13 *fig:Figure 2*25 *fig:Figure 3*35
*fig:Figure 4*45 *fig:Figure 5*55 *fig:Figure 6*65

Data Analysis on Movie Lens Dataset to gather business essential insights

Vikramaditya Tatke
MSc Data Analytics
National College of Ireland
Dublin, Ireland
x17163668@student.ncirl.ie

***Abstract*—In recent years internet connectivity has evolved significantly making it possible to maintain big cloud storages even for the average consumers. This has led to the adoption of streaming services by the consumers. These days consumers are provided with a plethora of options. This has provoked the need to implement recommendation systems. But for new users recommendation engines cannot function due to the complete lack of user-specific data. Until the user data generated it is possible to provide generic recommendation based on the users around them or based on the current trends. This paper talks about such generic queries which can help the existing recommendation systems to deal with new users. Through this project we have carried out experiments on the MovieLens dataset and discussed the results of data analysis. All of our results can be used to provide customers with initial recommendations. Our results are considerably trustworthy due to the sheer number of users who have provided ratings and tags to thousands of movies released in the past century.**

***Index Terms*—Big Data, HDFS, Map Reduce, Pig, Spark, Hive, Recommendation Stystems**

I. Introduction

Up until less than a decade ago, people have been recommending products and places to others, who were in need, based on their personal taste and opinion [1]. Although this gave way to better human interaction, it was not the most efficient way. This limited people's knowledge to their circle of friends or acquaintances. During the '90s collaborative filtering started gathering industry attention and Amazon was one of the first companies to effectively implement this technique to recommend products to their customers. This sparked academic interests as well, and a number of different types of recommendation systems started developing. [1]. In the recent years huge leaps have been made in the implementation of complex algorithms by top organizations as well as small scale organizations in order to facilitate business growth and gain better customer satisfaction and loyalty. Now-a-days recommendation systems have become a part of our lives as they are implemented by a plethora streaming and e-commerce platforms such as Netflix, Amazon, Spotify, XBox store, Apple App Store, etc. Recommendation systems are now embedded with advertisements as well.

With the increasing use of fiber optics by the broadband providers internet speed has seen a lot of growth. This has led to the adoption of digital content without any hardware media. Movies and music that used to occupy a lot a space on a computer's hard drive can now be very easily streamed from content providers. Also, there is a lot of content available online for the consumers to stream, which makes it problematic to find something that a person actually enjoys. To aid this, content providers have developed and implemented recommender engines, which take into consideration the preferences of the users to show the content they might enjoy on their homepage.

Providing customers with content they are more likely to consume has proven to be extremely beneficial for businesses. Furthermore, consumers genuinely benefit, as recommendation systems tend to reduce the problem of *"the paradox of choice"*. Collaborative filtering is often used in Web 2.0 recommendation systems, but it does come with a set of problems. Numerous attempts have been made to deal with these problems using hybrid techniques which will be discussed further below. Other methods such as content based filtering are also used by recommendation systems.

In section 2 we discuss the previous work carried out in relation to recommendation systems. In section 3 we describe the methodology used in our research, followed by section 4 in which we discuss and evaluate the results with the help of visualizations. [2].

II. Literature Review

In the following section we have reviewed a few papers and criticized them.

In a collaborative filtering approach a few neighbours (group of people or users with similar interests) are selected. In case, user interests cannot be determined properly this approach fails [2]. [1] talks about hybrid recommendation systems as an alternative to traditional collaborative filtering.

User and item based collaborative filtering approaches are used by the collaborative filtering algorithm, which give recommendations on user interaction and by finding items similar to those preferred by the user. Furthermore, deep learning can be used to extract features which are then applied to collaborative filtering [3]. In [3] authors have developed a word vector analysis model, trained and tested on the MovieLens dataset, to improve the accuracy of collaborative filtering by calculating item similarity based on Pearson correlation and film similarity using weighted summation. In [4], authors have

discussed about memory based collaborative filtering models that are scalable and easy to implement.

The failure of collaborative filtering algorithm due to the lack of data as discussed in [2] is due to the data sparsity problem which can be dealt with by providing recommendations based on target user's interest. This can be done with the help of feature selection and methods such at Precision-Recall at k to provide top N items as recommendations to the new users, then combining them with content-based filtering [4].

One more technique to deal with the sparsity problem is to use a weighted k-means algorithm that recommends items based on the weights assigned to the items. These weights are based on user interests [5].

The the purpose of our research is to solve comparatively smaller queries which can then help such techniques to perform better in the future.

III. Methodology

This section will discuss in detail the dataset, data preprocessing and storage along with the technologies used to answer 6 queries mentioned above. Furthermore, the process flow of the project is described with a flowchart below.

