Movie Recommendation System with Spark Machine Learning on Amazon Web Service EMR using Zeppelin

EE 542

Jiawei Wu

*Abstract*- This project aimed to develop a movie recommendation system with Spark ML based on AWS EMR using Zepplin. We try to scale the algorithm so it would be able to handle a data set of 10,000,000 ratings by using Spark ML. This project is implemented based on a model called Collaborative Filtering. It takes a set of movies that have highest ratings and recommending them to the other users who also rated high on them. The root mean squre error (RMSE) is used to represent the deviation between predicted and observed values.

SUMMARY

A recommendation system is a information filtering system which design to predict the performance and ratings that user would give to an item. Recommendation system brings profits to the online services such as Amazon, Youtube, Yelp, Netfilx. A recommendation system is a info Recommender systems typically produce a list of recommendations in one of two ways – through collaborative filtering or through content-based filtering (also known as the personality-based approach). Collaborative filtering approaches build a model from a user's past behaviour (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users.[1]Collaborative filtering is absed on large amount of data to analysis that people agreed on the past and predict the future based on the past record. For instance, if Amazon obersed that large amount of customers who buy a phone and at the same time, they also purchase a USB adapter, then Amazon can recommend the USB adapter to the new customer with the phone added to their shopping cart. However, every user has their own perferences and ratings for a specific items. In order to predict the new results, we have to get sufficient amount of data which belongs to the past user history. In our project, we used collaborative filtering to approch the solution.

Collaborative filtering (CF) predicts user preferences in item selection based on the known user ratings of items. As one of the most common approach to recommender systems, CF has been proved to be effective for solving the information overload problem. CF can be divided into two main branches: memory-based and model-based. Most of the present researches improve the accuracy of Memory-based algorithms only by improving the similarity measures. [2]Collaborative Filtering makes recommendations for users based on a hugh amount of data collected for users based on perferences and ratings. The idea is that if two users perfer the same item, then the items that one user liked but has not been discovered by the other users would be recommend to him. We used Alternating Least Square (ALS) algorithm to analysis the data.

In our project, the platform that we used is Amazon Web Service EMR based on Apache Spark Machine Learning. By using Spark, it can easily anlysis the big data through Spark machine learning algorithm library. By using Apache Zeppelin, we can easily interact with our data, running in an interactive node and analyzing the data step by step. The EMR makes easier to lunch a instance with different application types. Thus, the combination of EMR to create a Hadoop cluster and install Spark, Zeppelin, and Ganglia on it, which Spark provides powerful analysis ability for large datasets and Zeppelin to provide a notebook interface for data visualization. With Spark Machine Learning, it includes a lot of poplular Machine Learning algorithms that can be easily implemented for data analysis.

Problem Statement, Formulation, and Theories

A. Problem Statement

Recommender system is a filtering system which gives predictions according to feedbacks and preferences of users. For this project, this recommender system gives some recommendations of movies based on the ratings and preferences of users, which is the idea of machine learning. It uses one of Collaborative Filtering techniques to make predictions, which is Alternating Least Squares algorithm. Then it uses Root Mean Square Error function to pick the best prediction model. When the recommender system gets data, it uses 60% for training, 20% for validation, and 20% for testing. It runs the training process 8 times by using Alternating Least Squares algorithm. And it checks the performance by using Root Mean Square Error function. Then it selects the best training model to make predictions for different users or different types of movies [3].

B. Theories and Algorithms

* Collaborative Filtering:

Collaborative Filtering is a method which takes ratings of other users and items and specific user history, then gives recommendations of items, which the user did not see before. It is supposed that the preferences of items from other users could be taken as recommendations for the users did not see before. It is important that the users and items do not matter in the recommender system. However, the value of ratings and the ratings from which user do matter. And the preferences of users are used for many users with similar behaviors. [4]

* Alternating Least Squares:

Matrix factorization is used for reducing dimensions, which could reduce features but keep useful data. When there are large matrices of users and items, it could produce a smaller matrix with relations between users and items, which is Alternating Least Squares algorithm. [5]

