IMDb Reviews – Sentiment, Temporal, and Correlational Analysis

Rahul Rajesh Purswani
Department of EECS
University of Kansas
Lawrence, USA
rahulpurswani99@ku.edu

Joe Rubalcava
Department of EECS
University of Kansas
Lawrence, USA
rjrubalcava@ku.edu

Vijay Verma
Department of EECS
University of Kansas
Lawrence, USA
vijayverma@ku.edu

I. Data Extraction – Web Scraping

We used selenium and chromium web driver to scrap through the IMDb pages and collect movie and review data. To get a good balance in positive and negative reviews, we chose to scrap through top 100 movies from two IMDb charts – Top 250 movies on IMDb and Lowest rated movies on IMDb. We started crawling with these IMDb charts to extract movie titles, movie release year, and IMDb movie link. We then crawled through all the movie links to extract movie genre(s) and IMDb reviews link for the movie. Lastly, we crawled through IMDb review links for the movies and extracted review user, review date, review rating, review title, and review content. We did this process for 200 movies, giving us a raw dataset of 8970 rows and 8 columns. To extract data from the webpages, we searched through the HTML DOM tree to find the data (to be extracted) and then filtered it to meet the requirements. The most challenging part of this process was to choose elements and filter the data extracted in such a way that the process can be automated for 200 movies. Figure 1 below shows an example of such HTML DOM tree that we scraped through.

Figure 1: HTML DOM tree of IMDB movie page for one of the movies.

We used the same process to extract data for temporal analysis, however, instead of going through 200 movies, we only scraped through 3 movies to show the temporal analysis – Interstellar, Lord of the Rings, and The Dark Knight Rises. We extracted approximately 300 user reviews for each of these movies, giving us a total of about 950 rows.

II. Data Pre-processing

For dataset pre-processing, we deleted the missing rows and index column. We corrected the format of all the columns to suitable data types. Lastly, we created a new column that shows the sentiment of the review based on the user review rating as positive (Ratings range 8 – 10), negative (Ratings range 1 – 3), and neutral (Ratings range 4 – 7).

For text pre-procesing, we used regex to remove punctuation (like exclamtion marks, inverted commas, etc.), symbols (like $, =, +), and html tags (
, <p>, etc.). After extracting the user reviews for movies, we noticed that at the end of some reviews there some dead text - "Was this review helpful? Sign in to vote.", we used regex to delete this from all the user reviews. We then lowercased all the user reviews and removed stop words. We used stop words from nltk corpus and added some more words to the set relevant to movies context such as film, one, etc. Finally, we stemmed all the reviews using Porter Stemmer from nltk library. The final dataset consisted of 8970 rows and 9 columns. Figure 2 below shows the details of final pre-processed dataset to be used further for analysis and modelling.

```
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Unnamed: 0          8970 non-null   int64
 1   Movie_Title         8970 non-null   string
 2   Movie_Release_Year  8970 non-null   int64
 3   Movie_Genres        8970 non-null   string
 4   Review_User         8970 non-null   string
 5   Review_Date         8970 non-null   datetime64[ns]
 6   Review_Rating       8970 non-null   int64
 7   Review_Title        8970 non-null   string
 8   Review_Content      8970 non-null   string
 9   Review_Sentiment    8970 non-null   string
dtypes: datetime64[ns](1), int64(3), string(6)
```

Figure 2: Pre-processed dataset for Sentiment, Temporal and Correlational Analysis

III. Sentiment Analysis and Model

A. Sentiment Analysis

As mentioned earlier, we are using user review ratings to classify the extracted review as positive, negative, or neutral. We first grouped the data based on the sentiment of the reviews and created word clouds to understand the appearances of words in movie reviews of different sentiments. The bigger and bolder words on the wordcloud indicate a higher number of appreances of that word in user reviews of a particular sentiment, and vice versa. Figure 3 – 6 show word clouds for positve, negative, and neutral reviews. They contain only the top 50 most appreared words for the specific sentiment. It is imporant to note that there are some words such as “good” and “bad” that appear very frequently in postive, negative and neutral reviews; this is due to the very generic context these words can be used in. Another observation was that there are also some words that appear very frequently in word clouds for all three sentiments but they don’t necessarily contribute a lot to the building of the sentiment. For example, the word “make” appears in all three figures 3 – 6, however, “make” itself doesn’t have a meaning that can be classified as positive, negative, or neutral. These words are still important due to the context they can be used in (in the user reviews).