A. Dataset Details

The dataset used for this project consists of a 27 million ratings on 58,000 movies, mostly assembled for educational purposes. 2,80,000 users have provided ratings for all the users bases on a star system, that starts with 0.5 and increments by half a star to a maximum of 5 stars suggesting a particular user's strong liking to a specific movie.

The dataset consists of a total of 5 CSV files namely - Movies, Users, Ratings, Tags, Genome_tags Genome_scores. Movies and Ratings files contain movie details and user ratings respectively. Rest of the 3 files contain tag related information. According to [6] a genome tag is a number that measures the strength of a relationship between a user assigned tag and the movie ranging from zero (weak) to one (strong). We have a chance to deduce a relatively better and unbiased analytical conclusion from a large dataset such as the one chosen for this project, as it contains ratings and tags from a vast number of people, which can generally be more representative of a wider population of movie watchers. Furthermore, it has a good number of inter-related parameters which can lead to the formation of insightful queries.

B. Data Cleaning and Processing

All of the 6 CSV files were imported into 5 Pandas dataframes named appropriately using the 'read_csv' feature of the Pandas library before being checked for missing or NA values. The dataset did not consist of missing values, although there were a few fields in the genre column in the movies file that contained *"no genre"*. As only a few movies did not have a specific genre associated with them, these fields were not omitted. Moving on, we found some *","* within the title and tag columns of the movies and the genome_tag files, respectively. This interfered while importing data into our SQL Server database and hence the commas were removed. Further process is divided into different parts based on the technologies involved and is discussed below.

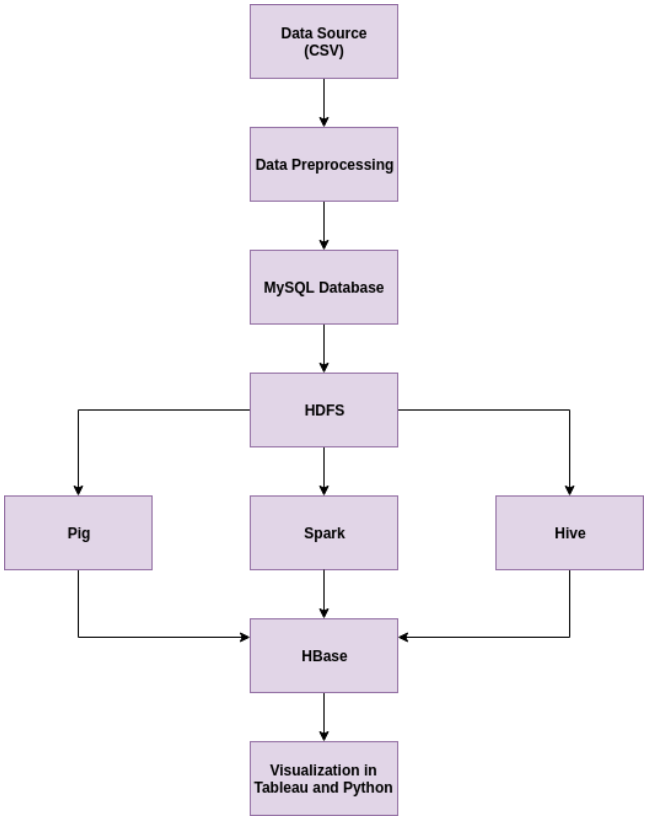

Figure 1. Project Flow

C. Technologies

1) Python: Python was used to import data from the CSV files into dataframes to carry out the required manipulation. Then the cleaned and processed dataframes were exported to new CSV files.

2) MySQL: A new database was created within MySQL for this project along with appropriately structured tables to accommodate these newly created CSV files, which were imported by running MySQL in local mode using the command *–local-infile=1*.

3) HDFS: Moving on, a Map Reduce job was performed on each of the MySQL tables in order to create files in Hadoop File System (HDFS). These files were created in separate directories for each of the tables and were then copied to the local file system using *-get*. Now the files were ready to be analyzed. We have chosen the following 3 technologies to answer our 7 queries.

4) Sqoop: Sqoop is a fast tool that facilitates data transfer from relational databases to hadoop file system. It is powerful and can handle large data transfers efficiently and can also

be used to transfer data to hive. In this project we have used sqoop to, initially transfer data from our MySQL database to HDFS and then to Hive.

5) Pig: Firstly, a **.pig** file was created with all the necessary commands, which was then run using the pig local grunt. Pig is more suited for easier queries, which require grouping, filtering, string matching, etc.

6) Hive: Built on top of HDFS, Hive is a data warehouse solution that is implemented by many tech giants including Apple, Netflix, Glassdoor etc. [**?**]. A centralized data warehouse is provided by implementing Hive for querying, analyzing, etc after getting data stored in HDFS. Hive also provides a querying language called HQL which offers a familiar SQL construct.