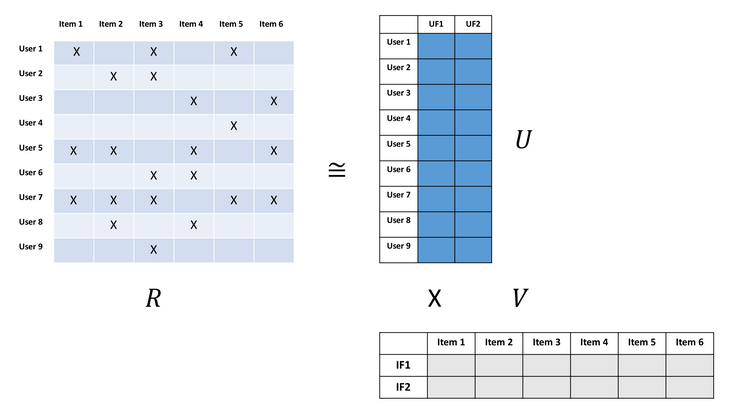


Figure 1: Matrix Factorization [5]

As Fig. 1, there are M users and N items. And matrix R of size MxN shows ratings of users. This matrix is large but only some entries are useful. By matrix factorization, this matrix could be divided into two small matrices. One is matrix U of size MxK with user feature from each user. The other is matrix V of size KxN with item feature of each item. Thus, the product of U and V is the approximation of R, but both of them are more useful than R. For getting U and V, Alternating Least Squares algorithm is used. Since the algorithm could get one feature once, several features could be gotten at the same time. First, U is used to solve V. Second, the result of V is used to solve U. Then this process is iterated until converged as the approximation of R. Eventually, the product of U and V could be used to make predictions for items which users did see before. [5]

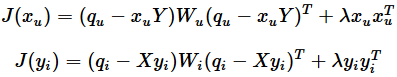


Where r is the rating of user u for item i. [4]

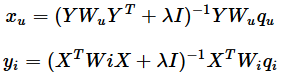
For m users and n items, there is a matrix of factors expressing movies. And the factor vector of each movie expresses the features of the movie. Without knowing the type of the movie, there is also a factor vector of each user. As the factor matrix of movies  and the factor matrix of users  there are still two unknown parameters. With Alternating Least Squares, Y is evaluated by X and X is also evaluated by Y. With several iterations, the results are converged, which means X and Y are similar. As there are never full data of users or movies, predictions are needed to fill the empty entries of ratings for movies. Thus, the data of the movies with known ratings are used, besides the movies with no ratings. And the weight matrix wui is shown: [4]



The cost functions are minimized:



The regularization is used to fit data best. The parameters are regularized by taking cross validation of data to make better performance [4]. With all data, factor vectors are shown:



Where  and  are diagonal matrices. As both matrices are regularized, the algorithm will be better [4].

C.Metric

Root Mean Square Error:



Where rij, rij\* are the real and calculated rating from user j for movie i. T is the test set, N is the number of ratings in the test set. The j and i are index of user and movie [6].

D. Problem Formulations

Definitions of Parameters [6]:

nn = number of users

nm = number of movies

r(i, j) = 1 if user j rated movie i

y(i, j) = rating of movie i from user j

θ(j) = parameter vector of user j

x(i) = feature vector of movie i

(θ(j))T(x(i)) = predicted rating of movie i for user j

Steps [6]:

Getting parameters of all users:



Finding features of movies:

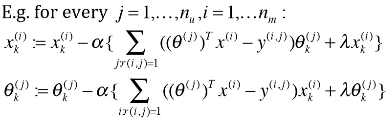


Minimizing features and parameters:



Implementation [6]:

1. Initialize  to small random numbers
2. Minimize  using gradient descent



1. With parameters θ of users and features x of movies, predict rating by θTx

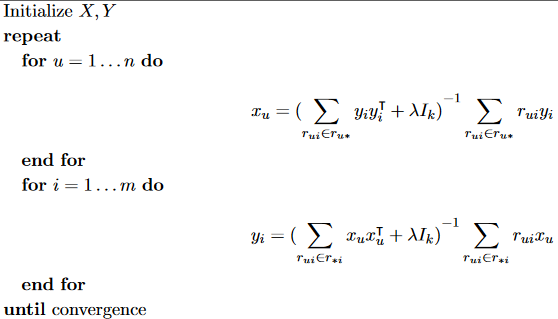


Figure 2: Metrix Factorization [10]

Experimental Settings

A. AWS

The following steps are the tutorial of how to set up the AWS EMR configuration.