Figure 3: Word cloud for Positive reviews.

Figure 4: Word cloud for negative reviews.

Figure 6: Wordcloud for neutral reviews.

B. Sentiment Prediction Model

We built a sentiment prediction model that predicts the sentiment of a given review. This model was solely trained on the preprocessed text of user reviews. The first step in building the model was to transform review text into tf-idf vectors. These vectors were then divided into training and testing data with 80/20 train-test split. Lastly, Logistic Regression model was applied on the training data and predicted the sentiment of testing data. As shown in figure 7, the model performed very well on the negative and positive reviews, with a very high precision and recall. However, for neutral reviews, the model had a lower precision and recall values. This can be explained with the graph in figure 8. As shown in figure 8, there are approximately 5000 positive reviews, 3000 negative reviews, and less than 1000 neutral reviews. This create a data imbalance, due to which the model has a very low precision and recall values for neutral sentiments. Overall, the model performed with an accuracy fo 84%.

	precision	recall	f1-score	support
0	0.81	0.88	0.85	622
1	0.61	0.11	0.19	182
2	0.86	0.95	0.90	990
accuracy			0.84	1794
macro avg	0.76	0.65	0.65	1794
weighted avg	0.82	0.84	0.81	1794

Figure 7: Sentiment classification report with Logistic Regresion Model.

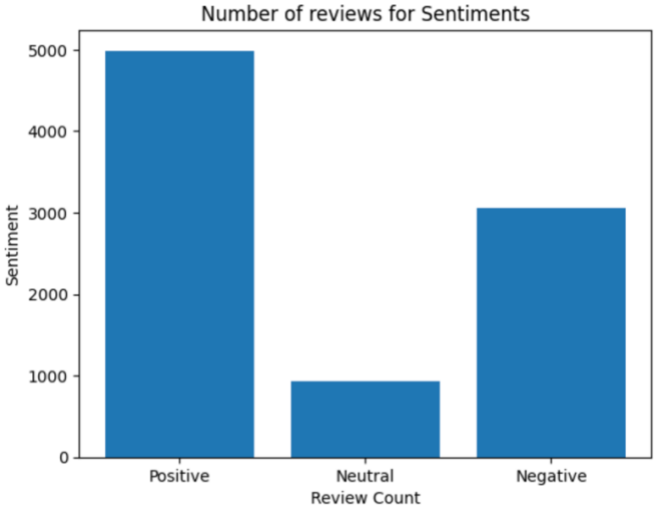


Figure 8: Number of reviews for different sentiments.

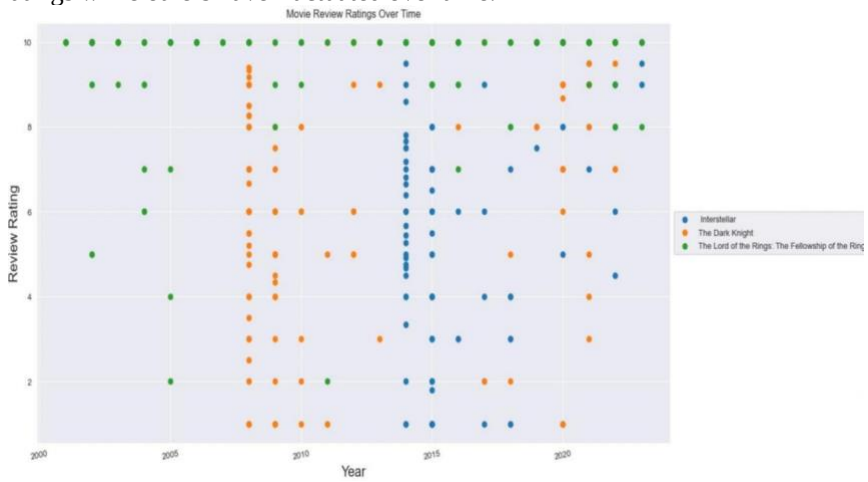
IV. Temporal Analysis

In temporal analysis, we examine the temporal features of the movie data and look into how different elements change over time. We examined specific changes to our movie data over time. We concentrated on five important areas:

- How movie ratings changed over time.
- The average rating users gave at different times.
- How many reviews each movie got every year.
- The average feeling or mood of reviews in different seasons.
- Review sentiment counts for each movie.

