Movie Analysis

Anonymous

1. INTRODUCTION

Data analytics plays a major role in making several business decisions. The end goal is to mine information which can be further used for business development.

As we know, movies have a lot of influence in our lives today. Movies not only provide us with entertainment but also help us bond with people. Socially, movies are a safe and easy means to start getting to know a person. Thanks to the digital world, we can find a lot of information about a movie before watching it and gain a fair idea on what to expect. We can see the ratings and reviews of the movie and judge how good or bad it may be.

In this project, we analyzed movie datasets IMDb [2] and MovieLens [7]. There are several reasons for choosing these datasets. IMDb is one of the largest movie dataset consisting of movie information along with the cast and crew information. MovieLens contains user wise ratings of each movie. Detailed description of these datasets can be found in Section 2. Moreover, the explanation behind the data aggregation performed to combine the datasets can be found in Section 3.

As part of our analysis, we performed two data mining tasks: (i) clustering described in Section 4, and (ii) frequent itemset mining described in Section 5. We also performed two data analytics tasks: (i) visualization on three attributes described in Section 6, and (ii) pairwise comparison of two attribute pairs described in Section 7. Finally, any possible future work has been detailed in Section 8.

2. DATASET DESCRIPTION

2.1 IMDb dataset

The IMDb (Internet Movie Database) dataset [2] contains information on content such as movies, television series and programs. Following is the information related to movies and other content on IMDb:

2.1.1 Title information

The information on titles (movies, television programs) such as:

1. primaryTitle: Name of the title.
2. startYear: The year when the title was released.
3. runtimeMinutes: The runtime length of the title.
4. genres: The genres of the title.

Information such as localized title release, region where the localized title release was done, languages and many more is also available on the IMDb database.

2.1.2 Crew and Cast information

The information on crew and cast involved in the titles such as:

1. primaryName: Name of the cast/crew member.
2. birthYear: The birth year of the cast/crew member.
3. deathYear: The death year of the cast/crew member.

2.1.3 Crew and cast roles

The information on the roles of the crew and cast involved in the titles such as:

1. category: The category of the role.

2.2 MovieLens dataset

The MovieLens dataset [7] is a dataset on movie ratings from the users. The dataset contains information from many users on ratings for multiple movies. This makes the dataset interesting because we can perform frequent item-set mining on the dataset to find patterns on users and their movie interests. We can also use the information from the IMDb dataset and see if we can find any patterns based on genres or cast/crew members.

3. DATA AGGREGATION

The IMDB dataset[2] and the user ratings from MovieLens dataset [7] were both inserted into a MongoDB database. Following are the collections in the database after the insertion process:

- **Movie**: This collection contains all the documents with the movie's information detailed in Section 2.1.1 along with the id of the movie. Since we only wanted movies, we chose only those titles with the titleType short, tvShort, movie and tvMovie.
- **Person**: This collection contains all the documents with the person's information detailed in Section 2.1.2 along with the id of the person.
- **Person_Roles**: This collection contains all the documents with the person's profession information detailed in Section 2.1.3 along with the personId and movieId for the person and the movie respectively.

- **Ratings**: This collection contains all the documents with the ratings by each user for certain movies provided in the MovieLens dataset. The collection has the following attributes: userId, movieId, rating and timestamp. The ratings file in the MovieLens dataset contained its own movieId for the movies. Hence, we exchanged the movieId with the corresponding imdbId value provided in the links file of the dataset to maintain coherence between our collections.

The insertion of data was performed using pymongo module in Python. The relevant source code is provided in the file named **merge_datasets.py**.

4. K-MEANS CLUSTERING

K-Means clustering [3] is a popular algorithm that is used to partition the given data into groups for further processing and analysis. Hence, we chose to perform k-means clustering on the normalized average rating of the movies. In order to perform the k-means clustering, we took the collection consisting of the individual ratings from the user and found the average user rating for each movie. Then, we created a new field called the 'normalizedAvgRating' consisting of the normalized average rating. This field was used to determine the cluster to which the movie belongs to.

We first plotted the Sum of Squared Errors (SSE) curve for values of the number of clusters (k) ranging from k = 1 to k = 10. The plot can be seen in Figure 1.

