduce and extends the MapReduce Model to efficiently use more types of computations. Spark application runs as a separate set of process on the cluster. All of the distributed processes are coordinated by a SparkContext object in the drive program. SparkContext connects to one type of cluster manager (standalone/Yarn/Mesos) for resource allocation across clusters. Cluster manager provides executors, which are essentially JVM process to run the logic and store application data. Then the SparkContext object sends the application code (jar files/python scripts) to executors. Finally, the SparkContext executes tasks in each executor.

4.2 Data Preprocessing

The dataset which is used in this work is MovieLens dataset. This dataset contains 24 million ratings and 670,000 tag applications applied to 40,000 movies by 260,000 users. This dataset contains three files called ratings.csv, movies.csv and tags.csv. ratings.csv contains tree column (userId, movieId, rating). While movies. csv contains movieId, title, genres. The genres have the format: Genre1, Genre2, Genre3. The tags file (tags.csv) has the format: userId, movieId, tag, timestamp and finally, the links.csv file has the format: movieId, imdbId, tmdbId. We can split the data into three portions which are training, validation, and test data to parse their lines once they are loaded into RDDs. Parsing the movies and rating files yields two RDDs: For each row in the ratings dataset, we have created a vector of (userId, movieId, rating). During preprocessing, we have dropped the timestamp attribute because we do not need it for this recommender. Similarly, each row in the movies dataset, we have created a vector of (movieId, title). We have dropped the genres attribute because we do not use it for this recommender.

In order to determine the best ALS parameters for our experiments, we need to break up the ratings RDD dataset into three pieces as follows: a training set which we will use 60% of the data to train models, validation set, which used 20% of the data to choose the best model and test set, which used 20% of the data for our experiments to randomly split the dataset into the multiple groups.

4.3 Experimental Environment

The test has been done on a machine which contains the subsequent descriptions P. A machine with Ubuntu 14.04 LTS, 4 GB memory, and Intel® Core™ i5-2400 CPU @ 3.10 GHz × 4 processor as well as a hard disk of 500 GB. In this machine,

Apache Spark with version 2.1.1 is installed and is used to develop the proposed system. The dataset which is used in research work is MovieLens dataset [13]. In the proposed model, root mean squared error (RMSE) is used as a performance measure. RMSE works by measuring the difference between error rate a user gives to the system and the predicted error by the model. Equation (1) depicts how RMSE works on movie recommender system.

$$\text{RMES} = \sqrt{\sum_{i=0}^{n} \frac{\left(x_{ui} - y_{gi}\right)^2}{n}} \tag{1}$$

whereby x_{ui} is the rating that user u gives to an item i in the experimental data, y_{gi} is a predicted rating that the movie that user u gives to an item and where n is the number of ratings in the test data.

4.4 Experimental Results Analysis and Discussion

Recommender system (RS) is becoming growingly popular. In this work, Apache Spark is used to demonstrate an efficient parallel implementation of a collaborative filtering method using ALS. ALS is used for dimensionality reduction purpose which helps in overcoming the limitations of collaborative filtering such as data sparsity and scalability. The challenges of data sparsity are appearing in numerous situations, specifically, another problem, when a new an item or user has just added to the system, it is difficult to find similar ones since there is no sufficient information, this problem is called cold start problem [14, 15]. When selecting the ALS algorithm as a part of building the proposed movie recommender system, there is basic parameter through them can determine the best rating of users for given movies. These parameters are Rank, Iterations, and Lambda.

The contribution of this paper is to study and determine the selection of parameters that affect the performance of ALS model in building a movie recommender system because from literature study, it is found that little research work focused on the study of the selection of ALS's parameters that can affect its performance in building a movie recommender engine using Apache Spark. The parameters, lambda, and iterations are used in order to control and adjust the predicting capability of matrix factorization which is depending on ALS technique which in turn affect the evaluation of movie RS. The iterations and lambda parameters are used as follows: Lambda which specifies the regularization parameter in ALS and iterations in which the proposed model should run the specified number of iterations. The ALS algorithm achieves its optimal solution between 5 and 20 iterations.