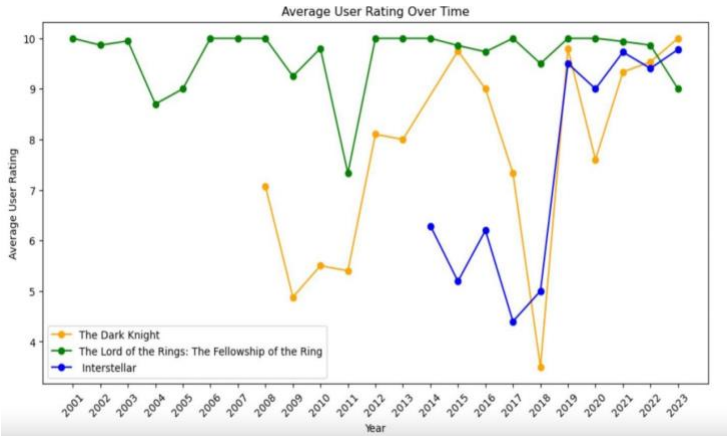
A. Movies Review Rating over time

The graph shows how the review ratings of different movies have changed over time. code is written in Python and utilizing libraries such as pandas, matplotlib, and seaborn, this temporal analysis provides a granular perspective, considering every unique review date for each movie. code reads the IMDB movies data from an Excel file, processes it to ensure correct data types, and then groups it by movie title and individual review dates. For each unique date and movie, the average of the review ratings is calculated. The data is then reshaped to associate each movie with its own column of average review ratings, and a scatter plot is produced to visually represent this data. This makes a detailed visualization that shows how ratings for each movie change over time. Each point shows the average review rate for a unique date. When we analyze the graph representation, it can be observed that Lord of the Rings movie has consistently received high ratings while others have fluctuated over time.



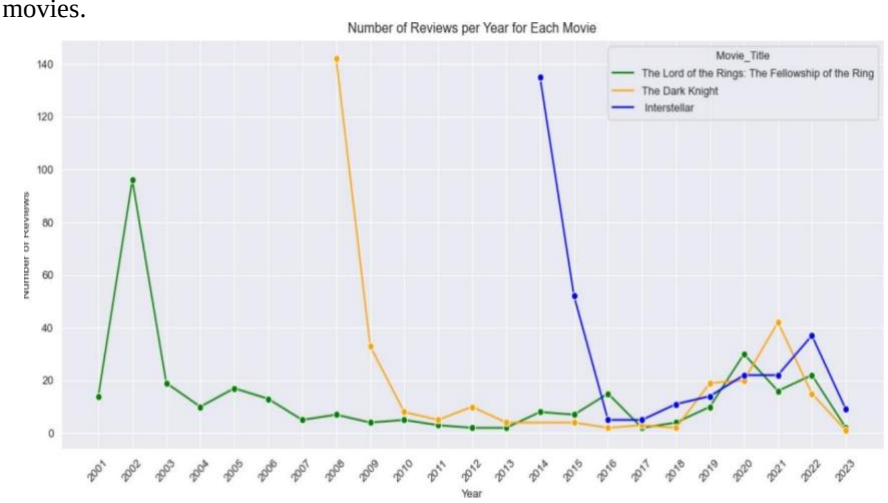
B. Average User Rating over time

This Python script gives a high-level overview of the average user review ratings for movies over time, based on an IMDB dataset. The script starts by reading the data from an Excel file and appropriately converting the review ratings and dates to suitable formats. The core part of the script involves grouping the data by movie title and review date, but this time, the review dates are resampled on a yearly basis. This allows the script to calculate the average review rating for each year for each movie. The reshaped data then forms a table where each column corresponds to a movie, and the rows represent years, with the values being the average review rating for that year. When we analyze the graph representation, The Dark Knight and The Lord of the Rings have both maintained average ratings which are regularly higher compared to that Interstellar.