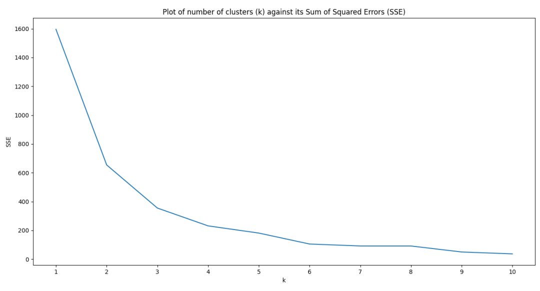

Figure 1: SSE - Sum of squared error

Based on the elbow method [1], we can determine that the optimal number of clusters is 3. Hence, we ran the k means algorithm with k = 3 and the results can be seen in Figure 2. We also ran the algorithm with k = 5 and its results are shown in Figure 3.

We performed k-means on a single dimension of data. Hence, for the scatter plots, we chose to plot ratings on both the axes for better presentation.

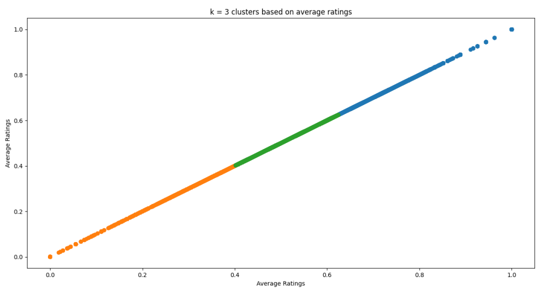

Figure 2: K Means clustering with k=3

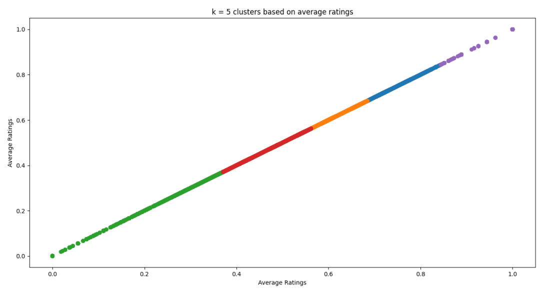

Figure 3: K Means clustering with k=5

We can clearly see that the distribution of points for each cluster in Figure 2 is much better than that of the points for each cluster in Figure 3.

5. FREQUENT ITEMSET MINING

Frequent itemset mining is a good way to understand certain patterns in the data. With respect to our dataset, the most natural frequent itemset to mine is the actors who worked together in a particular amount of movies. To obtain these frequent itemsets, we employed the Apriori algorithm [6]. We used the Person_Roles collection described in Section 3 to implement the algorithm. We first filtered the actor types roles (i.e., the category attribute is 'actor', 'actress', or 'self') and grouped the movies by each actor. In an effort to reduce the computation time of the algorithm, we limited the amount of entries to 15000 after applying the category filter. We ran the algorithm with a minimum support value of 5. The maximum number of actors observed in a frequent itemset was 6. In favor of brevity, we are not showing the frequent itemset of actors working together in at least 5 movies. Instead, we plotted two bar graphs; Figure 4 shows the number of itemsets in each level and Figure 5 shows the number of distinct actors in each level. Here, a level refers to each iteration of the Apriori algorithm. Each iteration generates the frequent itemsets of incremental size starting from 1.

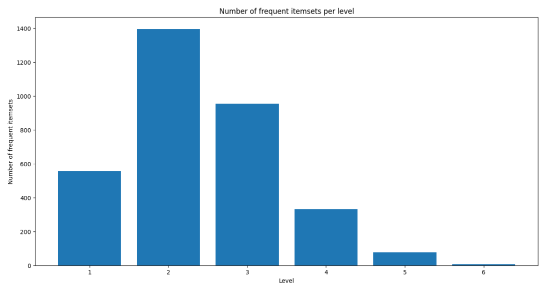

Figure 4: Number of frequent itemsets per level

Predictably, the number of distinct actors in each level keep on decreasing. However, the number of itemsets per level increase for $k = 2$ but decrease after that and eventually becomes 0 for $k = 7$ where the algorithm halts.