* Lunch Spark on EMR

1. On the EMR console, choose Create cluster
2. Choose Go to advanced options
3. In the Software configuration section, in the Software Configuration, choose Release emr-5.10.0 with Hadoop 2.7.3, Zeppelin 0.7.3, Ganglia 3.7.2, Hive 2.3.1, and Spark 2.2.0
4. In the software settings, enter the configuration shown in figure below.
5. In the Hardware configuration, change the instance type to e3.xlarge
6. In the Security and Access, choose an EC2 key pair
7. Create the cluster

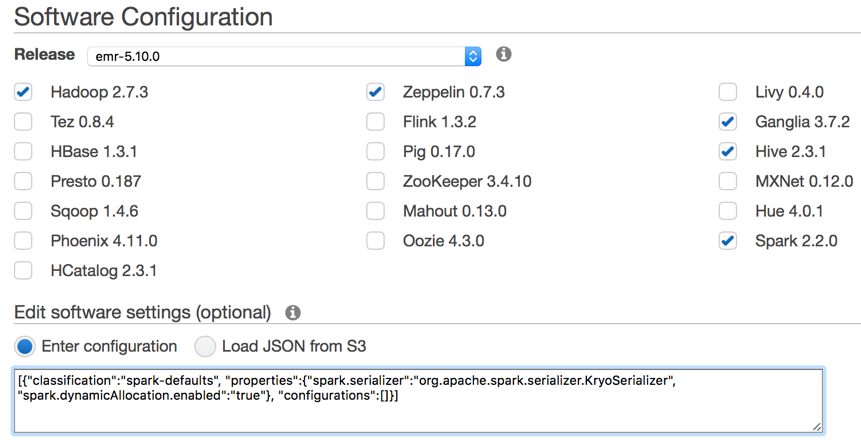


Figure 3: Software Configuration

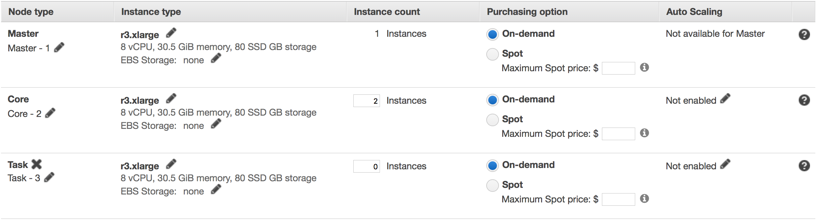


Figure 4: Instance Type

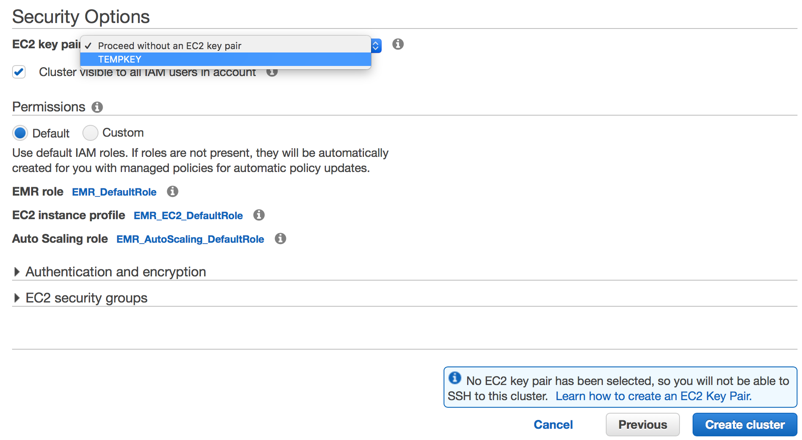


Figure 5: Security Option

* Connect to your EMR Cluster

Connect to the EMR cluster by the following command: ssh -i ~/<YOUR-KEY-FILE>.pem" hadoop@ec2-<MASTER-PUBLIC-IP>.<REGION>.compute.amazonaws.com