C. The number of reviews per year for each movie

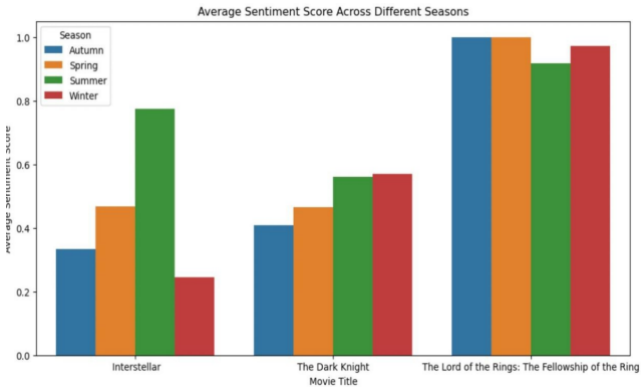
Python script analyzes the number of movie reviews over time, after reading the data from an Excel file, it processes the review dates and groups the data by movie title and year. Code calculates the count of reviews for each movie per year, which represents the number of reviews each movie received within a specific year. The data is then visualized using a line plot, where the x-axis represents years, and the y-axis shows the number of reviews. Each line on the plot corresponds to a different movie, and the points along each line indicate the number of reviews the movie received in a given year. The plot indicates that the number of reviews for each movie has been increasing over time, with some fluctuations, and The Dark Knight has consistently had the highest number of reviews among the three movies.



D. Average sentimental score across different seasons

This graph illustrates how users' average movie opinions changed over time. We analyzed four seasons to determine which season people preferred the film the most compared to the others, Python script is designed to analyze and visualize the average sentiment scores of movie reviews across different seasons. The sentiment scores are calculated based on review sentiments for each movie, which is categorized as 'Positive', 'Neutral', or 'Negative'. The script begins by reading the review data from an Excel file. It then processes the review dates and creates new columns for year, month, and season. The season for each review is determined by a custom function that assigns 'Spring', 'Summer', 'Autumn', or 'Winter' based on the month of the review, Next, the script assigns a sentiment score to each review: 'Positive' reviews are assigned a score of 1, 'Neutral' reviews a score of 0, and 'Negative' reviews a score of -1. The data is then grouped by movie title and season, and the average sentiment score is calculated for each group. This gives an overview of how the sentiment towards each movie changes with the seasons. The following are the statistics:

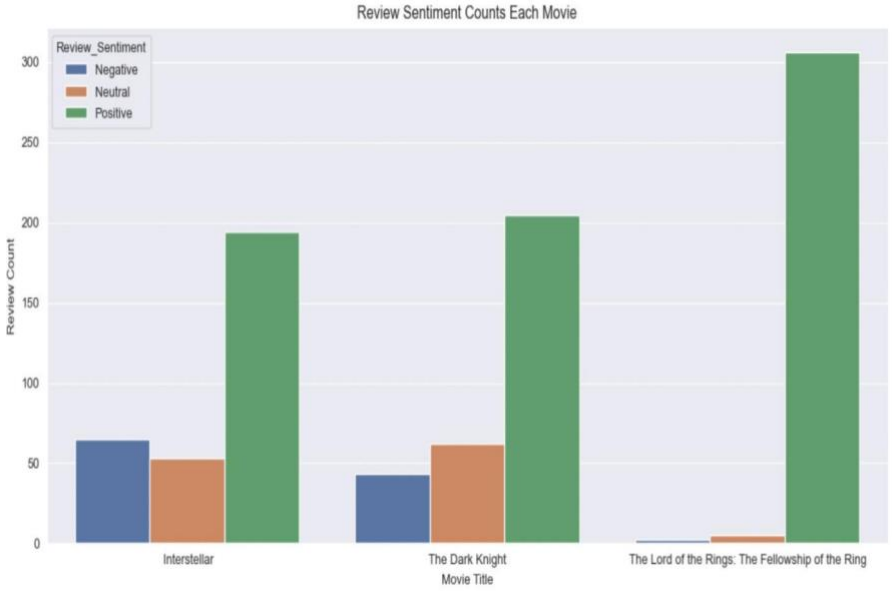
- Interstellar: The film received a higher average sentimental rating during the summer, indicating that people enjoy it more during the summer than during other seasons.
- The Dark Night: In comparison to other seasons, The Dark Night received the highest average sentimental score in the summer and winter.
- The Lord of the Rings: The graph demonstrates that The Lord of the Rings is a distinct winner and that the majority of people enjoy it in each season.



