Movie Recommendation System using Apache Spark

Ankit Maheshwari*, Anuradha Kumari, Anjali Kumari, Neeraj Kumar and Nandini B.M

Dept. of Information Science and Engineering, The National Institute of Engineering, Mysuru.

DOI: https://doi.org/10.21467/proceedings.1.45

* Corresponding author email: mankit752@gmail.com

Abstract

Recommendations have become a very important part of everyone's life as today we have a huge number of options for a particular product and choosing the one which meets our need is monotonous. Also, movies have become a very popular form of entertainment in the modern world. However manually searching for movies of our interest from such a mammoth set of present movies can be a tedious task. Movie recommendation System plays an important role in this scenario therefore the proposed system helps the user in selecting movies of their choice by collecting the user data and then analysing their preferences and collecting various related data to use in our algorithm to then recommend the best possible movies to the user. Weights are assigned to the various preferences in order to get the best possible results. To implement the proposed system the Apache Spark framework is used. The language used to implement the system is Scala which is the native language in which Spark has been implemented. We will also use the capabilities of MLLib which is the Machine Learning Library of Spark to implement the system. Many algorithms from MLLib will be used. The expected model would be able to propose movies to the user from a predefined dataset according to his preferences in a very short span of time i.e. - the model will give really good performance as compared to already existing systems.

Index Terms— Apache Spark, MLLib, Recommender system, Scala

1 NTRODUCTION

In this modern era, with the advancement in technology internet has become an important part of everyone's daily life. Due to increase in technology and communication between people and devices the amount of data availability is growing constantly. Modern consumers are flooded with options in every field right from looking for a hotel, study materials or buying something online. So, companies have deployed recommendation system to help their users which has also added to the economy of some e-commerce.

Recommendation system contains simple algorithms so that it can recommend user with relevant items by collecting user data and discovering data patterns in the data set and filtering user related data. Many research have been going on this area. Recommendation system has also been applied to many fields and one of those fields is movie recommendation system so that it can improve user experience. There are varieties in movies like educational, thriller, entertainment movies. It can also be classified by their genre like comedy, horror, thriller etc. Due to this large no of options, choosing the favorable one is a troublesome job.

There are two approaches for recommendation (1). Content based approach in this we create a user profile for each user and use movie features which is pre-determine to generate user preferences. (2). Collaborative approach which produces recommendation based on the knowledge of the user attitude to items.

The proposed model uses item based collaborative model using the Apache Spark framework [1]. Apache Spark is a recent technology for the processing on Big Data which keeps the data it is processing, in the RAM until it needs to be evacuated and provides much better performance as compared to the traditional frameworks. Spark which has Spark streaming capabilities which enables it to process incoming data on the go. It also provides real time streaming as compared to the old concepts which provided batch systems.

Spark uses RDD (Resilient Distributed Datasets) and data frames. A Dataset is a distributed collection of data. Dataset is a new interface added in Spark 1.6 that provides the benefits of RDDs (strong typing, ability to use powerful lambda functions) with the benefits of Spark SQL's optimized execution engine. A Dataset can be constructed from JVM objects and then manipulated using functional transformations (map, flatMap, filter, etc.). The main abstraction Spark provides is a Resilient Distributed Dataset (RDD), which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. RDDs are created by starting with a file in the Hadoop file system (or any other Hadoop-supported file system), or an existing Scala collection in the driver program, and transforming it. Users may also ask Spark to persist an RDD in memory, allowing it to be reused efficiently across parallel operations. Finally, RDDs automatically recover from node failures. Spark machine learning libraries (Spark MLLib) which helps us in recommendation of more accurate movies is also used. MLLib's goal is to make practical machine learning scalable and easy.

2 LITERATURE SURVEY

Many algorithms have been developed in recent times to generate movie recommendations all differing by small factors. Bobin K' Sunny *et al.*[1] used Apache Spark to developed a TV channel recommendation system using Apache Spark's streaming capabilities to increase the velocity of data transmission to cope up with real time recommendations. Md. Tayeb Himel *et al.*[2] developed a system by monitoring the users actions and then assigning weights to the movies depending on those actions. The recommender system internally uses the k-means algorithm to form the required clusters and then to provide the necessary recommendations. Shreya Jain *et al.* [3] devised a hybrid approach using collaborative filtering and content-based filtering using Support Vector Machine as a classifier and genetic algorithms and comparative results were an improvement in the quality, accuracy and capability of the movie recommender as compared to traditional approaches. We try to implement a model which utilizes the Cosine similarities metric to group similar movies, depending on the movie pairs and user ID pairs. We will use the inbuilt Spark Resilient Distributed Dataset(RDD) data structures to make our working easier. The data used has been downloaded from the Grouplens website which is run by the University of Minnesota. The Apache Spark platform clubbed with the use of Scala language provides a tremendous speed boost.

