**CSP-554**

**Movie Recommender System - Project Final Phase (Report)**

Group Members: Sriram Rani(A20437890), Darshan Halyal(A20448502), Sneja Shah (A20435827), Akhil Deshmukh (A20428150)

* **INTRODUCTION**

The modern age of knowledge has contributed to the exponential growth of data processing. To build more effective processes, data is used, and this is where recommendation systems come into play. Recommendation systems are a type of knowledge filtering system because they enhance the quality of search results and include items that are more important to the search item or linked to the user's search history. They are used to predict the rating or preference is given to an object by a consumer. They have been applied in some way or the other by almost every major tech company: Amazon uses it to recommend items to consumers, YouTube uses it to determine which video to play next on auto play, and Facebook uses it to suggest pages to like and follow. Also, businesses such as Netflix and Spotify rely heavily on the efficacy of their market recommendation engines and the performance of big-data technology such as Apache Spark, a highly developed data processing engine for data processing on large scale over thousands of compute engines in parallel. This enables the processor capability to be maximized over these computing engines. Spark can perform several tasks for data processing. Thus, to reach our use case, we use spark parallel processing.

The project work is about building a Movies Data recommendation framework with the aid of Bigdata Technology through data transformation and visualization. We will use the Kaggle Movies Dataset Data Set of about 5000 tmdb movies with story, cast, crew, budget, and revenue data for several thousand movies. Name, Gender, Definition, Director, Actors, Year, Runtime, Ranking, Votes, Revenue, Metascore are the data points included. Using a number of big data tools to explore and gain knowledge from the IMDB data collection by ingesting, transforming, profiling, summarizing, and visualizing the data and developing a suggested framework. Models will be developed using important evaluated outcomes that can forecast the movie characteristics for future purposes with regard to the dependent variable.

* **LITERATURE REVIEW**

*Wang, Lidong & Alexander, Cheryl. (2016). Machine Learning in Big Data. International Journal of Mathematical, Engineering and Management Sciences*

This paper talks about the different machine learning algorithms used is big data analytics. It gives an overview of what each algorithm does and can be used for. It compares all the algorithms together and gives us examples of application in the big data domain. Also provides us with an insight to the challenges of using machine learning algorithms in a big data environment. All this will help with decision making in how to proceed and what to use for the project. Since our data set is structured supervised Learning methods will provide more accurate results and training will be simpler. Methods such as Naïve Bayes, Random Forest, support vector machines (SVM), Decision Trees etc.

*Shevate, P. (2016). Sentiment Analysis for Movie Reviews using NLP Prachi Shevate (Team 5) California State University, Sacramento. Retrieved 2016*

This paper talks about analysis of the sentiments on Movie datasets from platforms like IMDB. It talks about the various tools or libraries available for cleaning and preprocessing text data for sentiment analysis. It briefly describes the different types of models used in sentiment analysis such as bag of words model, Natural Language Tool Kit (NLTK), Word2Vec etc. It discusses the pros and cons of these tools and models. Later, it talks about IMDB movie reviews and how we can capture word sentiment using representation learning and a few more details about the data set itself. This paper helps us analyze datasets which have more text than numerical data. Also shows different tools, libraries and models that could be used to perform analysis on IMDB datasets. Since our data set also contains lot of text, we could use the NLTK, word embeddings, and few other concepts from this paper.

*Choudhry1, N., Xie2, J., & Xia3, X. (2020, June 01). IOPscience. Retrieved November 12, 2020*

This paper’s main focus is to create a Movie Predictive System which will recommend movie supported other customer’s ratings for dissimilar movies, the preferences of customer, by using Apache Spark we'll analyze and assume the superlative executed movies supported customer opinion and reviews. It talks about using matrix factorization technique to coach model to find out user-item interaction by capturing user information in user latent factors and item information in item latent factors. Also focuses on collaborative filtering and use ALS (Alternating Least Squares) algorithm to form movie predictions. this provides us a practical example on how ml algorithms are used on movie data sets and alternative ways we could use the features to coach the model and obtain the specified outputs from the model.

*Zhenlin Kan, Xinru Cheng, Seung Hyun Kim, Yuting Jin. “Apache Hive-Based Big Data Analysis of HealthCare Data”. International Journal of Pure and Applied Mathematics Volume 119 No. 18 2018, 237-259.*

This paper analyzes healthcare data using Apache Hive. The study uses several Hive queries for data analysis. For improved data analytics, Apache Hive is integrated with Tableau as a strong analytics tool. This helps to look at the conversion of knowledge into useful information. the utilization of Tableau helps analyze high-alert states with maximum deaths, major diseases leading to more deaths, and deaths like deceasing diseases. Exponential International Journal of Pure and applied math Special Issue 242 smoothing is employed to forecast disease trends. This paper presents the proposed work and experiments for the analysis of massive data within the EHR. This study uses exponential smoothing for forecasting purposes, and as an extension to the present study, machine learning techniques like neural networks may produce better prediction results.

*C. Chiru, C. Preda, V. Dinu and M. Macri, "Movie Recommender system using the user's psychological profile,"*2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, 2015, pp. 93-99, doi: 10.1109/ICCP.2015.7312611.*

In this paper we present Movie Recommender, a system which provides movie recommendations supported the knowledge known about the users. These recommendations are done using the analysis of the users' psychological profile, their watching history and thus the films scores from other websites. they're supported aggregate similarity calculation. The paper shows the usage of both filtering methods of Collaborative and Content-based. Although there are similar applications available, they have a bent to ignore the info specific to the user, which in our opinion is significant for his/her behavior.

*C. M. Wu, D. Garg and U. Bhandary, "Movie Recommendation System Using Collaborative Filtering," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 11-15, doi: 10.1109/ICSESS.2018.8663822.*

As the business needs are accelerating, there's an increased dependence on extracting meaningful information from humongous amount of data to drive business solutions. an equivalent is true for digital recommendation systems which are getting a norm for consumer industries like books, music, clothing, movies, news articles, places, utilities, etc. These systems collect information from the users to enhance the longer-term suggestions. This paper aims to explain the implementation of a movie recommender system via two collaborative filtering algorithms using Apache Mahout. Furthermore, this paper also will specialize in analyzing the info to realize insights into the movie dataset using Matplotlib libraries in Python.

* **DATA DESCRIPTION**

*We used two different datasets just to cover more movies data and separate implementation of different filtering techniques on each dataset*

(TMDb), T. (2017, September 28). TMDB 5000 Movie Dataset. Retrieved December 09, 2020, from <https://www.kaggle.com/tmdb/tmdb-movie-metadata>

Here I will be using the above-mentioned dataset which has been downloaded from the repository. Here's a data set of 1,000 most popular movies on TMDB in the last 10 years. The data points included are: Title, Genre, Description, Director, Actors, Year, Runtime, Rating, Votes, Revenue, Metascrore

Here the Number of columns are: **12**

Number of Instances are: **108544**

 The first dataset contains the following features:-

* movie\_id - A unique identifier for each movie.
* cast - The name of lead and supporting actors.
* crew - The name of Director, Editor, Composer, Writer etc.

The second dataset has the following features:-

* budget - The budget in which the movie was made.
* genre - The genre of the movie, Action, Comedy ,Thriller etc.
* homepage - A link to the homepage of the movie.
* id - This is infact the movie\_id as in the first dataset.
* keywords - The keywords or tags related to the movie.
* original\_language - The language in which the movie was made.
* original\_title - The title of the movie before translation or adaptation.
* overview - A brief description of the movie.
* popularity - A numeric quantity specifying the movie popularity.
* production\_companies - The production house of the movie.
* production\_countries - The country in which it was produced.
* release\_date - The date on which it was released.
* revenue - The worldwide revenue generated by the movie.
* runtime - The running time of the movie in minutes.
* status - "Released" or "Rumored".
* tagline - Movie's tagline.
* title - Title of the movie.
* vote\_average - average ratings the movie recieved.
* vote\_count - the count of votes recieved.

MovieLens. (2020, May 21). Retrieved December 09, 2020, from <https://grouplens.org/datasets/movielens/>

MovieLens 25M movie ratings. Stable benchmark dataset. 25 million ratings and one million tag applications applied to 62,000 movies by 162,000 users. Includes tag genome data with 15 million relevance scores across 1,129 tags. Released 12/2019

In our case, we will use the latest datasets:

* Small: 100,000 ratings and 2,488 tag applications applied to 8,570 movies by 706 users. Last updated 4/2015.
* Full: 21,000,000 ratings and 470,000 tag applications applied to 27,000 movies by 230,000 users. Last updated 4/2015.
* **DATA LOADING, EXTRACTION AND PREPROCESSING**

In order to build an on-line movie recommender using Spark, we need to have our model data as preprocessed as possible. Parsing the dataset and building the model everytime a new recommendation needs to be done is not the best of the strategies.