E. Review sentiment counts for each movie

Python code written for this analysis performs analysis to understand the distribution of movie review sentiments across different seasons for selected movies. It starts by loading an Excel file that contains movie review data and separates the date of each review into year and month. It then filters out reviews for three specific movies: 'The Dark Knight', 'Interstellar', and 'The Lord of the Rings: The Fellowship of the Ring'. It then groups the reviews by movie title, season, and review sentiment and counts the number of reviews in each group. Finally, it creates a bar chart to visualize the total number of reviews each movie received in each season.

The Lord of the Rings was given an outstandingly sentimental review, along with a majority of sentimental reviews that were positive. However, the other two movies have also received more positive sentimental reviews compared to negative and neutral ones.



V. Correlational Analysis

In correlational analysis, we examine if a correlation exists between distinct genres. If such a correlation exists, we are interested in how those genres are correlated. To look for a correlation, we cannot examine all reviews, but rather only the reviews penned by frequent reviewers. For this analysis, we consider a frequent reviewer to be anyone who has written over five reviews in the dataset from IMDb. Code for the correlational analysis was written in Python utilizing Panda's data-frames to work with csv files, and matplotlib as well as Numpy to plot information. From the initial review list of ~8,900 IMDb reviews, we get a list of 161 frequent reviewers, as well as 2500+ reviews made by those frequent users.

A. Initial User Extraction

We began by parsing the initial data set of around 8900 reviews to look for only the users who have penned five or more reviews. This was done by performing SQL style commands on the dataset in Google Sheets. Using these commands, we were able to select only the frequent reviewers. This list was then exported as a .csv file, to be examined in the code.

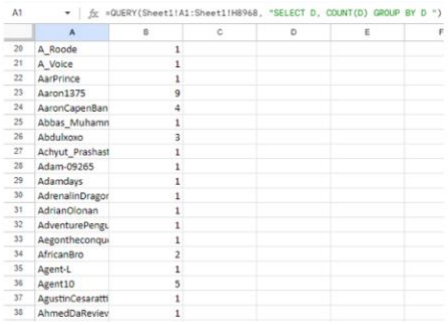


Figure 9 The Query executed to parse the reviews and the users with count.

B. Review and Genre Parsing

Once the list of reviews as well as the list of frequent reviewers were imported as csv files, the python script then looks through all 8900+ reviews and copies reviews made by the frequent reviewers. This leaves a set of ~2600 reviews to analyze. The python script then looks through these reviews and gets a list of all the genres reviewed. This provides a list of 21 genres which we can then look for correlations between.

C. Correlating Reviews

With this new shortened review set of only frequent users reviews, we can now examine correlations between genres. Because there are 21 different genres reviewed, there are 21 choose 2, or 210 graphs which can be viewed. To not show 210 graphs at once, the python script takes in two integers associated with each genre, and outputs a graph displaying the correlation.

