SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Movie Recommendation System Review Analysis

Submitted for

DIGITAL MARKETING

Submitted by:

(E23MCAG0013) Isha Kumari

(E23MCAG0109) Kanupriya

Submitted to-

Dr. Arun Chaudhary

Jan-May 2024

INDEX

Abstract

Movie recommendation system participate in a significant role in entertainment industry by recommending users new movies based on their previous history. It is an indispensable tool for navigating our users for the comprehensive range of movie options available. This system utilizes or practice various technologies, such as content-based filtering, collaborative filtering and hybrid approaches to produce personalized recommendations. Content based filtering main focal point is on the attribute of the movies and user likings to meet the user's taste. Collaborative filtering analyses users' reactions to pick out similar article. Hybrid approach merge these formulas to enhance the accuracy and content of recommendations. The review inspects the role of deep learning for boosting the recommendation system performance. In essence, our project recognizes key challenges, like data deficiency, cold start problems and also suggests future research supervision to label these questioning and revamp the mostly effectiveness of the recommendation system.

1. INTRODUCTION AND RELATED WORK

If we talk about the Movie recommendation System, it uses a machine learning based approach which refine or forecast users' films liking based on their previous choice and search history pattern base behaviour. It is a highly developed prediction system which demonstrate few feasible movie choices and desires for users in relation to domain specific articles, such as movies. The main objective of the recommendation system is to refine and speculate only those movies that a consistent user is quite possibly wanted to watch. In this system the Locality-sensitive Hashing (LSH) algorithm can be use. Basically, LSH algorithm is a technique for executing probabilistic dimension depletion of high dimension data and also can be used for nearest neighbour search in high dimensional spaces which is very functional for recommendation systems. Moreover, LHS can be used to briskly find homogeneous movies based on their characteristics. This algorithm tasks by generating a "fingerprint" for respective movie, which is a condense representation of the movie features analysis. These fingerprints can be used to swiftly find the homogeneous movies by differentiating the fingerprints using a hash function.

The related work in the area of movie recommendation system using the LSH algorithm incorporate different distance metrics for this algorithm like Cosine distance or Jaccard distance. Also, when using the non-identical data structures for the LSH algorithm, for example Hyper Loglog and Bloom Filters. So that these dissimilarities can revamp or enhance the unambiguousness and productivity of the recommendation system. Movie recommendation system incorporate hybrid filtering methods that collaborate both content-based and communal filtering approaches and furthermore with deep learning-based technique that utilize neural networks to become competent in complex presentation of users and movies. These methods can make preparations more precise and give a personal touch to movie recommendations for users.

In this the following sections this paper contains: -

- Collection of Data
- Cleaning the Data
- Evaluation of the Performance
- Training the Data

- Testing the Data
- Model Building

2. LIBRARIES USED

   2.1 Python

   2.2 PySpark

3. METHODOLOGY

For Building a movie recommendation engine using PySpark involves several steps. PySpark is a Python API for Apache Spark, which is a fast and general-purpose cluster computing system for Big Data processing. Below is the high-level methodology you can follow: -

Data Preparation: Data preparation is the foundational step in any analysis or recommendation system. In our scenario, we are working with two main datasets: **ratings** and **movies**. These datasets serve distinct purposes, with **ratings** containing information about user interactions with movies and **movies** providing details about the movies themselves.

The **ratings** dataset typically consists of several columns, including **userId**, **movieId**, and **rating**. Each row in this dataset represents a user's rating for a specific movie. The **userId** column contains unique identifiers for users, while the **movieId** column contains identifiers for movies. The **rating** column contains the ratings given by users for the corresponding movies, typically on a scale from 1 to 5.

Conversely, the **movies** dataset typically contains information such as **movieId**, **title**, and **genres**. The **movieId** column serves as a unique identifier for each movie, while the **title** column contains the titles of the movies. The **genres** column provides information about the genres to which each movie belongs.

Joining Data: To leverage the information from both datasets effectively, we perform a data integration step by joining the **ratings** dataset with the **movie's** dataset. This is typically achieved through an inner join operation based on the common column **movieId**. By combining these datasets, we enrich the **ratings** data with additional details about the corresponding movies, such as their titles and genres. This integration facilitates a more comprehensive analysis and enables us to make personalized recommendations based on user preferences.

Filtering Ratings by User: Once the datasets are joined, we filter the resulting Data Frame to focus on ratings submitted by a specific user. For example, we may choose to analyse the ratings provided by user 100 exclusively. This filtering step allows us to isolate the ratings given by a particular user, providing a targeted view of their preferences and interactions with movies. By narrowing our focus to a specific user, we can tailor our recommendations to better suit their tastes and preferences.

Sorting Ratings: Following the filtering step, we sort the filtered Data Frame in descending order based on the ratings assigned by the user. This sorting arrangement ranks the movies according to the user's preferences, with the highest-rated movies appearing first. Sorting the ratings in this manner allows us to identify the movies that resonate most strongly with the user and are likely to be well-received recommendations. By prioritizing movies with higher ratings, we ensure that the recommendations are aligned with the user's preferences and interests.

Limiting Results: To maintain clarity and relevance, we limit the output to display only the top-rated movies for the selected user. Typically, we set a threshold to show the top 10 rated movies, although this number can be adjusted based on preferences or specific requirements. Limiting the results ensures that the presentation remains concise and focused, highlighting the movies that received the highest ratings from the user. This approach allows us to provide personalized recommendations that are tailored to the user's preferences while avoiding information overload.