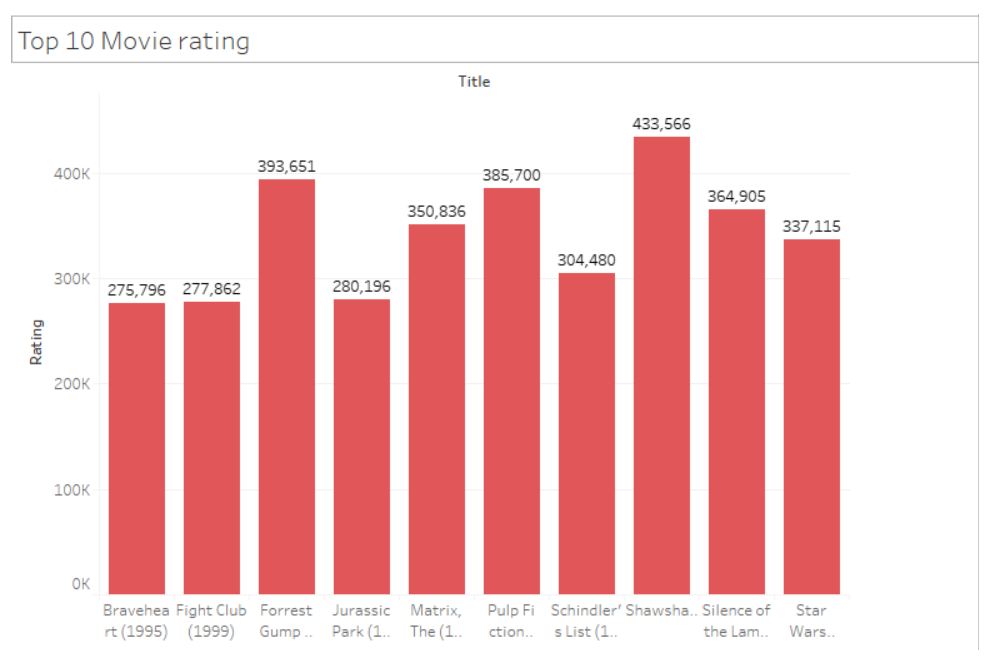
The list of task we can pre-compute includes:

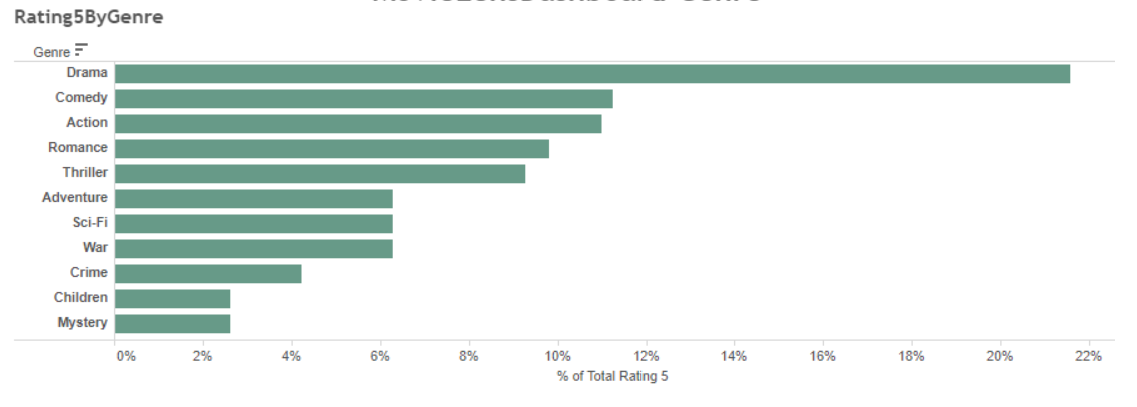
* Loading and parsing the dataset. Persisting the resulting RDD for later use.
* Building the recommender model using the complete dataset. Persist the dataset for later use.

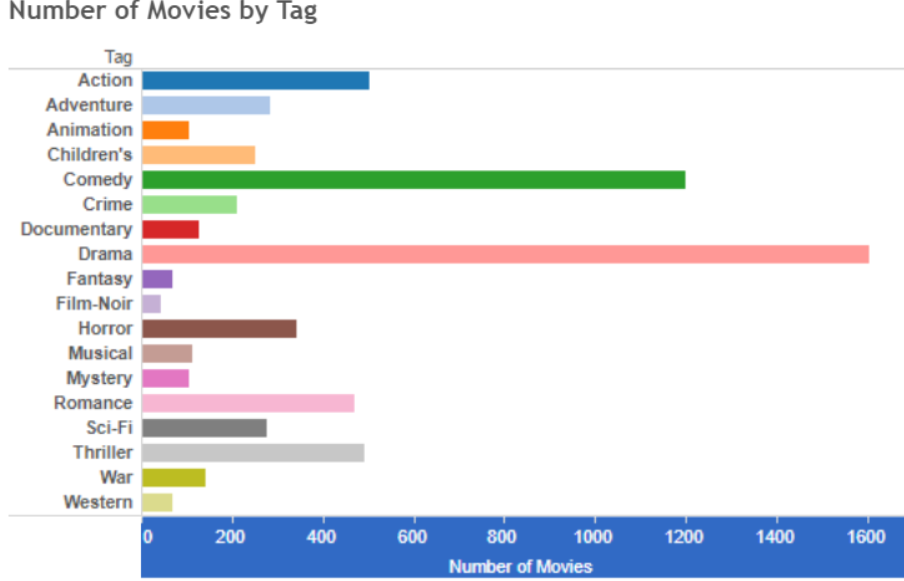
The format of these files is uniform and simple, so we can use Python [split()](https://docs.python.org/2/library/stdtypes.html#str.split) to parse their lines once they are loaded into RDDs. Parsing the movies and ratings files yields two RDDs:

* For each line in the ratings dataset, we create a tuple of (UserID, MovieID, Rating). We drop the *timestamp* because we do not need it for this recommender.
* For each line in the movies dataset, we create a tuple of (MovieID, Title). We drop the *genres* because we do not use them for this recommender.
* **DATA VISUALIZATION**

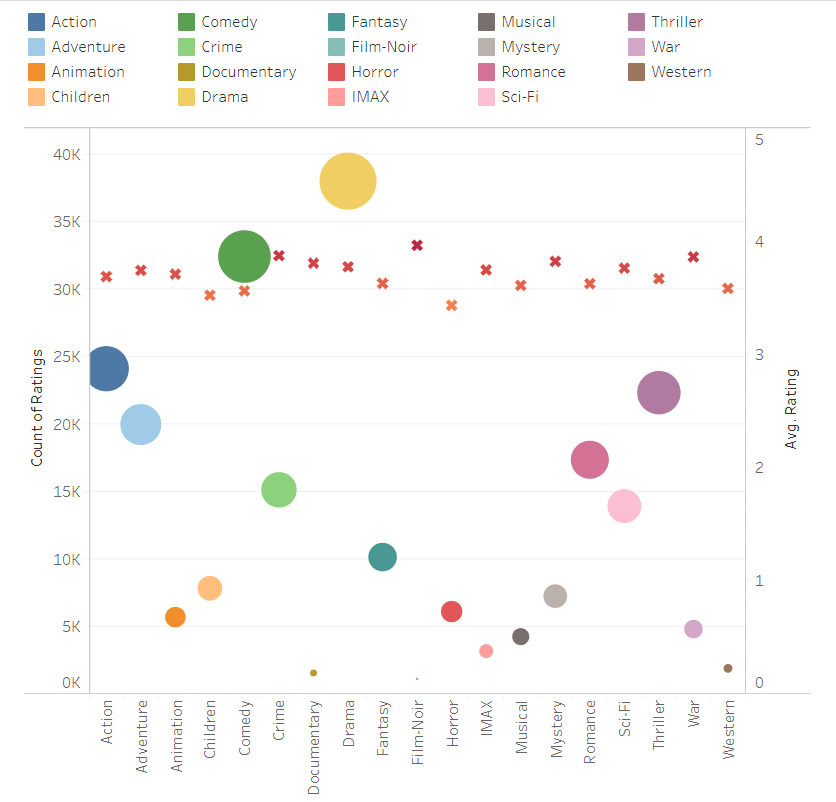
Data visualization is that the graphical representation of data and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible thanks to see and understand trends, outliers, and patterns in data. within the world of massive Data, data visualization tools and technologies are essential to research massive amounts of data and make data-driven decisions.