Figure 6: Connect to the Master Node Using SSH

* Connect to the Zeppelin Notebook

Connect to the Zeppelin Notebook by enter the following command: ssh -i <YOUR-KEY-FILE.pem> -N -L 8157:ec2-<MASTER-PUBLIC-IP>.<REGION>.compute.amazonaws.com:8890 hadoop@ec2-<MASTER-PUBLIC-IP>.<REGION>.compute.amazonaws.com

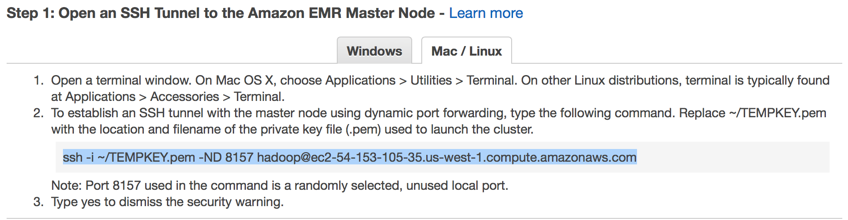


Figure 7: Open and SSH Tunnel to the EMR Master Node

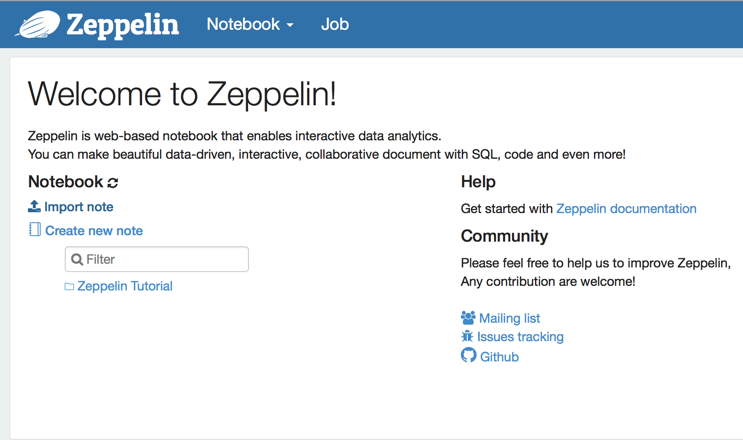


Figure 8: Zeppelin Interface

* Build the Recommender System with Spark ML

1. Upload the datasets to the S3 bucket, which the datasets can be downloded from Movielens website: wget http://files.grouplens.org/datasets/movielens/ml-10m.zip
2. Upload the note.json file to Zeppelin
3. Run the paragraph to see the outputs

B. Software Tools

Software tools that need for this project is terminal for Mac OS and Linux or Putty for Windows. Then install the AWS Command Line Interface and its dependencies on pip, a package manger for Python, after that login to EMR clusters through terminal.

C. Dataset

The datasets that we used is from Movielens website. There are a datasets for movie ratings such as 100K, 1M, 10M, and 20M. The project conducted on 10M size of the dataset. This data set contains 10,000,054 ratings and 95,580 tags applied to 10,681 movies by 71,567 users of the online movie recommender service MovieLens. Users were selected at random for inclusion. Each user is represented by an id, and no other information is provided. The data are contained in three files, movies.dat, ratings.dat and tags.dat. Also included are scripts for generating subsets of the data to support five-fold cross-validation of rating predictions. [7] We spilit the dataset into three partitions: training, vlidation, and testing by sampling ramdomly with a ratio of 60%, 20%, and 20% respectively.