The parameters lambda and iteration in ALS model are used with different thresholds to realize the effects of matrix factorization performance on the performance of recommendation results and thus take the most appropriate parameters for

the following test setups. Tables 1, 2 and 3 show the performance of movie recommendation engine based on ALS under different values of lambda and iteration. Table 1 illustrates the execution of time with the changes of lambda with iterations parameters of ALS model, while Table 2 the rank of best-trained model with the changes of lambda with iterations parameters of ALS algorithm, finally Table 3 indicates the RMSE with the changes of lambda with iterations parameters of ALS model. The results presented in Table 1 indicate that when lambda is set to **0.6** and iteration set is **10**, the time value is minimum which is **1.41323 s**, and rank value is **8** as shown in Table 2. Moreover, the RMSE register for this rating is **1.07424** as indicated in Table 3. On the other hand, as it is indicated in Table 1 when lambda is set to **0.2** and iteration is **15**, running time becomes **1.463743** and rank is **12** for this item as shown in Table 2. The RMSE value for this item is the minimum, which is **0.9167**, as presented in Table 3.

As mentioned above, the analysis for movie recommendation system is done using three quality metrics which are RMSE, time, and rank. Using these three metrics, two cases are achieved as shown in Table 4, case 1 with high time and low RMSE rate while the case 2 with low time and high RMSE rate. According to results in Table 4, the prediction for Top 25 movies is shown in Figs. 3 and 4.

Table 1 Time of matrix factorization using lambda and iteration parameters

Lambda	Iteration				
	5	10	15	20	25
0.1	1.489	1.454	1.473	1.469	1.512
0.2	1.485	1.437	**1.464**	1.438	1.514
0.3	1.472	1.481	1.4671	1.441	1.494
0.4	1.658	1.486	1.476	1.495	1.473
0.5	1.431	1.492	1.468	1.478	1.528
0.6	1.615	**1.413**	1.442	1.459	1.480
0.7	1.443	1.475	1.471	1.446	1.543
0.8	1.554	1.470	1.459	1.449	1.527
0.9	1.491	1.478	1.482	1.471	1.446

Table 2 Rank of matrix factorization using lambda and iteration parameters

Lambda	Iteration				
	5	10	15	20	25
0.1	4	12	4	4	4
0.2	8	12	**12**	12	12
0.3	4	8	8	8	8
0.4	4	8	8	8	8
0.5	8	8	8	8	8
0.6	8	**8**	8	8	12
0.7	12	12	4	4	12
0.8	12	12	4	4	4
0.9	12	4	4	4	4

Table 3 RMSE of matrix factorization using lambda and iteration parameters

Lambda	Iteration				
	5	10	15	20	25
0.1	0.947	0.942	0.940	0.938	0.938
0.2	0.919	0.917	**0.9167**	0.917	0.917
0.3	0.941	0.941	0.941	0.941	0.941
0.4	0.975	0.980	0.980	0.981	0.981
0.5	1.018	1.024	1.024	1.024	1.024
0.6	1.069	**1.074**	1.074	1.074	1.074
0.7	1.127	1.130	1.131	1.131	1.131
0.8	1.192	1.193	1.193	1.193	1.193
0.9	1.261	1.261	1.261	1.261	1.261

Table 4 Two cases for selecting parameters for ALS

Metrics	Case	
	Case 1	Case 2
Time	1.41323	**1.463743**
Rank	8	**12**
RMSE	1.07422	**0.9167**

```
 My 25 highest rated movies as predicted (for movies with more than 75 reviews):
+-------+------+----------+-------+-----+-----------------+--------------------+
|movieId|userId|prediction|movieId|count|          average|               title|
+-------+------+----------+-------+-----+-----------------+--------------------+
| 159817|     0| 3.8603234| 159817|  193| 4.450777202072539| Planet Earth (2006)|
|    318|     0| 3.8078098|    318|84455|  4.43308862707951|Shawshank Redempt...|
| 160718|     0|  3.797955| 160718|   88| 4.232954545454546|        Piper (2016)|
| 134849|     0|  3.745416| 134849|  120| 4.041666666666667|   Duck Amuck (1953)|
|    858|     0| 3.7449322|    858|53547| 4.343623358918333|Godfather, The (1...|
|  26048|     0| 3.7382393|  26048|   76|3.9539473684210527|Human Condition I...|
|     50|     0| 3.7262065|     50|56348| 4.308635621494996|Usual Suspects, T...|
| 100044|     0|  3.722707| 100044|   97|  4.15979381443299| Human Planet (2011)|
| 142115|     0| 3.7207592| 142115|  150|             4.31|The Blue Planet (...|
|   2019|     0| 3.7023807|   2019|13394| 4.256196804539346|Seven Samurai (Sh...|
|  94466|     0| 3.7008467|  94466| 2477|4.2426322163907955| Black Mirror (2011)|
| 162376|     0| 3.6991944| 162376|  703| 4.312944523470839|     Stranger Things|
|  77658|     0| 3.6968207|  77658| 1740|4.1818965517241375|       Cosmos (1980)|
|    527|     0| 3.6902485|    527|63889|4.2759629983252205|Schindler's List ...|
|    904|     0| 3.6900668|    904|20443| 4.238590226483393|  Rear Window (1954)|
|  26082|     0| 3.6886785|  26082|  542| 4.100553505535055|Harakiri (Seppuku...|
|   1178|     0| 3.6881678|   1178| 4064|  4.20435531496063|Paths of Glory (1...|
|    922|     0| 3.6870744|    922| 7606|   4.2058900867736|Sunset Blvd. (a.k...|
|   7926|     0| 3.6856618|   7926|  703| 4.105263157894737|High and Low (Ten...|
|   1212|     0| 3.6842163|   1212| 7412| 4.214854290339989|Third Man, The (1...|
|   6669|     0| 3.6837091|   6669| 1340|4.1093283582089555|        Ikiru (1952)|
|   3030|     0| 3.6817052|   3030| 3988| 4.186434302908726|      Yojimbo (1961)|
| 127052|     0|  3.675985| 127052|  160|              4.1|Operation 'Y' & O...|
|   3435|     0|  3.672548|   3435| 5421| 4.203836930455635|Double Indemnity ...|
|   2920|     0| 3.6651812|   2920| 1115|4.1246636771300444|Children of Parad...|
+-------+------+----------+-------+-----+-----------------+--------------------+
only showing top 25 rows
```