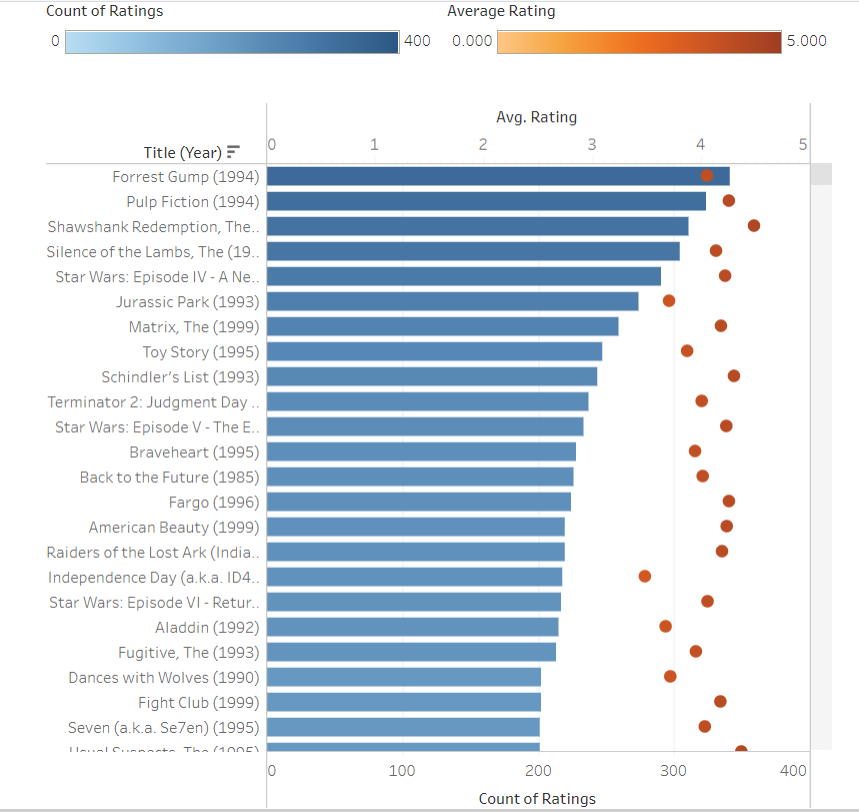


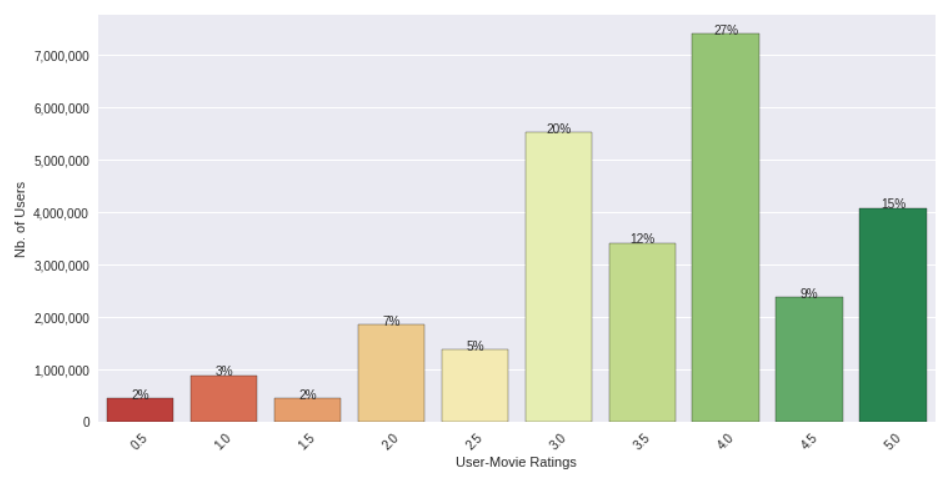


Popular Genre – Count of rating for Genre vs Average rating



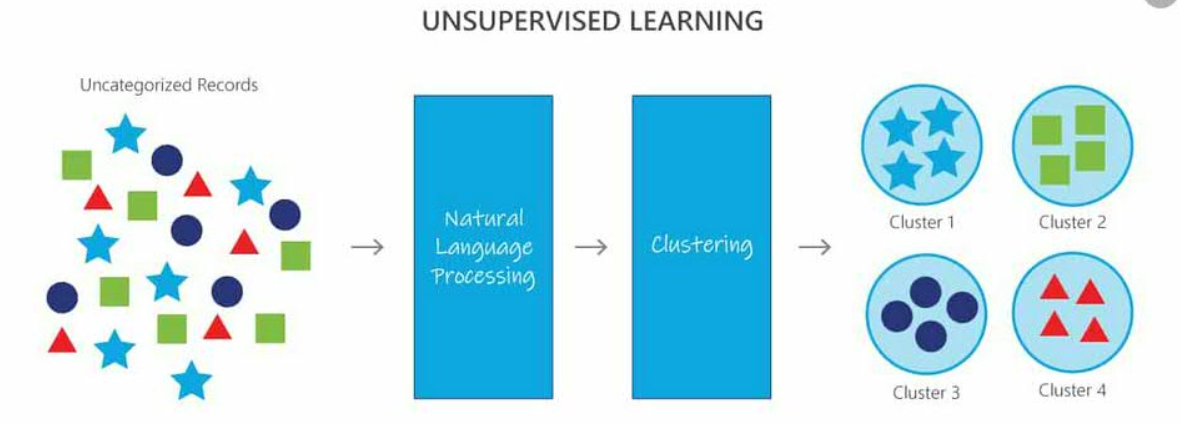
Movie Popularity – Total count and average rating





* **MACHINE LEARNING ALGORITHMS**
* **NLP Clustering**

**Clustering** is one of the most common exploratory data analysis technique used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. In other words, we try to find homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as Euclidean-based distance or correlation-based distance. The decision of which similarity measure to use is application specific.



* **Implementation:**

We have created a ML-Model which has been trained to analyse parts of the data-set that include text as its parameters. This method helps us to segregate our dataset into clusters which can then yield insightful information. The created clusters consist of data that has been grouped according to various parameters, such as similar actors, similar ratings.

* **Data Ingestion:**

Data is ingested from the CSV file and used for training the models.



Clusters are created using K-Means Algorithm.

* **K means:**

**K-means** algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster’s centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters; the more homogeneous (similar) the data points are within the same cluster.

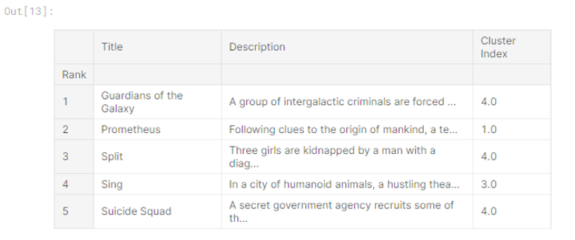


* **K-means applied to our Dataset:**

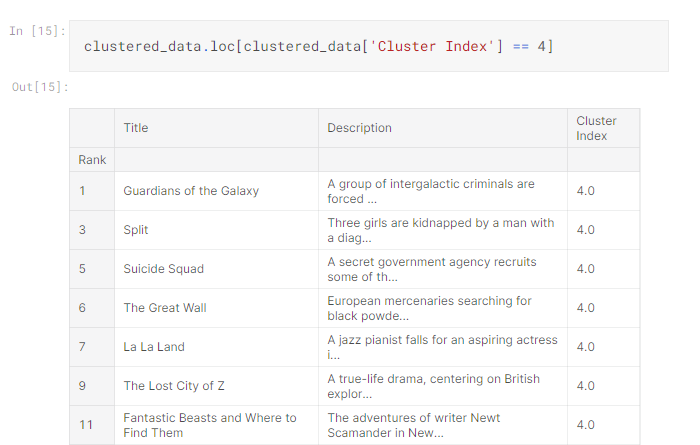
We can use K-Means to derive various incites about our dataset.

The following are some results that have been derived after cluster formation of our datset.

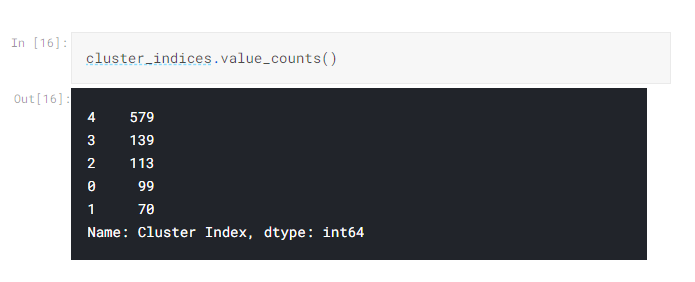
* Centre of the cluster that "Guardians of the Galaxy" is a member of:



* Other movies are also part of this cluster:



* Counts of movies in other clusters:



* **Alternate implementation using NLP Clustering and Glove:**

**Data Pre-processing:**

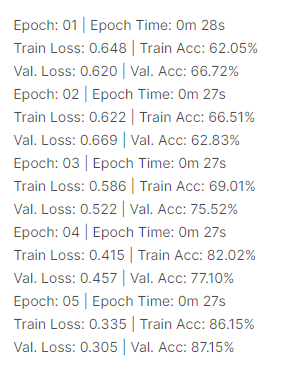
To prepare the data we will set the seed, define the Fields, and get the train/valid/test splits. Then we will be using packed padded sequences, which will make our RNN only process the non-padded elements of our sequence, and for any padded element the output will be a zero tensor. We then download the dataset and divide the data into training/validation/test sets. Next, we use pre-trained word embeddings, instead of having our own word embeddings initialized randomly. We use "glove.6B.100d" vectors". Glove is the algorithm used to calculate the vectors, go here for more. 6B indicates these vectors were trained on 6 billion tokens and 100d indicates these vectors are 100-dimensional. Pre-trained vectors already have words with similar semantic meaning close together in vector space. This gives the embedding layer a good initialization as it does not have to learn these relations from scratch.

**Building of the model:**

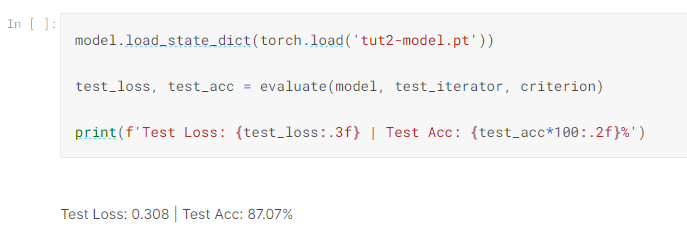
This model does not learn the embedding for the token, because we explicitly tell the model that padding tokens are irrelevant to determining the sentiment of a sentence. We use nn. RNN and implement bidirectionality and add additional layers by passing values for the num\_layers and bidirectional arguments for the RNN/LSTM. The final hidden state, hidden, has a shape of [num layers \* num directions, batch size, hid dim]. These are ordered: [forward\_layer\_0, backward\_layer\_0, forward\_layer\_1, backward\_layer 1, ..., forward\_layer\_n, backward\_layer n]. As we want the final (top) layer forward and backward hidden states, we get the top two hidden layers from the first dimension, hidden [-2,:,:] and hidden[-1,:,:], and concatenate them together before passing them to the linear layer (after applying dropout).



We train our model and below you can see its performance on the validation dataset.



Once we have done our hyper parameter tuning/optimization we use our final model of on test set and get the following accuracy.



Thus, NLP Clustering approach towards understanding and analyzing natural language data could prove instrumental in unlocking the enormous value and insights currently trapped within it. It vastly improves our understanding of the dataset and harnessing the information gained to make enormous contributions towards development.