7) Spark: Spark provides superior performance and flexibility that is required while dealing with large amount of data. In this project we have utilized PySpark through Jupyter Notebook which made it simple to transform and query the data. Using dataframe manipulation alongside powerful SQL provides a great deal of flexibility and open up opportunities for gathering various insights.

8) HBase: Outputs of the map-reduce jobs, obtained after executing all the queries will be transferred to HBase from HDFS using Sqoop. As HBase is an efficient environment used to store big data, it will be suitable for storing our results. Furthermore, it runs on top of HDFS which makes it easier to transfer data.

9) Tableau: Tableau is another extremely powerful tool when it comes to data visualization. It can connect with various data sources such as Excel, MySql, .csv files, etc. Furthermore, it can generate appealing and informative visuals with ease for all types of data. All the query results generated were transferred into Tableau to generate visualizations.

D. Map-Reduce Design Patterns

1) ***Highest rated Comedy Movies****:* This map-reduce was carried out to identify the highest rated comedy movies in the entire dataset. It was performed in Pig to identify top 20 movies with the highest average rating.

2) ***Lowest Rated Action Movies****:* This map-reduce was carried out to identify the lowest rated comedy movies in the entire dataset. It was performed in Pig to identify 20 movies with the lowest average rating.

3) ***Most User Tagged Movies****:* We have implemented this map-reduce pattern in Hive, to count the number of tags assigned to each movie and then select the top 20 movies along with their respective tags. This study can also be beneficial to figure out the most popular movies, as the number of user tags provided to each movie is generally corresponds to the number of viewers.

4) ***Movies with most relevant Genome Tags associated with them****:* This map-reduce pattern assigns the relevance of genome-tags to the actual tag and the movies. This will help us identify the movies that have been assigned a genome-tag with the highest accuracy.

5) ***Change in the average rating for Drama movies over the years****:* Here we have used PySpark to make full use of it flexibility by create spark dataframes from the map-reduce output files and then query these dataframes using the inbuilt *'sqlContext'*.

E. ***Challenges Faced***

In our dataset, multiple genres were originally concatenated into a single column. As this was already a very big dataset it was not possible to introducing redundancy by assigning multiple genres to a single movie by increasing the number of rows. Running such a big dataset locally on a machine with only 16GB RAM proved to be extremely hectic. Initially, most of the Map-reduce jobs kept failing due to low java heapspace errors. To overcome this problem hadoop and java heapsizes were set to 8GB. Also, the default memory allocation to map and reduce were set to 8GB each. Most of the time the java processes involved in the used environments would take upto 1.5GB memory each. The ipython kernels had to be restarted several times when running pyspark queries because of memory erros. While trying out different operations on the data, the environment would overload and slow down to eventually stop responding altogether.

IV. Results and Evaluations

A. ***Highest rated Comedy Movies***

This map-reduce was carried out to identify the highest rated comedy movies in the entire dataset. It was performed in Pig by filtering the join of the movies and the ratings table, based on genre by using string matching. Thereafter, same movies with different ratings were grouped to calculate the average rating. Finally, 20 movies with the highest average rating were selected. It is important to note that the movies which had very few ratings were omitted altogether in order to get comparatively fair results.

B. ***Lowest Rated Action Movies***

We have again utilized pig to implement this map-reduce pattern, as Pig utilizes the a construct similar to SQL while also allowing the use of java regex format. This made it easier to find the movies falling under the action genre. Results were calculated and grouped similar to the above query. Although, this time we have gathered the lowest rated action movies of all time.

C. ***Most User Tagged Movies***

We have implemented this map-reduce pattern in Hive, which offers an extremely familiar SQL construct. To find the most tagged movies, we have joined movies and tags table and then we count the number of tags assigned to each movie before grouping the result by the title of the movie. Lastly, we

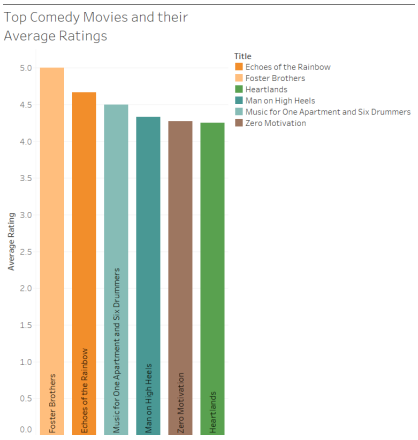

Figure 2. Highest Rated Comedy Movies

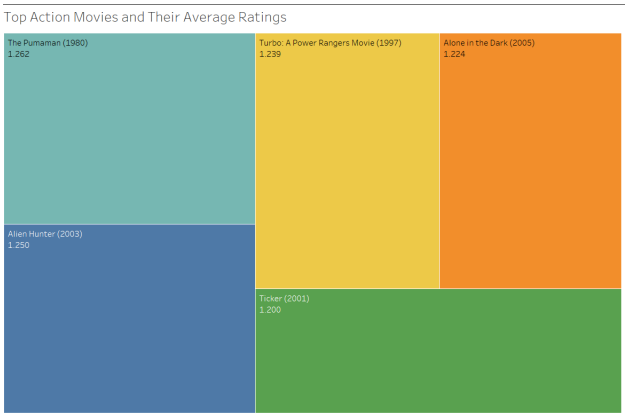

Figure 3. Lowest Rated Action Movies

limited the output to 20 rows and sorted in descending order of the count.