The datasets can be downloaded from the grouplens site (for example using command line to download: wget http://files.grouplens.org/datasets/movielens/ml-10m.zip). After downloading the datasets, unzip the file and upload the file to newly created S3 bucket under s3://ee542proj/input.

The file contents are described as following:

* User Ids

Movielens users were selected at random for inclusion. Their ids have been anonymized. Users were selected separately for inclusion in the ratings and tags data sets, which implies that user ids may appear in one set but not the other. The anonymized values are consistent between the ratings and tags data files. That is, user id n, if it appears in both files, refers to the same real MovieLens user. [7]

* Ratings Data File Structure

All ratings are contained in the file ratings.dat. Each line of this file represents one rating of one movie by one user, and has the following format: UserID::MovieID::Rating::Timestamp

The lines within this file are ordered first by UserID, then, within user, by MovieID. Ratings are made on a 5-star scale, with half-star increments. Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970. [7]

* Tags Data File Structure

All tags are contained in the file tags.dat. Each line of this file represents one tag applied to one movie by one user, and has the following format:UserID::MovieID::Tag::Timestamp

The lines within this file are ordered first by UserID, then, within user, by MovieID. Tags are user generated metadata about movies. Each tag is typically a single word, or short phrase. The meaning, value and purpose of a particular tag is determined by each user. Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970. [7]

* Movies Data File Structure

Movie information is contained in the file movies.dat. Each line of this file represents one movie, and has the following format: MovieID::Title::Genres

MovieID is the real MovieLens id. Title is the name of the movie. Genres are a pipe-separated list, and are selected from the following: Action, Adventure, Animation, Children's, Comedy,Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western [7]

D. App Codes

The recommender system is implement on AWS EMR Zeppelin notebook.

Attached Document: note.json

Results and Analysis

Plots & Tables

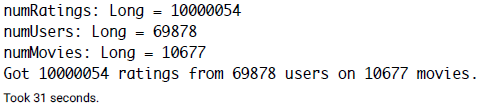


Figure 9: Data Summary

As Fig. 9 shows, there are 69878x10677=746087406 entries, but there are only 10000054 ratings. So only 1.34% entries are useful. With Alternating Least Squares algorithm, data matrix will be smaller and easier to use.

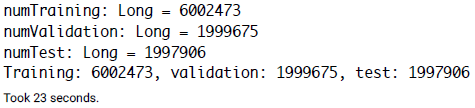
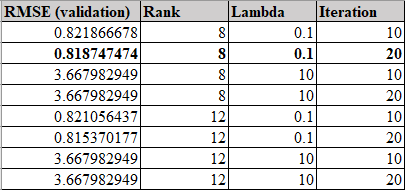


Figure 10: Data Summary

In the Fig. 10, the results show that about 60% of data are trained, 20% are used for validation, and 20% are for testing.

Table 1: RMSE Results



The Table 1 gives 8 different runs of RMSE, and the second one is selected to be the best model. However, this step takes 459 seconds to complete. And the whole program takes 573 seconds to finish, which means the several runs of RMSE function is the most important factor to slow the program down.

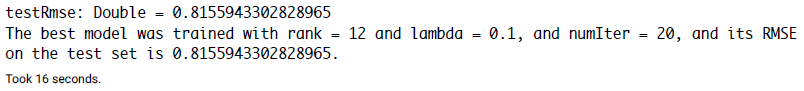


Figure 11: Best Model

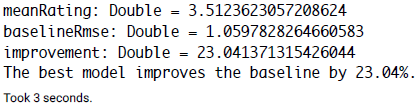


Figure 12: Model Improvement

As Fig. 12 shows, the selected model is 23.04% better than other results.

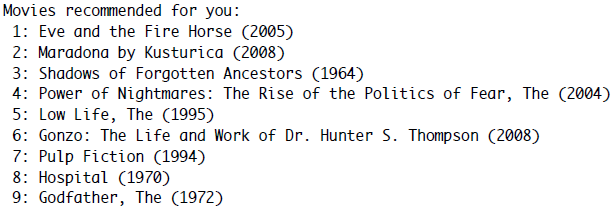
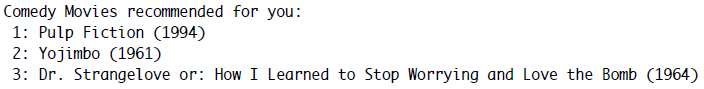




Figure 13:User Recommendations

The Fig. 13 shows an example of 10 recommendations for a specific user, according to the user preference.