* **BUILDING A RECOMMENDER SYSTEM**

We have tried to implement two widely used recommender system i.e. collaborative and content based

|  |  |
| --- | --- |
| **Collaborative** | **Content Based** |
| The main idea behind collaborative filtering is inter-user similarity based on their respective preferences which is calculated on past score of an item | The main idea behind content based recommender system is to recommend items that are similar to what the user liked before and to calculate the similarity between items. |
| We use mllib library which supports model based collaborative filtering where prediction of missing entries is calculated by users and products and their dependent factors through user-item association matrix. | It uses a well know concept of TF-IDF (Term frequency - Inverse Document Frequency) where the occurence of a term is calculated in the document(TF) and whole collection (IDF) |
| Matrix factorization solves two problems  1-It provides better movie ratings personalized for users as models factorizes the rating matrix into user and movie representations  2-It helps in improved ability to recommend less-known movies | as mentioned above we get the word importance my multiplying the two values i.e. TF \* IDF which results into a matrix where column represents all the words that appear in at least one document and each row represents a movie |
| Collaborative filtering in implemented using ALS (alternate least square factorization) method. In order to determine the best ALS parameters, we will use a small dataset. We need first to split it into train, validation, and test datasets. | scikit-learn gives you a built-in TfIdfVectorizer class that produces the TF-IDF matrix in a couple of lines |

* **Alternating Least Square (ALS) with Spark ML**

Alternating Least Square (ALS) is a matrix factorization algorithm which is implemented in Spark ML and capable of computing large-scale parallel filtering problems where the regularization parameter lambda which solves each least squares problem by the number of ratings is upscaled so that the user-generated in updating user factors, or the number of ratings the product is received in updating product factors.

* **TFIDF**

Although there are different methods to calculate similarity scores which are used in different scenarios we decided to choose the cosine similarity score since it is independent of magnitude and is relatively easy and fast to calculate and use sklearn's linear\_kernel() instead of cosine\_similarities() since it is faster.

A common method called Vector Space Model which models an item is used for calculating the weight of keyword extracted of the item by the model through TF-IDF. Thus, it emphasizes the significance of different genres from the user profile and movies can be sorted and top N candidates can be recommended to suggested users based on sum-product

* **Choosing between collaborative and content-based**

Even though the collaborative filtering technique has its outstanding advantage, its other side of the coin is also apparent: it cannot resolve the “cold start” problem. This problem refers to things where a replacement item or a replacement user added to the system and therefore the system has no thanks to either promote the item to the consumers or suggest to the user any available options. This is thanks to that the system doesn’t keep track of the properties of users and items. Unless users start rating the new item, it will not be promoted; and likewise, the system has no idea what to recommend until the user starts to rate. And content filtering is the solution to it. It enables the system to understand users’ preferences when the user/item profiles are provided.

* **Evaluating parameter for comparing the result is RMSE values**

Root Mean Square Error (RMSE) is widely adopted in many recommendation systems to measure the difference between the predicted scores and users’ actual ratings, The above two metrics are well known within the field of knowledge science and machine learning which leaves me with nothing to speak about them. But one thing to note is that they are not complete withing themselves in case of Recommendation Systems i.e RMSE value of 0.8766 for an algorithm doesn’t mean anything until there is another algorithm with another RMSE value with which we can compare our current RMSE value. To be honest, MSE or RMSE doesn’t matter in the real world. What matters the most is which movies you post in front of a user in top N recommendations and how users react to it. In 2006, Netflix offered 1M dollars to its users during a competition supported RMSE score in-order to enhance its recommendation system. It would are better if Hit Rate was used rather than RMSE scores.

* **Hive queries**

Well most of us have done some basic SQL, and Hive is very similar to SQL. It is fastest way to work with huge datasets if a person does not know Java and wants to work on Big Data. In fact, it is faster than writing MapReduce jobs. Hive automatically creates a plan for the query and submits it to the Hadoop cluster.

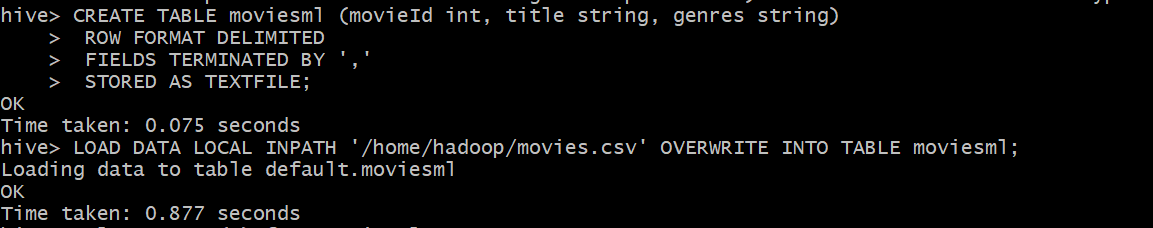
***Hive queries***

* Create Table for Movies

CREATE TABLE moviesml (movieId int, title string, genres string) ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/hadoop/movies.csv' OVERWRITE INTO TABLE moviesml;

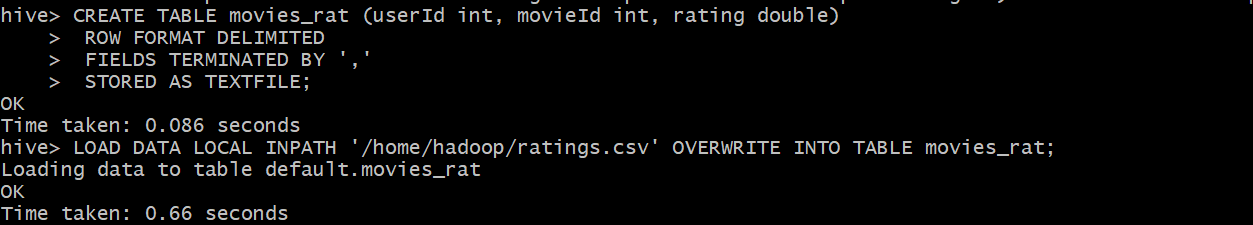


* Create Table for Ratings

CREATE TABLE movies\_rat (userId int, movieId int, rating double) ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/hadoop/ratings.csv' OVERWRITE INTO TABLE movies\_rat;

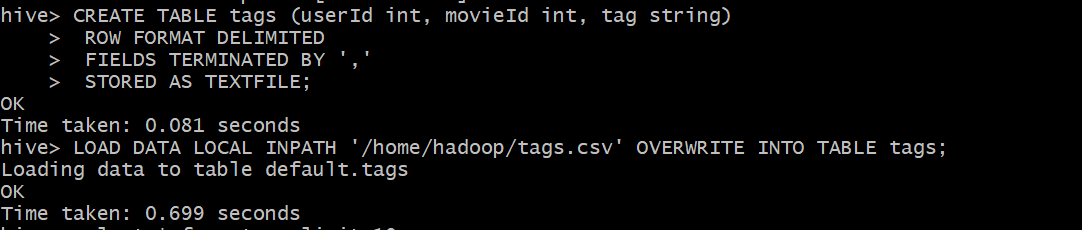


* Create table for Tags

CREATE TABLE tags (userId int, movieId int, tag string) ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

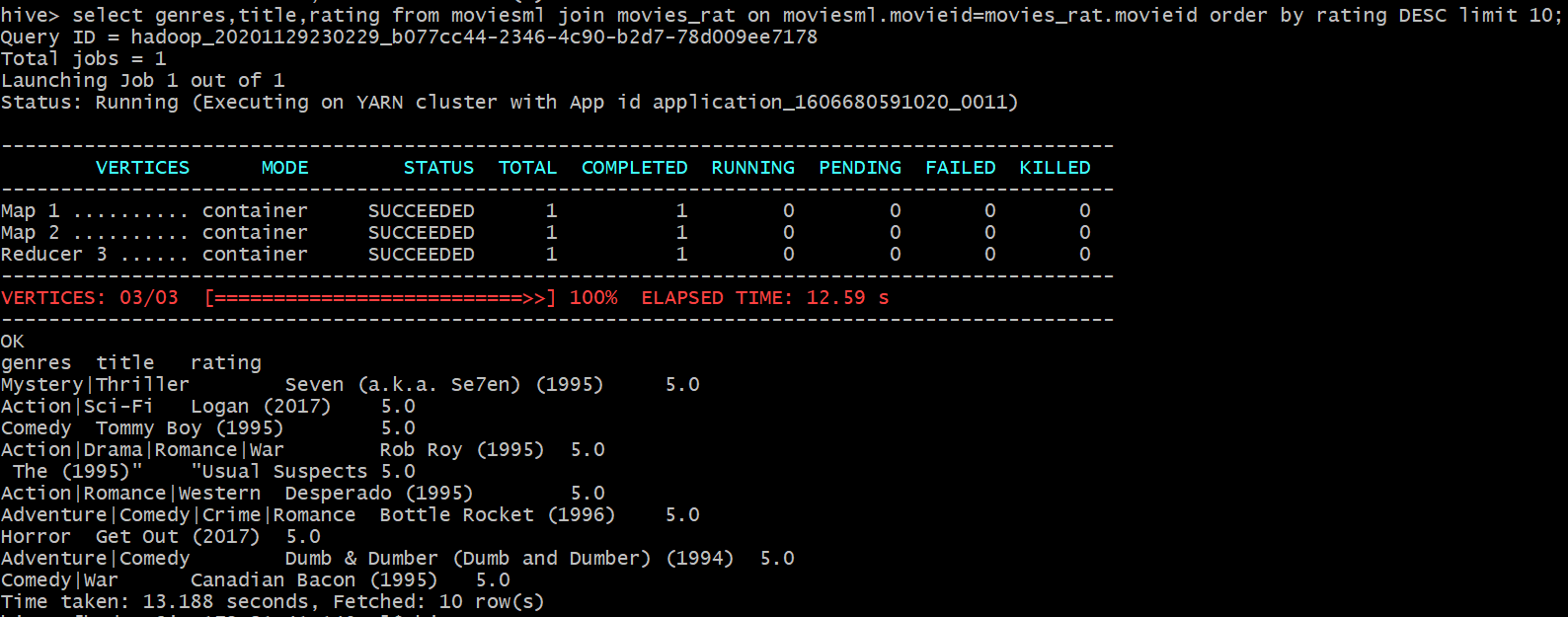
LOAD DATA LOCAL INPATH '/home/hadoop/tags.csv' OVERWRITE INTO TABLE tags;