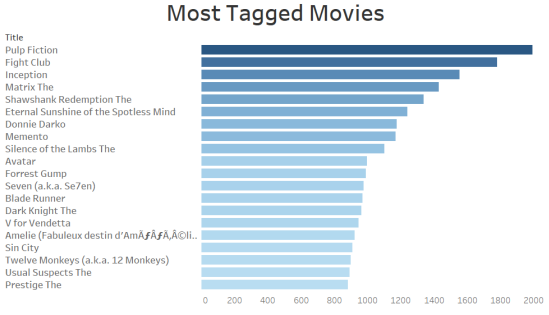

Figure 4. Most Used Tagged Movies

D. *Movies associated with most relevant Genome Tags*

This map-reduce pattern assigns the relevance of genome-tags to the actual tag. We can in turn apply this relevance to the movie the tag is associated with. In other words, we can identify the movies that have be been tagged most accurately using the Tag Genome model.

Figure 5. Movies associated with most relevant Genome Tags

E. *Change in the average rating for Drama movies over the years*

For this query we have made use of PySpark through Jupyter notebook. The map-reduced text files for movies and ratings tables were imported into PySpark. Then appropriate dataframes were created before combining these tables in a single large dataframe and registering it as a table, in order to perform SQL operations on it. Moving on, this large dataframe was used to calculate average ratings of Drama movies and to observe the change over time.

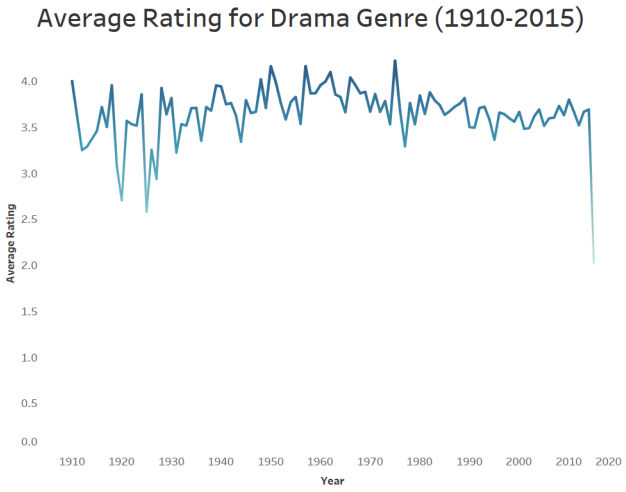

Figure 6. Change in the average rating for Drama movies over the years

V. Conclusion and Future Work

The results we obtained gave us clear insights into the huge database of 20 million rows. In further works it is recommended to properly sort genres to generate better insights. Also, integrating these results with machine-learning models

such as content-based filtering and collaborative filtering will be very useful for creating more efficient recommendation systems. This research has mostly focused on gathering insights that have some business rationale and hence lacks the complexity involved research that includes training and testing of machine learning models. Our research has provided some really accurate results such as, those provided by *the most tagged movies* query. If we perform a cross check against Google and IMDB reviews we can definitely sense the accuracy of the results.

REFERENCES

[1] M. D. Ekstrand, J. T. Riedl, J. A. Konstan *et al.*, "Collaborative filtering recommender systems," *Foundations and Trends® in Human–Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[2] S.-M. Choi, S.-K. Ko, and Y.-S. Han, "A movie recommendation algorithm based on genre correlations," *Expert Systems with Applications*, vol. 39, no. 9, pp. 8079–8085, 2012.

[3] G. Liu and X. Wu, "Using collaborative filtering algorithms combined with doc2vec for movie recommendation," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE, 2019, pp. 1461–1464.

[4] Y. Afoudi, M. Lazaar, and M. Al Achhab, "Impact of feature selection on content-based recommendation system," in *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, April 2019, pp. 1–6.

[5] M. T. Himel, M. N. Uddin, M. A. Hossain, and Y. M. Jang, "Weight based movie recommendation system using k-means algorithm," in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct 2017, pp. 1302–1306.

[6] J. Vig, S. Sen, and J. Riedl, "The tag genome: Encoding community knowledge to support novel interaction," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 2, no. 3, p. 13, 2012.