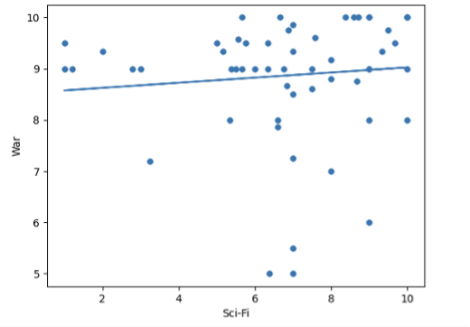


Figure 10 Correlation Graph between War Movies and Sci-Fi Movies

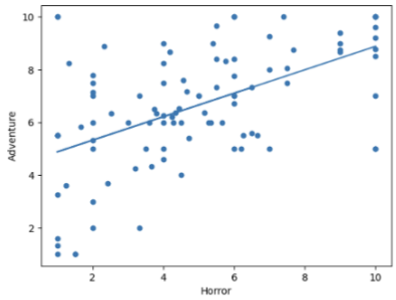


Figure 11 Correlation Graph between Horror Movies and Adventure Movies

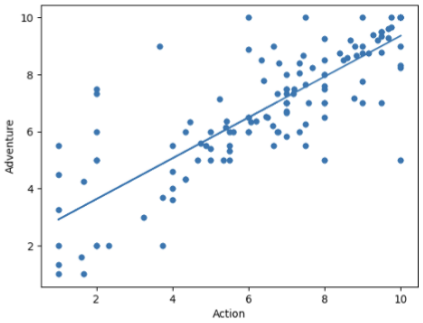


Figure 12 Correlation Graph between Adventure Movies and Action Movies

In Figure 10, we see the correlation between reviews of War movies and Sci-Fi movies. On each graph, each point represents one user's average rating of each genre. These points are also weighted, such that a user who rating many movies in each genre, matters more than a user who has only rated a few movies in each genre. Looking at this figure, we can see the line of best fit is mostly horizontal. Indicating that there does not appear to be much correlation between reviews of Sci-Fi and War Movies.

In Figure 11, we see the correlation of ratings between Horror and Adventure movies. Here we can see that there is a mostly positive correlation between the two genres. This can be seen in that the line of best fit has a steep positive slope. This indicates that the more you like horror movies, the more likely you are to also like adventure movies.

In Figure 12, we see the most positive correlation yet. This shows that across a wide range on review scores, most users' ratings towards action movies are positively correlated towards their ratings towards adventure movies. In addition, it means that if you do not like action movies, it is highly unlikely you will like adventure movies. Across these three graphs, we can see a wide range of how reviews are or are not correlated between genres.

VI. Contributions Made by Each Team Member

- Data Extraction, Data Pre-processing, Sentiment Analysis and Model – Rahul Purswani
- Temporal Analysis – Vijay Verma
- Correlational Analysis – Joe Rubalcava

VII. References

Pham, H. (2021, April 13). A beginner guide for scraping data from IMDB for user reviews using selenium and beautifulsoup. Medium. https://hungpham89.medium.com/a-beginner-guide-for-scraping-data-from-imdb-for-user-reviews-using-selenium-and-beautifulsoup-c60e89a4ad1a

Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.

Mueller, A. (2020). Wordcloud.wordcloud. wordcloud.WordCloud - wordcloud 1.8.1 documentation. http://amueller.github.io/word_cloud/generated/wordcloud.WordCloud.html

P. D. (2023, May 12). GitHub - pandas-dev/pandas: Flexible and powerful data analysis / manipulation library for Python, providing labeled data structures similar to R data.frame objects, statistical functions, and much more. GitHub. https://github.com/pandas-dev/pandas

M. (2023, May 12). GitHub - matplotlib/matplotlib: matplotlib: plotting with Python. GitHub. https://github.com/matplotlib/matplotlib

M. (2023, May 9). GitHub - mwaskom/seaborn: Statistical data visualization in Python. GitHub. https://github.com/mwaskom/seaborn

Analyzing IMDB Movie Dataset. (n.d.). Analyzing IMDB Movie Dataset. https://www.linkedin.com/pulse/analyzing-imdb-movie-dataset-preetish-panda

P. D. (2023, May 12). GitHub - pandas-dev/pandas: Flexible and powerful data analysis / manipulation library for Python, providing labeled data structures similar to R data.frame objects, statistical functions, and much more. GitHub. https://github.com/pandas-dev/pandas

M. (2023, May 12). GitHub - matplotlib/matplotlib: matplotlib: plotting with Python. GitHub. https://github.com/matplotlib/matplotlib

NumPy. GitHub. (2023, May 12). GitHub – numpy: The fundamental package for scientific computing with Python. Github. https://github.com/numpy