1. To find the last 10 movies with highest rating with its genre, title, rating columns.

**Query:**

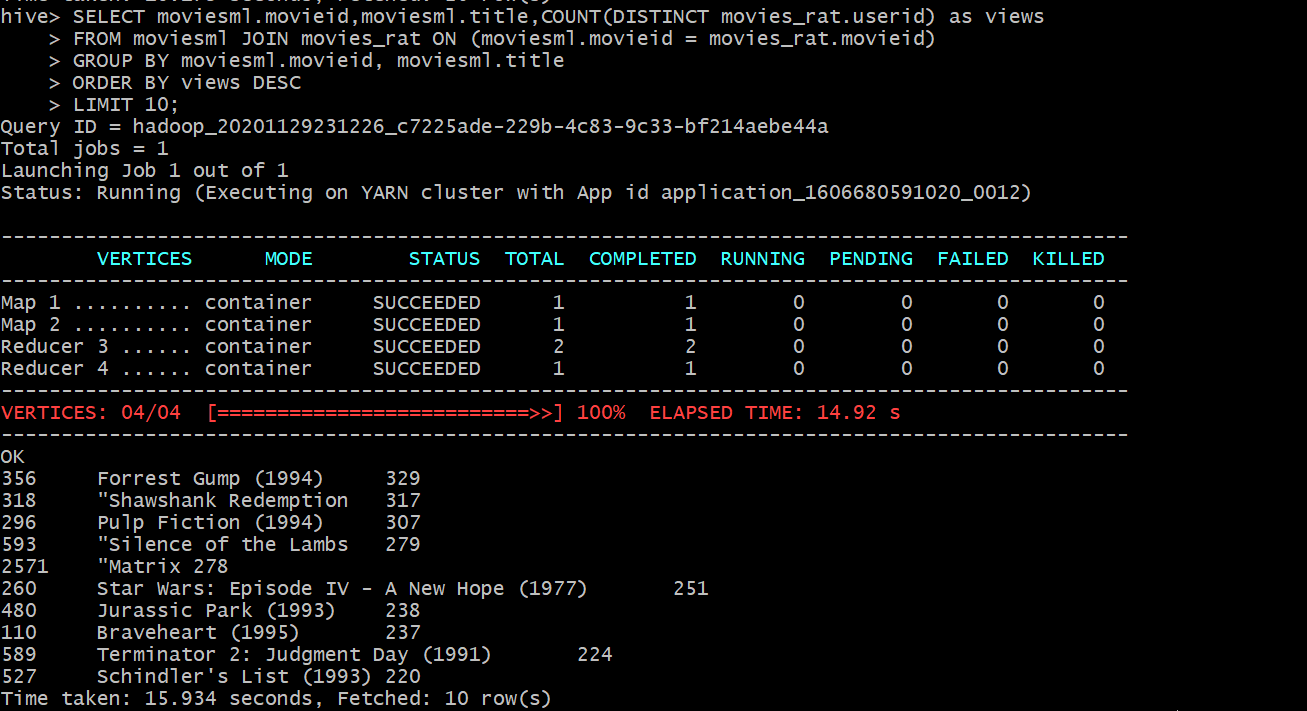
select genres,title,rating from moviesml join movies\_rat on moviesml.movieid=movies\_rat.movieid order by rating DESC limit 10;



1. To get the number of user ratings (based on highest number of ratings) to movies.

**Query:**

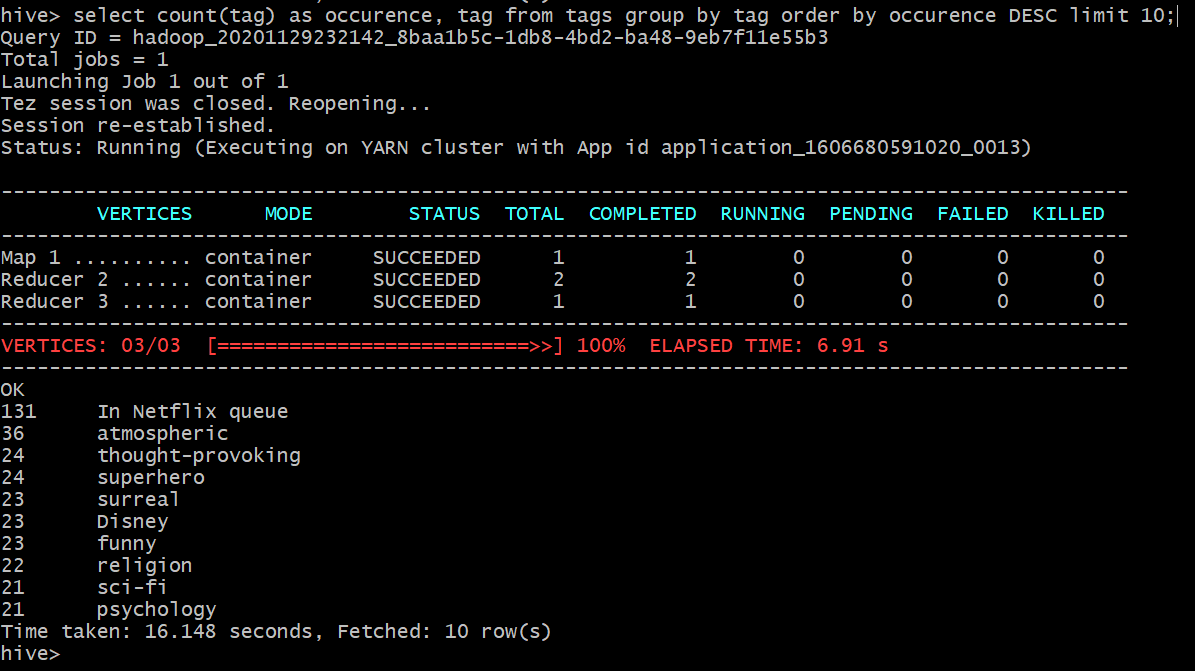
Select moviesml.movieid,movieml.title,COUNT(DISTINCT movies\_rat.userid) as views from moviesml JOIN movies\_rat ON (moviesml.moviesid = movies\_rat.movieid) GROUP BY moviesml.movieid, moviesml.title ORDER BY views DESC LIMIT 10;



1. To get the number of tags (source of rating) for movies (based on highest number of tags)

**Query:**

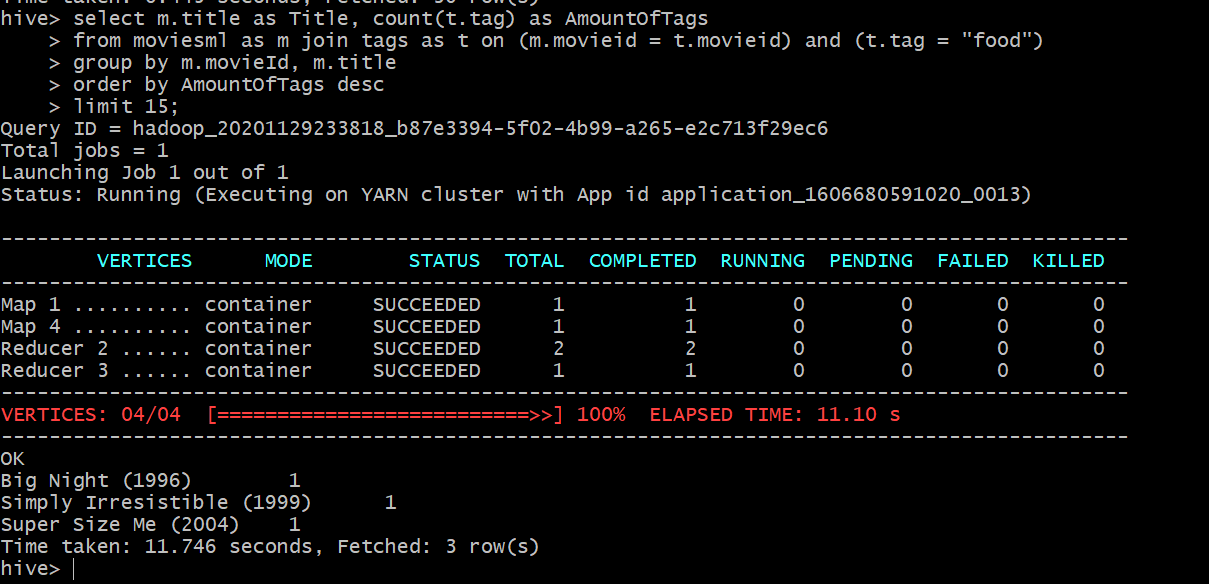
select count(tag) as occurence, tag from tags group by tag order by occurence DESC limit 10;