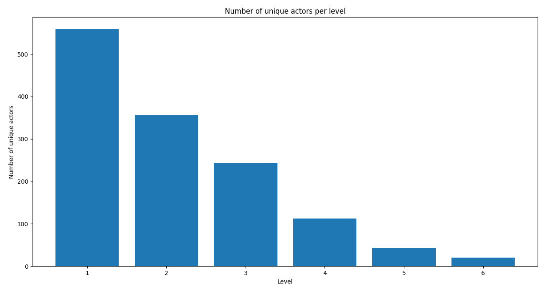

Figure 5: Number of distinct actors per level

6. VISUALIZATION

In order to visualize the data given, we have used Python along with PyMongo [5] and Matplotlib [4] modules to fetch data from Mongo database and plot the data respectively.

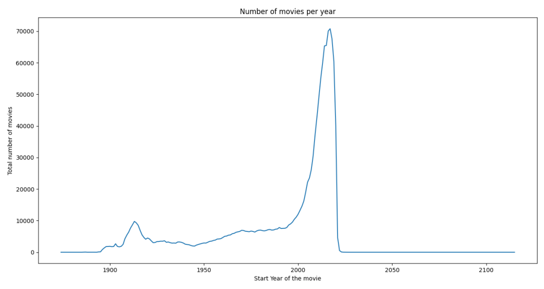

Figure 6: Time series graph - Movies per year

The graph shown in Figure 6 shows the number of movies released per year as a time series graph. From this figure, we can infer that around the second decade of 2000 i.e., 2010-2020, the maximum number of movies have been released. We can also see that the graph has been on the increase over the years.

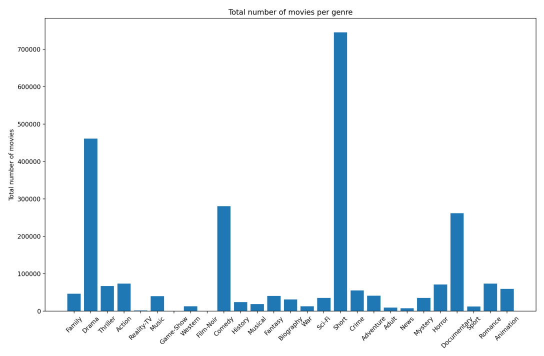

Figure 7: Bar graph - Movies per genre

From the graph shown in Figure 7, we can see that the short film genre has the maximum number of movies followed by the Drama genre. Comedy movies and documentary movies are both a close third in this list. We can also see that movies of Reality-Tv and Game Show genres are minimal.

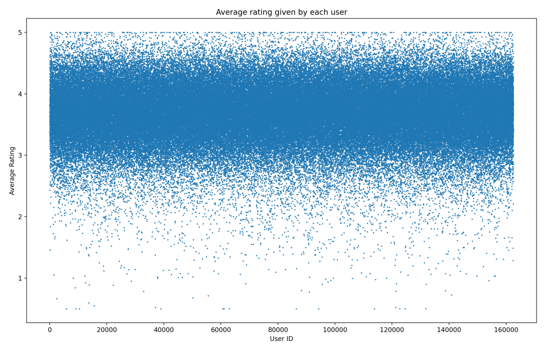

Figure 8: Scatter Plot - Average Rating per user

From the graph in Figure 8, we can see that majority of the users seem to rate movies between 3-4.5. We can also see that there are some users that do tend to give movies a higher rating. Similarly, there are very few users who give movies low ratings. These statistics can be taken into consideration before calculating the overall rating of a movie.

7. PAIRWISE COMPARISON

7.1 Start year versus Runtime minutes

The attribute runtimeMinutes in Movie collection represents the length of the movie and the attribute startYear represents the year during which the movie was released. The following is a scatter plot which shows the relationship between the attributes startYear and runtimeMinutes for all movies from the Movie collection.