3 PROPOSED SYSTEM DESIGN

There are no specific criteria for a recommendation to be correct or incorrect. The accuracy of the results depends on its acceptance by the user. 2 out of 3 movies watched on Netflix are recommended. Similarly, other sites also have their own criteria for generating recommendations. The recommendation model needs to provide as accurate predictions possible in the least possible time.

The system we build runs on the Apache Spark framework. We can either perform distributed computing on Spark or even computation on a local system using all the available cores. We would be using the latter to implement our small scale model but it can easily be upgraded for a larger model using distributed systems due to Spark's flexibility and ease of use. Spark is easy to install and use and also can deal with a large variety of data such as structured, unstructured, etc. Spark also contains in built machine learning libraries which contain several ready to use machine learning algorithms. Either these libraries(MLLib) can be used or an algorithm can be formulated. The proposed model uses the latter approach for better understanding of the process adopted.

The data set consists of various files such as the data file which consists of the userID, movieID and the ratings the user gives. A genre file consists of the different genres along with integer codes. There are a total of 19 genres. An item file consists of the mappings between the movie IDs and user names and which also contains a '1' for all the genres to which it belongs to, else a '0' is present in that field.. This file also contains the date of release for the various movies. A movie may belong to more than 1 genre. The training and test data can also be split in different ways and tested for accuracy. In our case we would be using an 80% training – 20% test data split. This can be changed to different combinations in different approaches.

The basic working of the model is explained below:

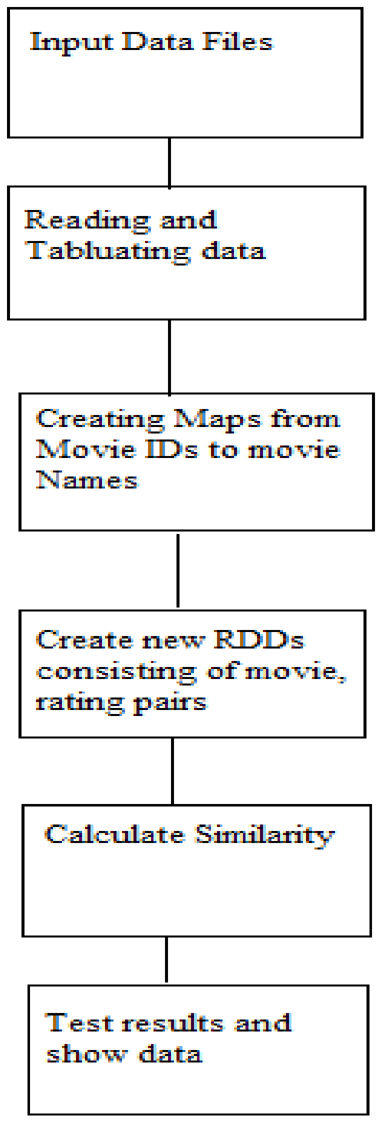
Input Data Files
Reading and Tabluating data
Creating Maps from Movie IDs to movie Names
Create new RDDs consisting of movie, rating pairs
Calculate Similarity
Test results and show data

Fig. 1 : Basic architecture of the model

The spark context is created using every available core of the local machine. Spark has inbuilt commands to start this context. The data i.e. the actual ratings file is then loaded up into the system where it will be cached.

A map of the movie IDs to movie names is created as a key value pair in the form of: userID => (movieID, ratings).

A map of Integers to Strings i.e. from movieIDs to movie names is created from the items file. For this step we emit every movie rated together by the same user. We use a self-join to find every combination to achieve this aim. This point our RDD consists of userID => ((movieID,rating), (movieID, rating)). Every pair of movies that we can evaluate that are similar to each other have to be found out. Then the duplicate pairs are removed to make the later processing easier by having less data to deal with. This can be achieved by eliminating all pairs in which movieID1 < movieID2.

Everything is later keyed by (movieID, movieID) pairs. This will help us in later computations. Now we have (movieID1, movieID2) => (rating1, rating2) pairs. Then we collect all ratings for each pair and compute similarity. For this we collect all the ratings from users who have watched both movies in the pair. At this point we have a list of all the movies and another list of all the respective ratings.

The Proposed Approach for the model is shown in the below:

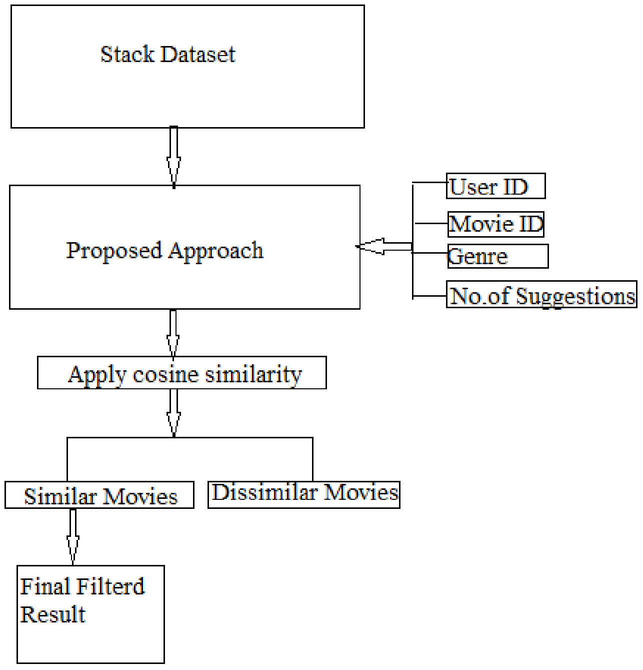

Fig :2 Proposed Approach

The next step involves computing the similarities. For this we will use the Cosine Similarity metric which is one of the many similarity metrics available in Machine Learning. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. For computing the Cosine Similarity, we can assume Vectors in a higher dimensional space. The cosine of these vectors measures how close these vectors are to each other or how closely they are related. In our case the vectors can be taken as the movies and their respective user ratings. Spark has an inbuilt function for calculating this. However, the mathematical formula can be shown as:

$$\text{Similarity}=\cos(\Theta)=\frac{\mathbf{A.B}}{||\mathbf{A}||\,||\mathbf{B}||}=\frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

This value will be stored in the cache since it will be used more than once.

After extract the similarities we get the movies similar to the movies that we pass as the argument. We find all the pairs that contain the given movie. After this step we refine the results by eliminating the results that have below a minimum threshold rating. This ensures the minimum quality of the recommendation list. Then finally we display the top 10 movies based on the highest rating that could be recommended for that particular user.

4 FUTURE ENHANCEMENT

Several methods can be used to enhance the quality of the aforementioned recommended system. Some of the improvements may include:

- Attain the perfect clean dataset containing details of almost all movies to attain accurate results.
- Run the platform on a distributed cluster. This will greatly improve the performance, however it comes with the trade-off of having to pay extra charges for the usage of online clusters. As we improve the number of nodes in the clusters the cost will also increase.
- Discard bad ratings – Only recommend good movies.
- Try different similarity metrics. These can include Pearson Correlation Coefficient, Jaccard Coefficient, Conditional Probability etc. An optimum metric is tough to predict. Different metrics may give different results depending on the datasets used.
- Adjust the thresholds for minimum co-raters or minimum scores.
- Invent a new similarity metric that takes the number of users into account.
- Adjust the training and test ratios t5o find the optimum results.
- Spark's Machine Learning Libraries (MLLib) can also be used to reduce the size of the code and make it easier.
- To make the system diversifiable so that it can satisfy users of different geographical locations.

5 CONCLUSION

We have modeled a system which provides outputs i.e. movie recommendations at a very good speed because of the Apache Spark framework used. We will implement this on a small scale but is highly scalable and can be performed on distributed clusters as well. Also we will implement the system on MovieLens dataset but can also be applied on other datasets. The proposed algorithm is self designed but we can also easily design the system using Spark's MLLib to implement the system. Also various variations to the metrics and algorithms can be used to enhance performance since there is no single fixed algorithm that provides best results. We can conclude by saying that our proposed system is pretty efficient when compared to existing systems but can be enhanced even further.

6 REFERENCES

[1] Bobin K Sunny; P S Janardhanan; Anu Bonia Francis;Reena Murali "Implementation ofa selfadaptive real time recommendation system using spark machine learning libraries" 2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)Year: 2017 ,Pages: 1 – 7.

[2] Md. Tayeb Himel; Mohammed Nazim Uddin; Mohammad Arif Hossain; Yeong Min Jang "Weight based movie recommendation system using K-means algorithm"2017 International Conference on Information and Communication Technology Convergence (ICTC)

[3] Year: 2017,Pages: 1302 – 1306.

[4] Shreya Agrawal; Pooja Jain "An improved approach for movie recommendation system" 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC),Year: 2017, Pages: 336 – 342 ,IEEE Conferences.

[5] Mohammed Nazim uddin; Jenu Shrestha; Geun-Sik Jo "Enhanced Content-Based Filtering Using Diverse Collaborative Predictionfor Movie Recommendation " 2009 First Asian Conference on Intelligent Information and Database Systems ,Year: 2009 ,Pages: 132 – 137.

[6] Anshul Gupta; Hirdesh Shivhare; Shalki Sharma "Recommender system using fuzzy c-means clustering and genetic algorithm based weighted similarity measure " 2015 International Conference on Computer, Communication and Control (IC4) ,Year: 2015, Pages: 1 – 8.

[7] Truong Khanh Quan; Ishikawa Fuyuki; Honiden Shinichi "Improving Accuracy of Recommender System by Clustering Items Based on Stability of User Similarity " 2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06) ,Year: 2006 , Pages: 61 – 61.

[8] Sajal Halder; A. M. Jehad Sarkar; Young-Koo Lee "Movie Recommendation System Based on Movie Swarm " 2012 Second International Conference on Cloud and Green Computing ,Year: 2012 ,Pages: 804 – 809.