1. Which 15 movies (titles) have been most frequently tagged with the label "food"?

**Query:**

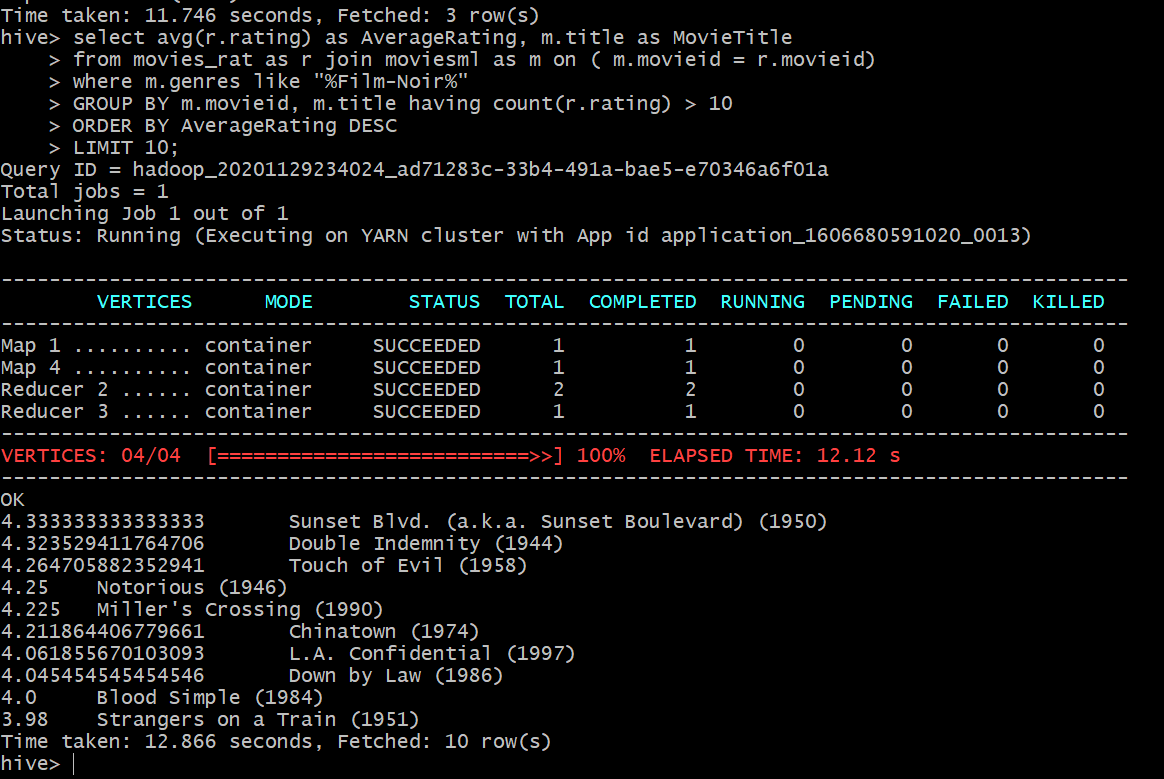
select m.title as Title, count(t.tag) as AmountOfTags from moviesml as m join tags as t on (m.movieid = t.movieid) and (t.tag = "food") group by m.movieId, m.title order by AmountOfTags desc limit 15;



1. Which are the highest-rated "Film-Noir" movies with more than 10 ratings?

**Query:**

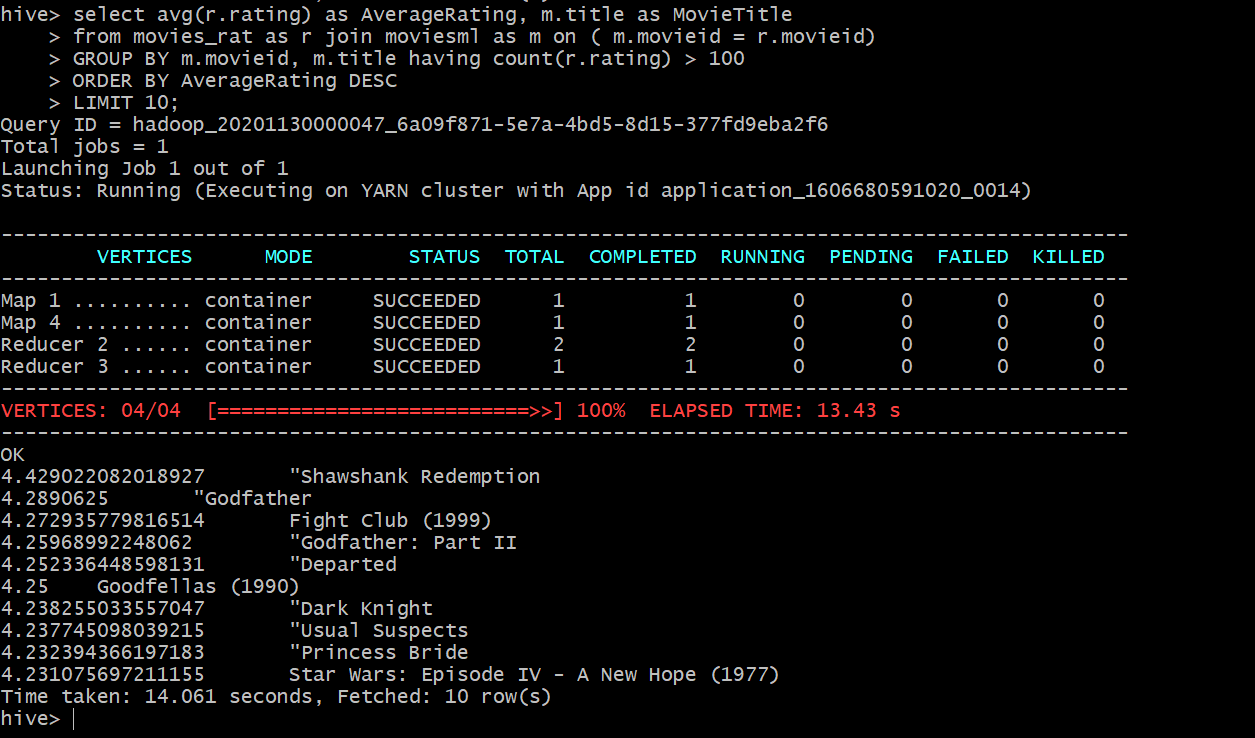
select avg(r.rating) as AverageRating, m.title as MovieTitle from movies\_rat as r join moviesml as m on ( m.movieid = r.movieid) where m.genres like "%Film-Noir%" GROUP BY m.movieid, m.title having count(r.rating) > 10 ORDER BY AverageRating DESC LIMIT 10;



1. Which are the 10 best-rated movies (on average; list titles) with more than 1000 ratings?

**Query:**

select avg(r.rating) as AverageRating, m.title as MovieTitle from movies\_rat as r join moviesml as m on ( m.movieid = r.movieid) GROUP BY m.movieid, m.title having count(r.rating) > 100 ORDER BY AverageRating DESC LIMIT 10;



* **RECOMMENDER SYSTEMS**

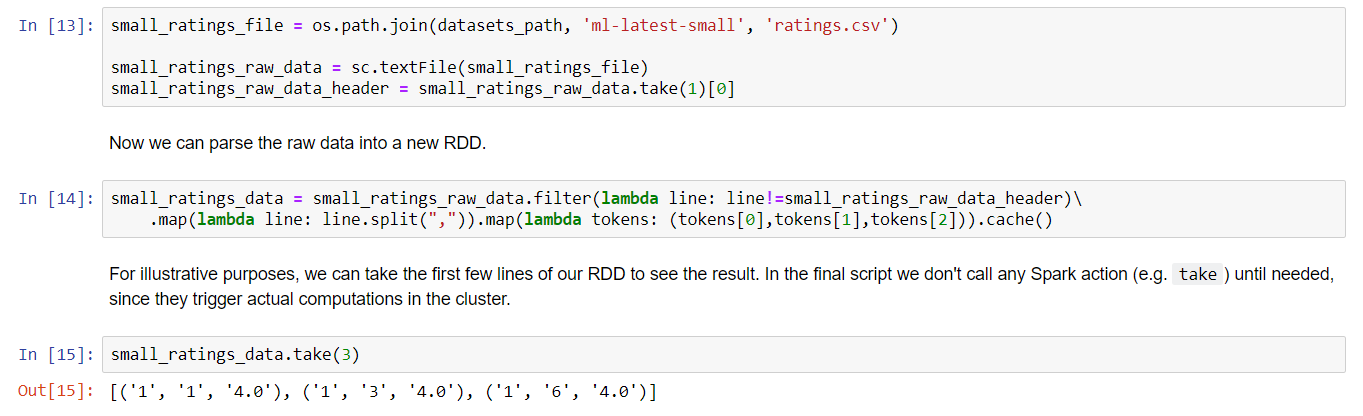
The recommendation system may be a reference for recommending and providing consumers to shop for goods. These suggestions are supported many choices, like what products do consumers buy? Which movie did the buyer see? Alternatively, what articles do consumers read online? thanks to the explosive growth of digital information and therefore the increasing number of tourists using the network, information overload has become a possible challenge nowadays. A recommendation system may be a system that helps users filter information. Its core task isn't only to filter information effectively but also to seek out users’ preferences and provides users interesting information. With the support of the recommender system, the flooding of data and therefore the complexity of online search is often reduced and the convenience of searching and filtering network data are often improved.