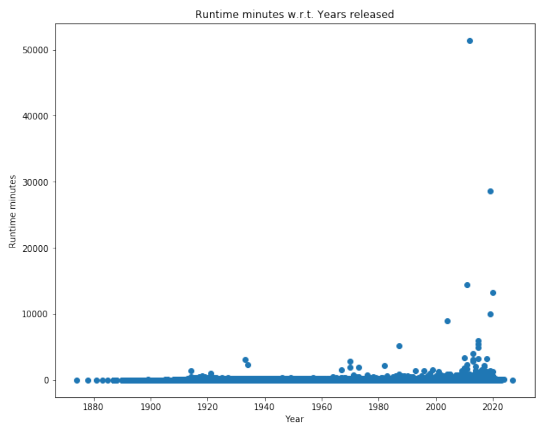

Figure 9: Scatter Plot - Start year versus Runtime minutes

The above plot denotes that there are several movies acting as outliers as their runtime minutes values extremely high. Outliers can cause severe loss in the machine learning modeling performance. It is also important to note that there are about 1.5 million records in the Movie collection which makes the scatter plot cluttered and difficult to understand. In order to be able to better visualize the relationship between the two attributes, 1000 samples from the Movie collection are sampled at random and then plotted as below:

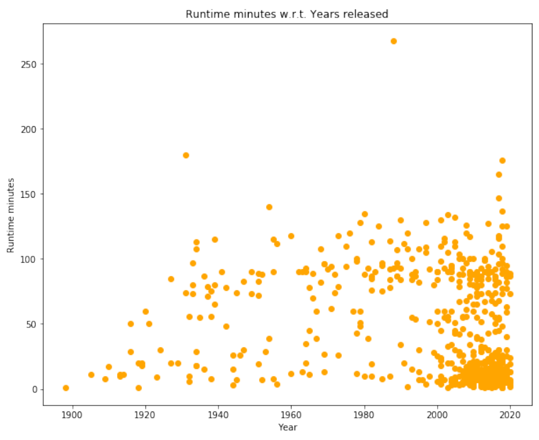

Figure 10: Scatter Plot - Start year versus Runtime minutes for 1000 samples

It is noticeable from the above scatter plot that there are a lot of movie records released between the years 2000 and 2020. Most of the movies released between those years are under 50 minutes of runtime.

7.2 Birthyear versus Deathyear

The attributes birthYear and deathYear in Person collection can be compared and the relationship between them can be studied.

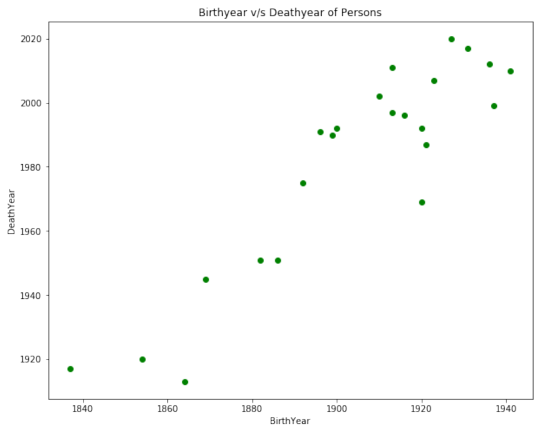

Figure 11: Scatter Plot - Birth year versus Death year

The above scatter plot is created by sampling 1000 samples at random from the Person collection in the database. One can notice that the scatter plot contains fewer records than 1000, which may be because of either one of the two attributes can have a NaN value. One can notice that there is a linear relationship between the attributes birthyear as well as deathyear.

8. CONCLUSION AND FUTURE WORK

In this project, we performed clustering on the normalized average ratings of each movie. We also performed frequent itemset mining on actors and the movies they acted in. We also used Bar Graph, Scatter Plot and Time Series graphs to visualize the data and mine information. We observed that the number of movies released per year increase every year. Furthermore, we can also see that short films genre consists of the most number of movies.

In future, we would be looking into some other queries for visualization. We would also be looking at other movie and ratings based datasets that can be combined with the datasets that we have used - IMDb and MovieLens for extracting further information. We can also experiment by merging the current dataset with another dataset with textual reviews and then perform sentiment analysis using text mining.

9. REFERENCES

[1] Elbow method. `https://en.wikipedia.org/wiki/Elbow_method_(clustering)`.

[2] IMDb dataset. `https://www.imdb.com/interfaces/`.

[3] K-means clustering. `https://en.wikipedia.org/wiki/K-means_clustering`.

[4] Matplotlib. `https://matplotlib.org`.

[5] PyMongo. `https://pymongo.readthedocs.io/en/stable/`.

[6] R. Agarwal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, pages 487–499, 1994.

[7] F. M. Harper and J. A. Konstan. The MovieLens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), Dec. 2015.