Displaying Results: Finally, we present the resulting Data Frame, showcasing the top-rated movies for the user along with pertinent details such as movie titles and genres. This presentation format facilitates easy interpretation and provides actionable insights for making recommendations. By displaying the top-rated movies in a clear and structured manner, we enable users to explore and engage with content that aligns with their interests and preferences.

In summary, the methodology outlined above provides a structured approach for analysing user ratings and generating personalized movie recommendations based on individual preferences. By leveraging data integration, filtering, sorting, and limiting techniques, we can deliver targeted recommendations that enhance user satisfaction and engagement with the platform. This methodology serves as a foundation for building recommendation systems that deliver relevant and compelling content to users, ultimately driving user retention and loyalty. Below are the table which give the information about the data and results.

userId	movieId	rating	timestamp
1	1	4.0	964982703
1	3	4.0	964981247
1	6	4.0	964982224
1	47	5.0	964983815
1	50	5.0	964982931
1	70	3.0	964982400
1	101	5.0	964980868
1	110	4.0	964982176
1	151	5.0	964984041
1	157	5.0	964984100
1	163	5.0	964983650
1	216	5.0	964981208
1	223	3.0	964980985
1	231	5.0	964981179
1	235	4.0	964980908
1	260	5.0	964981680
1	296	3.0	964982967
1	316	3.0	964982310
1	333	5.0	964981179
1	349	4.0	964982563

Table 1. Movie rating Dataset by User

userId	count
414	2698
599	2478
474	2108
448	1864
274	1346
610	1302
68	1260
380	1218
606	1115
288	1055
249	1046
387	1027
182	977
307	975
603	943
298	939
177	904
318	879
232	862
480	836

Table 2. Total number of ratings given by individual user

movieId	count
356	329
318	317
296	307
593	279
2571	278
260	251
480	238
110	237
589	224
527	220
2959	218
1	215
1196	211
50	204
2858	204
47	203
780	202
150	201
1198	200
4993	198

Table 3. Total number of ratings received on every movie

userId	recommendations
1	[{3379, 5.7526016...
2	[{131724, 4.79097...
3	[{70946, 4.915376...
4	[{3851, 4.8457103...
5	[{3379, 4.607003}...
6	[{3925, 4.8263626...
7	[{33649, 4.455137...
8	[{3379, 4.695211}...
9	[{5490, 4.8849015...
10	[{71579, 4.529316...

Table 4. Generated recommended movie list for all users

userId	movieId	rating
1	3379	5.7526016
1	33649	5.581407
1	5490	5.531987
1	5416	5.400252
1	5328	5.400252
1	3951	5.400252
1	171495	5.3989573
1	5915	5.3693156
1	78836	5.3661513
1	6460	5.336748

Table 5. Elaborated movie recommendation set for the user

EXPERIMENTAL RESULTS

It sets up the ALS model with different hyperparameters (rank and regularization parameter) using cross-validation then splits the data into training and testing sets after that fits the cross-validator to the training dataset. It extracts the best model based on the cross-validation results. It evaluates the best model on the test dataset using RMSE (Root Mean Squared Error). It generates recommendations for all users. This output includes the best model parameters (rank, maxIter, and regParam), the RMSE value, and recommendations for each user. You can interpret the results as follows:

- **Best Model Parameters**: These are the hyperparameters that resulted in the lowest RMSE during cross-validation.
- **RMSE**: This is the evaluation metric indicating how well the model performs on the test dataset. Lower RMSE values indicate better model performance. The rmse which we are getting is `0.8692406004565547`
- **Recommendations**: These are the top recommendations for each user based on the trained model.

movieId	userId	rating	title	genres
1101	100	5.0	Top Gun (1986)	Action|Romance|
1958	100	5.0	Terms of Endearme...	Comedy|Drama|
2423	100	5.0	Christmas Vacatio...	Comedy|
4041	100	5.0	Officer and a Gen...	Drama|Romance|
5620	100	5.0	Sweet Home Alabam...	Comedy|Romance|
368	100	4.5	Maverick (1994)	Adventure|Comedy|...|
934	100	4.5	Father of the Bri...	Comedy|
539	100	4.5	Sleepless in Seat...	Comedy|Drama|Romance|
16	100	4.5	Casino (1995)	Crime|Drama|
553	100	4.5	Tombstone (1993)	Action|Drama|Western|

Table 6. Final outcome of movie recommendation based on user's interest

CONCLUSION

In conclusion, we have accomplished a movie recommendation system using pyspark, advance edge filtering approach to provide filtering customize movie recommendations to users. Through our evaluation, we displayed the effectiveness of our approach in providing precise and appropriate recommendation by estimating the performance of recommendation system using metrics such as RMSE. By manipulating PySpark's portability and productivity we are able to come up with real time recommendations and operate large datasets which exhibit the system efficiency and feasibility in a manufacturing environment. In comprehensive, our recommendation system

has conveyed great potential in assuming customize and appealing movie recommendations using pyspark by reaping benefits to both the users and the business.

REFERENCES

[1]https://notebook.community/jeroarenas/MLBigData/5_RecommenderSystems/Recommender%20systems%20-%20Part%202-Students.

[2] https://medium.com/analytics-vidhya/crafting-recommendation-engine-in-pyspark-a7ca242ad40a.

[3]https://www.analyticsvidhya.com/blog/2022/08/build-your-recommendation-system-using-mlib/ .

[4]https://datascience.fm/movie-recommender-system-with-pyspark/.

[5]https://www.datacamp.com/courses/recommendation-engines-in-pyspark.

[6]https://stackoverflow.com/questions/68767426/how-to-get-pyspark-als-for-recommendation-system-to-return-the-test-set-data.