Fig. 3 Prediction of top 25 movies for case 1

```
 My 25 highest rated movies as predicted (for movies with more than 75 reviews):
+-------+------+----------+-------+-----+-----------------+--------------------+
|movieId|userId|prediction|movieId|count|          average|               title|
+-------+------+----------+-------+-----+-----------------+--------------------+
| 159817|     0| 4.6047864| 159817|  193| 4.450777202072539| Planet Earth (2006)|
|    318|     0| 4.5063915|    318|84455|  4.43308862707951|Shawshank Redempt...|
| 160718|     0|  4.466501| 160718|   88| 4.232954545454546|        Piper (2016)|
| 134849|     0| 4.4416428| 134849|  120| 4.041666666666667|   Duck Amuck (1953)|
|    858|     0|  4.429425|    858|53547| 4.343623358918333|Godfather, The (1...|
|    527|     0|  4.413663|    527|63889|4.2759629983252205|Schindler's List ...|
| 100044|     0|  4.395683| 100044|   97|  4.15979381443299| Human Planet (2011)|
| 100553|     0| 4.3911767| 100553|  239| 4.062761506276151|Frozen Planet (2011)|
|     50|     0|  4.387128|     50|56348| 4.308635621494996|Usual Suspects, T...|
|    912|     0|  4.383373|    912|29001| 4.220164821902693|   Casablanca (1942)|
|    904|     0|  4.382594|    904|20443| 4.238590226483393|  Rear Window (1954)|
|  77658|     0| 4.3616266|  77658| 1740|4.1818965517241375|       Cosmos (1980)|
|  77177|     0| 4.3589344|  77177|   91| 3.791208791208791|   Wild China (2008)|
|   1203|     0| 4.3545713|   1203|15938| 4.224777261889823| 12 Angry Men (1957)|
|   2019|     0| 4.3540235|   2019|13394| 4.256196804539346|Seven Samurai (Sh...|
| 108583|     0| 4.3512716| 108583|  921| 4.074918566775244|Fawlty Towers (19...|
| 127052|     0|  4.349377| 127052|  160|              4.1|Operation 'Y' & O...|
| 142115|     0| 4.3470507| 142115|  150|             4.31|The Blue Planet (...|
|  44555|     0|  4.343796|  44555| 8241|4.1996723698580265|Lives of Others, ...|
|   1207|     0| 4.3423305|   1207|16738| 4.162474608674872|To Kill a Mocking...|
|    908|     0|  4.339527|    908|18467| 4.208669518600748|North by Northwes...|
|  93040|     0| 4.3393526|  93040|  380| 4.044736842105263|Civil War, The (1...|
|  94466|     0| 4.3387938|  94466| 2477|4.2426322163907955| Black Mirror (2011)|
|  82143|     0| 4.3382115|  82143|  295| 3.906779661016949|Alone in the Wild...|
| 162376|     0| 4.3322854| 162376|  703| 4.312944523470839|     Stranger Things|
+-------+------+----------+-------+-----+-----------------+--------------------+
only showing top 25 rows
```

Fig. 4 Prediction of top 25 movies for case 2

In general, the lowest value of the RMSE is considered the best case for prediction in building recommendation system. Therefore, we will adopt the second case because the value of the RMSE is smaller compared to the value in the first case as well as adopt the second case as the best case because there is no significant difference in the amount of time execution between the two cases. Now, we can get the top recommended movies by using the second case. Finally, we concluded that from these results the best case is the second case which has the best value for RMSE, which can be useful for building recommendation engines for predicting the top 25 ranked movies.