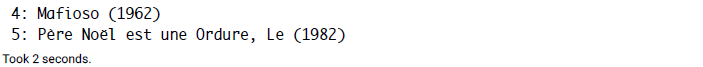


Figure 14:Movie Recommendations

The Fig. 14 gives an example of 5 recommendations for a specific type of movies.

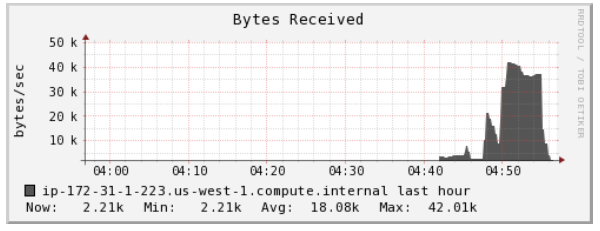


Figure 15: Request Received

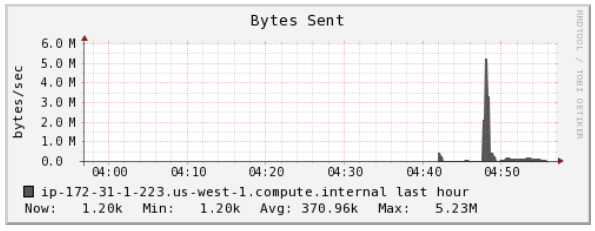


Figure 16: Throughput

The Fig. 15 shows the average requests received are about 18k bytes/s, and the Fig. 16 gives the throughput of around 371k bytes/s in average.

Analysis

The recommendations system is designed for personalized movie information recommendation with the data driven model. Before building the machine learning model, the movie dataset is separated into three parts, one for training (60%), one for validation (20%), and one for testing (20%). In the prediction process, the prediction engine could get the top 10 movies for preferred type and recommend for various users based on three Spark SQL data columns of movie genre, personality, and rating score. In this case the predictions of third element which is the rating for that movie and user, is the predicted by our Alternating Least Squares Algorithm model.[8] After the recommendation finish. In order to measure the accuracy of recommendation result, the function of Root Mean Squared Error (RMSE) is defined to evaluate the performance of ALS algorithm provided by spark MLlib library with 3 following parameters: rank is the number of latent factors in the model, iterations is the number of iterations to run and lambda specifies the regularization parameter in ALS. The model selects different values for these parameters and measure the RMSE for each combination.[3] It is shown in the result that the best combination is generated to be the larger rank of 12, Iteration of 20 and the smaller lambda of 0.1. In addition to the accuracy, we can compare the model proposed in this report to a more naïve item-based Collaborative Filtering prediction based on the average rating. It is shown in the result that mean rating is 3.0507 and the recommendation baseline is 1.0591, the improvement of baseline to the test case using ALS algorithm increase to 23.0381 and the best model improve the baseline by 23.04%. After the evaluation of model quality and performance. With a collaborative Zeppelin notebook, developers can quickly visualize your data, explore the algorithm, and share the results with others in the organization. [3]