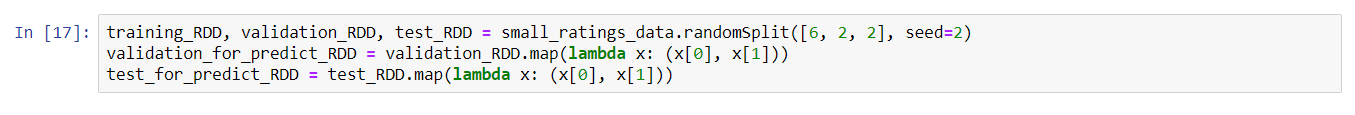
Since we are implementing the recommender system with the help of Spark we used pyspark module in python as above

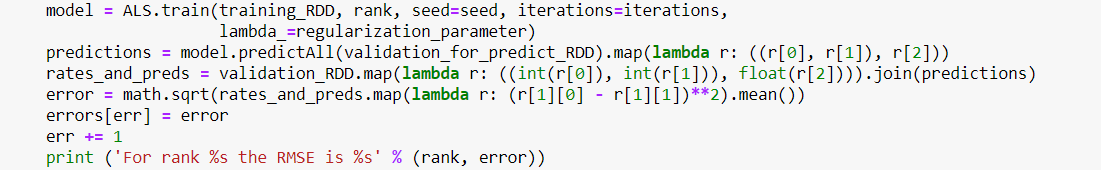
**1- Collaborative Filtering:**

Creating RDDs for further processing



* **Alternate Least Square (ALS) implementation where select hold evaluation with train and test dataset**

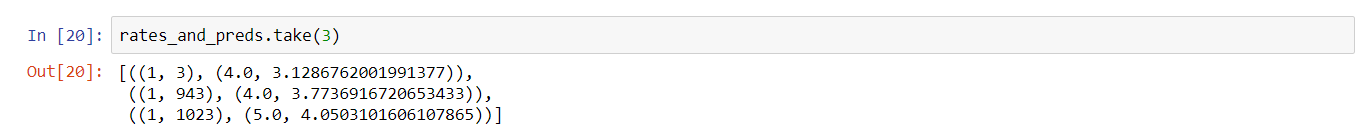




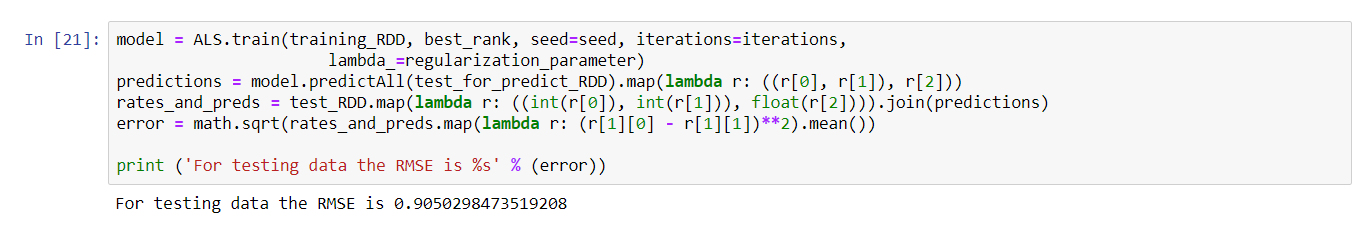
Basically, we have the UserID, the MovieID, and the Rating, as we have in our ratings dataset. In this case the predictions third element, the rating for that movie and user, is the predicted by our ALS model.



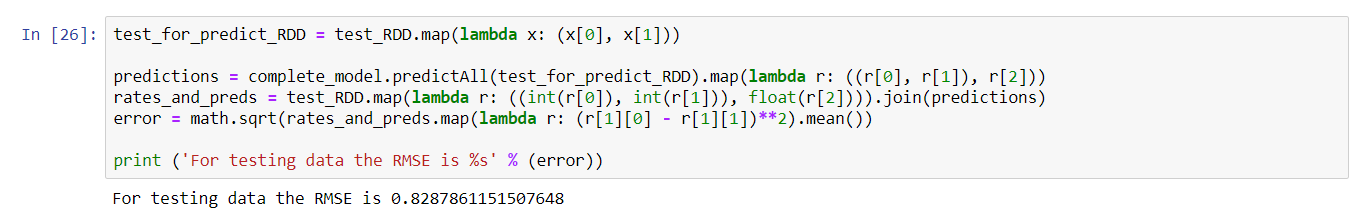
Then we join these with our validation data (the one that includes ratings) and the result looks as follows:



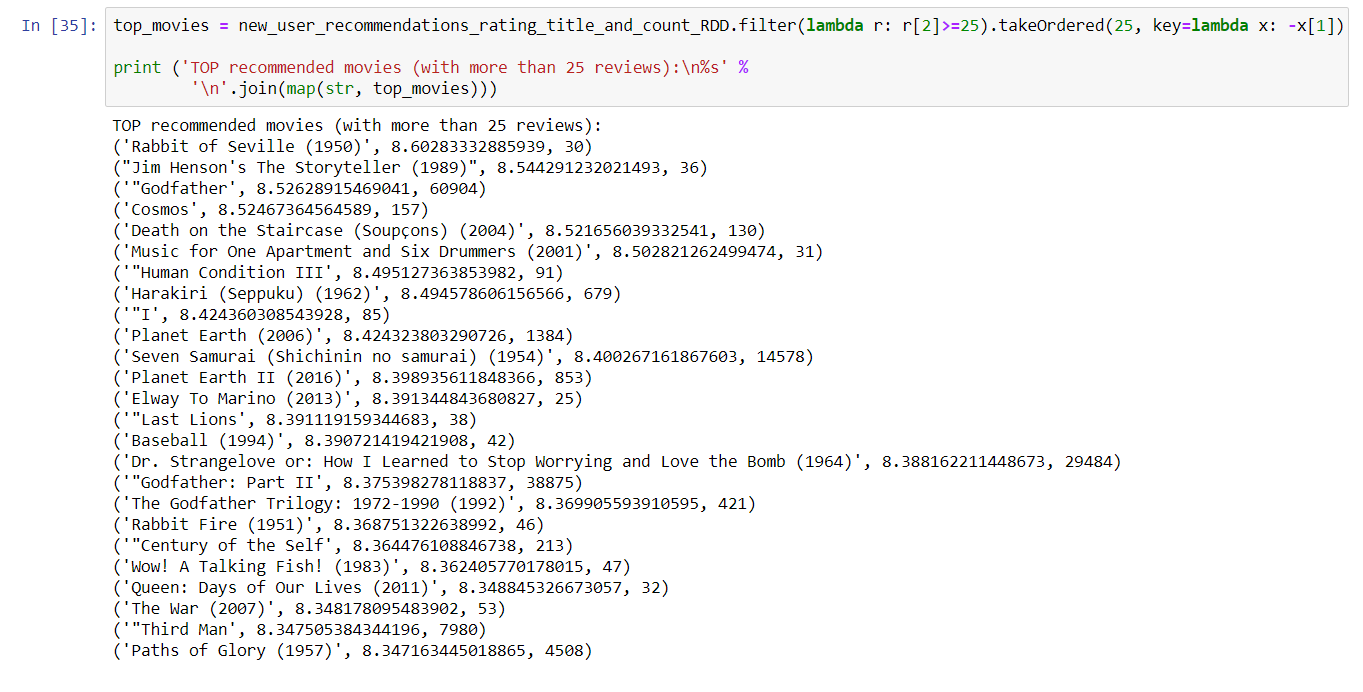
Finally, we select the model with the RMSE value as follows:



RMSE value for the test dataset

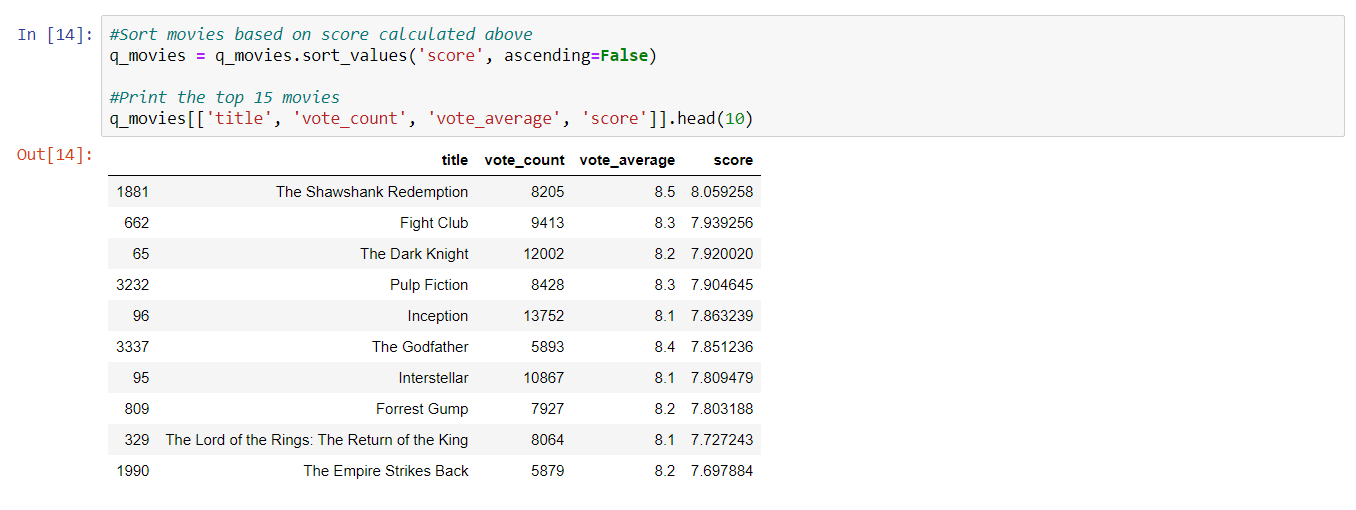


Getting the recommendations:

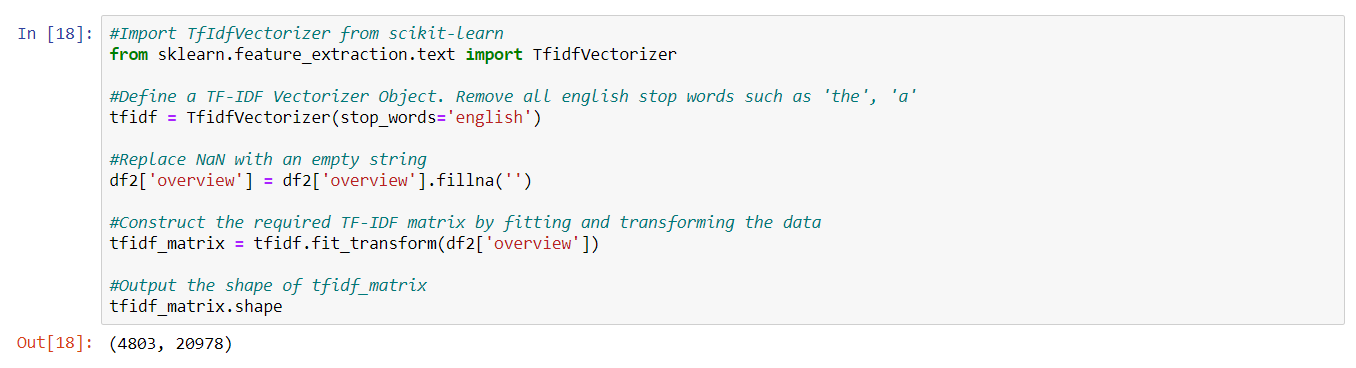


**2- Content based filtering**

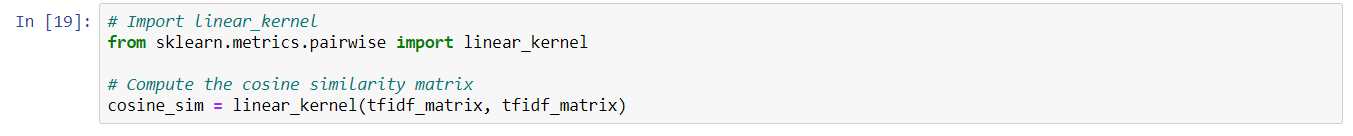
We need score for further processing which calculated as follows



* **TFIDF concept implementation**

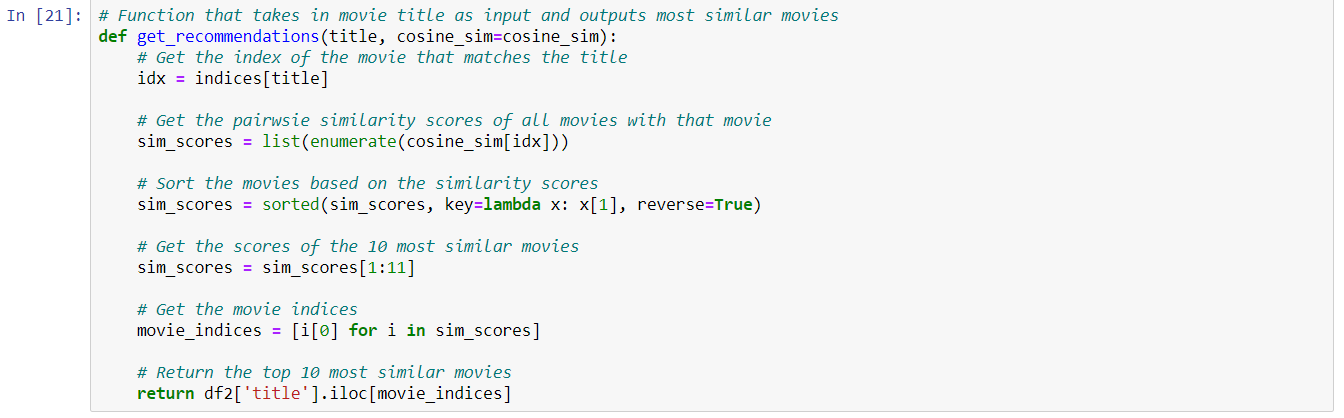


Since we have used the TF-IDF vectorizer, calculating the dot product will directly give us the cosine similarity score. Therefore, we will use sklearn's **linear\_kernel()** instead of cosine\_similarities() since it is faster.

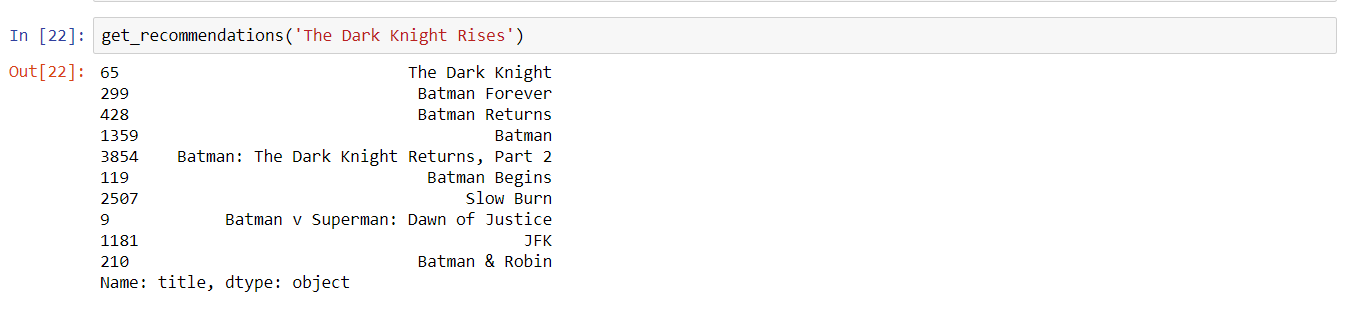


We are now in a good position to define our recommendation function. These are the following steps we'll follow: -

* Get the index of the movie given its title.
* Get the list of cosine similarity scores for that particular movie with all movies. Convert it into a list of tuples where the first element is its position and the second is the similarity score.
* Sort the aforementioned list of tuples based on the similarity scores; that is, the second element.
* Get the top 10 elements of this list. Ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself).
* Return the titles corresponding to the indices of the top elements.



Display the recommendations



* **CONCLUSION AND FUTURE WORK**

We have tried to implement machine algorithms models and build a recommender system using collaborative and content- based filtering. While both the methods are useful in their own ways, **Hybrid Systems** can take advantage of content-based and collaborative filtering as the two approaches are proved to be almost complimentary. This model was very baseline and only provides a fundamental framework to start with. We also tried to make use of Bigdata technologies such Hadoop, Apache Spark, Hive, mllib libraries to efficiently process large dataset with ease and efficiency.

In our effort to implement this we certainly faced some challenges which we can try to troubleshoot and implement as future scope to further enhance the overall productivity and efficiency of the project.

* To begin with use of pig queries for gaining data insights can be fruitful since Pig is an efficient tool for bigdata query processing.
* In data science, more the data better the prediction; although we used data with about 1-25M rows, we can use larger datasets to get better accuracy.
* The above project is an implementation for static dataset where we process a csv datafile, we would try to implement the more dynamic dataset based a web based application where the data recommendation will be on the fly.
* **STUDENT’S CONTRIBUTION**

Since this was a group project the overall workload was distributed for each group members as individual tasks but, each member of our group has contributed directly or indirectly to every step of the project. This was every member was able to contribute more to the overall success of the project. The following table provide individual responsibility and contribution not confined to mentioned areas but person responsible to ensure its successful completion.

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Responsibility to complete the milestone** | **Major Contribution but not limited to the areas as follows** |
| Sneja | Data gathering, preprocessing and analysis | Data preprocessing  Content based Recommender system  Data visualization |
| Akhil | Recommender System | Data gathering and analysis  Collaborative Recommender system,  Project report |
| Sriram | Machine learning models | Literature review  Data visualization  Machine learning models |
| Darshan | Visualizations and the project report | Data preprocessing  Machine learning models  Project report |

* **REFERENCES**

[1] Wang, Lidong & Alexander, Cheryl. (2016). Machine Learning in Big Data. International Journal of Mathematical, Engineering and Management Sciences

[2] Shevate, P. (2016). Sentiment Analysis for Movie Reviews using NLP Prachi Shevate (Team 5) California State University, Sacramento. Retrieved 2016

[3] Choudhry1, N., Xie2, J., & Xia3, X. (2020, June 01). IOPscience. Retrieved November 12, 2020

[4] Zhenlin Kan, Xinru Cheng, Seung Hyun Kim, Yuting Jin. Apache Hive-Based Big Data Analysis of HealthCare Data. International Journal of Pure and Applied Mathematics Volume 119 No. 18 2018, 237-259.

[5] C. Chiru, C. Preda, V. Dinu and M. Macri, Movie Recommender system using the user's psychological profile, 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, 2015

[6] C. M. Wu, D. Garg and U. Bhandary, Movie Recommendation System Using Collaborative Filtering, 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018

[7] Yin, N. (2019, December 31). A Big Data Analysis Method Based on Modified Collaborative Filtering Recommendation Algorithms. Retrieved December 09, 2020, from https://www.degruyter.com/view/journals/phys/17/1/article-p966.xml

[8] Real Python. (2020, November 07). First Steps With PySpark and Big Data Processing. Retrieved December 09, 2020, from https://realpython.com/pyspark-intro/

[9] Collaborative Filtering - RDD-based API. (n.d.). Retrieved December 09, 2020, from https://spark.apache.org/docs/latest/mllib-collaborative-filtering.html