5 Conclusion and Future Work

Movie recommender system plays a significant role in identifying a set of movies for users based on user interest. Although many move recommendation systems are available for users, these systems have the limitation of not recommending the movie efficiently to the existing users. This paper presented a movie recommender system based on collaborative filtering using Apache Spark. From the results, the selection of parameters of ALS algorithms can affect the performance of building of a movie recommender engine. System evaluation is done using various metrics such as execution time, RMSE of rating prediction, and rank in which the best

model was trained. Two best cases are chosen based on best parameters selection from experimental results which can lead to building god prediction rating for a movie recommender engine. From these cases, the lowest value of the RMSE is considered the best case for prediction in building movie recommendation system. Therefore, the second case is recommended to be used since the value of the RMSE is smaller compared to the value in the first case as well as adopt the second case as the best case, because there is no significant difference in the amount of time execution between the two cases. Finally, we concluded that from these results that the best case is the second case which has the best value for RMSE, which can be useful for building recommendation engines for predicting the top 25 ranked movies. In the future work, we plan to develop and improve a new loss function because of the shortcomings of the recommender system algorithm based on ALS model based on the parameter of the best case which has the best value for RMSE using Apache Spark.

References

1. Verma, J. P., Patel, B., & Patel, A. (2015). Big data analysis: Recommendation system with Hadoop framework. In *2015 IEEE International Conference on Computational Intelligence & Communication Technology (CICT)*. IEEE.
2. Katarya, R., & Verma, O. P. (2016). A collaborative recommender system enhanced with particle swarm optimization technique. *Multimedia Tools and Applications, 75*(15), 9225–9239.
3. https://docs.databricks.com/_static/notebooks/cs100x-2015-introduction-to-big-data/module-5–machine-learning-lab.html.
4. Wei, J., et al. (2016). Collaborative filtering and deep learning based hybrid recommendation for cold start problem. In *2016 IEEE 14th International Conference on Dependable, Autonomic and Secure Computing, 14th International Conference on Pervasive Intelligence and Computing, 2nd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE.
5. Kupisz, B., & Unold, O. (2015). Collaborative filtering recommendation algorithm based on Hadoop and Spark. In *2015 IEEE International Conference on Industrial Technology (ICIT)*. IEEE.
6. Zeng, X., et al. (2016). Parallelization of latent group model for group recommendation algorithm. In *IEEE International Conference on Data Science in Cyberspace (DSC)*. IEEE.
7. Ponnam, L. T., et al. (2016). Movie recommender system using item based collaborative filtering technique. In *International Conference on Emerging Trends in Engineering, Technology, and Science (ICETETS)*. IEEE.
8. Halder, S., Sarkar, A. M. J., & Lee, Y.-K. (2012). Movie recommendation system based on movie swarm. In *2012 Second International Conference on Cloud and Green Computing (CGC)*. IEEE.
9. Dev, A. V., & Mohan, A. (2016). Recommendation system for big data applications based on set similarity of user preferences. In *International Conference on Next Generation Intelligent Systems (ICNGIS)*. IEEE.
10. Chen, Y.-C., et al. (2016). User behavior analysis and commodity recommendation for point-earning apps. In *2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE.

11. Zhou, Y. H., Wilkinson, D., & Schreiber, R. (2008). Large scale parallel collaborative filtering for the Netflix prize. In *Proceedings of 4th International Conference on Algorithmic Aspects in Information and Management* (pp. 337–348). Shanghai: Springer.
12. https://spark.apache.org/docs/latest/. Accessed March 10, 2017.
13. https://grouplens.org/datasets/movielens/. Accessed May 15, 2017.
14. Delgado, J. A. (2000, February). *Agent-based information filtering and recommender systems on the internet* (Ph.D. thesis). Nagoya Institute of Technology.
15. Mooney, R. J., & Roy, L. (1999). Content-based book recommendation using learning for text categorization. In *Proceedings of the Workshop on Recommender Systems: Algorithms and Evaluation (SIGIR '99)*. Berkeley, CA, USA.