Conclusion

An ever-increasing quantity of data has made the search for desired information still more difficult for users. Personalized services have attracted attention as a means to cope with this problem. A personalized service can be referred to as a recommender system in which a target user is provided with her/his preferred services or products. Recommender systems that are frequently used in e-commerce include collaborative filtering systems.[9] However, they have faced such issues as sparsity, scalability, transparency, and collaborative filtering is a costly procedure since requires updating its model when new user preferences arrive. To provide a solution to these problems, this report presents a recommender system with following advantages.[9] Firstly, the Alternating Least Square Algorithm is used in Spark’s MLlib library on Amazon Web Service to develop a recommendation engine with intelligent machine learning methods. Second, the introduced distributed file system HDFS and distributed batch framework MapReduce on Hadoop platform with Spark assistance can not only store the growing mass of data, but also process the data in parallel, which improves the performance of the algorithm and the response speed of the system. Third, the web server is designed to persistent the prediction model as on-line recommendations system to incorporate new user rating data and update model timely. Finally, this system uses as personalization factors only the information that a user puts in when logging into the system, with which it effectively provides a solution to such issues as sparsity and scalability and is thus able to supply an optimal list of recommended movies to users in real world.[9] In our project, collaborative filtering algorithm is implemented by using Zeppelin Notebook to predict users’ movie pereferences. The MovieLens datasets that has 10 million of movies’ ratings choosen, trained, validated, and tested. The RMSE algorithm is used to calculate the accuracy of collaborative filtering algorithm; as a result, the algorithm implemented has a good performance for prediction.

References

1. “Recommender system,” Wikipedia, 19-Nov-2017. [Online]. Available: https://en.wikipedia.org/wiki/Recommender\_system. [Accessed: 20-Nov-2017].
2. Collaborative Filtering for Recommender Systems - IEEE Conference Publication. [Online]. Available: http://ieeexplore.ieee.org/document/7176109/. [Accessed: 20-Nov-2017].
3. G. Ernest, “Building a Recommendation Engine with Spark ML on Amazon EMR using Zeppelin,” Amazon Web Services, 14-Nov-2015. [Online]. Available: https://aws.amazon.com/blogs/big-data/building-a-recommendation-engine-with-spark-ml-on-amazon-emr-using-zeppelin/. [Accessed: 20-Nov-2017].
4. B. Akyildiz, “Alternating Least Squares Method for Collaborative Filtering,” Alternating Least Squares Method for Collaborative Filtering | Bugra Akyildiz, 19-Apr-2014. [Online]. Available: https://bugra.github.io/work/notes/2014-04-19/alternating-least-squares-method-for-collaborative-filtering/. [Accessed: 20-Nov-2017].
5. J. Steinweg-Woods, “A Gentle Introduction to Recommender Systems with Implicit Feedback,” A Gentle Introduction to Recommender Systems with Implicit Feedback – Jesse Steinweg-Woods, Ph.D. – Data Scientist, 30-May-2016. [Online]. Available: https://jessesw.com/Rec-System/. [Accessed: 20-Nov-2017].
6. Y. Zhuang, B. Xu, H. Wu, S. Han, and H. Jin, “Movie Recommender System,” GitHub, 06-Dec-2015. [Online]. Available: https://github.com/StephenWuHao1212/COMPSCI-571-Machine-Learning/blob/master/Project%20Report(Movie%20Recommender%20System).pdf. [Accessed: 20-Nov-2017].
7. “The MovieLens Datasets: History and Context,” ACM Transactions on Interactive Intelligent Systems (TiiS). [Online]. Available: https://dl.acm.org/citation.cfm?doid=2866565.2827872. [Accessed: 20-Nov-2017].
8. J. A. Dianes, “Building a Movie Recommendation Service with Apache Spark & Flask - Part 1,” Codementor. [Online]. Available: https://www.codementor.io/jadianes/building-a-recommender-with-apache-spark-python-example-app-part1-du1083qbw. [Accessed: 20-Nov-2017].
9. D.-S. Park, “Improved Movie Recommendation System based-on Personal Propensity and Collaborative Filtering,” KIPS Transactions on Computer and Communication Systems. [Online]. Available: http://www.koreascience.or.kr/article/ArticleFullRecord.jsp?cn=JBCRIN\_2013\_v2n11\_475. [Accessed: 20-Nov-2017].
10. H. Li, angzheng He, M. Lublin, and Y. Perez, “atrix Completion via Alternating Least Square(ALS),” DAO: Distributed Algorithms and Optimization, 13-May-2015. [Online]. Available: https://web.stanford.edu/~rezab/classes/cme323/S15/notes/lec14.pdf. [Accessed: 20-Nov-2017].