CS6120 PROJECT - MOVIE RECOMMENDATION

Group 24:
Jia Xu, Xuan Zhang, Yi Chen
Meishan Li, Qia Lin
Northeastern University
Boston, MA 02115

ABSTRACT

In this article, we present a content-based recommendation system that utilizes bag-of-words (BoW) to generate recommendations for users. The system calculates the cosine similarity between the user and movie files, with the goal of recommending 10 movies to each user. To improve efficiency, we use K-means clustering to classify the large number of movies in the movie data. The system has a low root mean squared error (RMSE) of less than 0.14 and a low R-squared value of less than 0.10, on average, for 100 users.

1 Introduction

As the number of movies available to watch online grows, it can be overwhelming for people to choose which ones to watch. To help people find movies that suit their interests, we are developing a movie recommendation tool using unsupervised learning to group movies and cosine similarity to more effectively select movies that users will enjoy. By providing personalized movie recommendations, this tool aims to make it easier for people to find the films they want to watch.

1.1 TMDB 5000 Movies dataset

We are using the TMDB 5000 film dataset from Kaggle, which includes two datasets: $tmdb\_5000\_movies$ and $tmdb\_5000\_credits$. The $tmdb\_5000\_movies$ dataset contains information on 4083 movies, such as $budget$, $genre$, $keywords$, $overview$, $popularity$ and $votingaverages$. The $tmdb\_5000\_credits$ dataset includes $movie\_id$, $title$, $cast$, and $crew$. We will use the features from these two datasets to create a recommendation system.

1.2 BoW

The Bag-of-Words(BoW) model, which converts a sentence into a vector representation, is a relatively straightforward approach that does not consider the order of words in a sentence, but only the number of occurrences of words in the vocabulary in that sentence.

1.3 K-Means

The k-means clustering algorithm is an iterative clustering analysis algorithm in which the data is divided into K groups, then K objects are randomly selected as the initial cluster centroids, and the distance between each object and each seed cluster centroids is calculated, and each object is assigned to the cluster centroid nearest to it. The clustering centroids and the objects assigned to them then represent a cluster. Each time a sample is assigned, the cluster centroids are recalculated based on the existing objects in the cluster. This process is repeated until a termination condition is met. The termination conditions can be that no (or a minimum number of) objects are reassigned to different clusters, no (or a minimum number of) cluster centroids change again, and the error sum of squares is locally minimised.

2 Method

2.1 Data Pre-processing and Analysis

To begin our analysis, we extracted data from the TMDB 5000 Movies dataset and created a heat map for observational purposes. Next, we pre-processed the movie descriptions and selected appropriate data features for labeling. We then presented data on the top genres, actors, and directors, and created a WordCloud plot of keywords for further observation and analysis. Finally, we sorted the movies by vote rating and popularity.

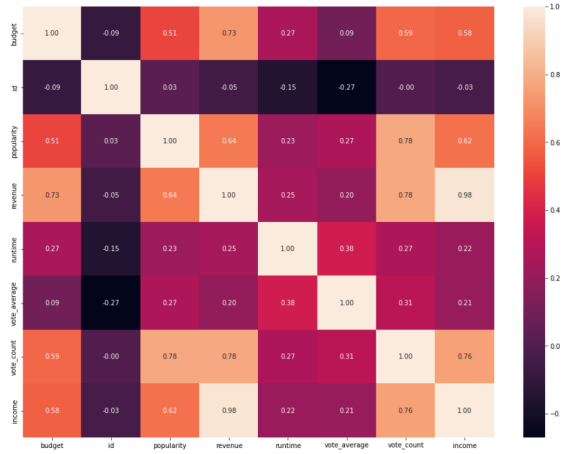

Figure 1: Heat Map

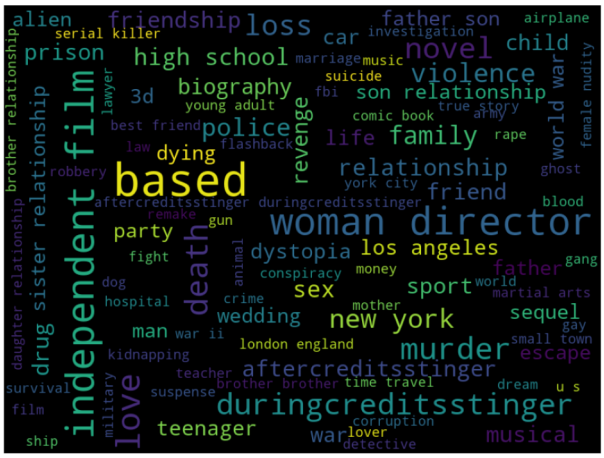

Figure 2: WordCloud

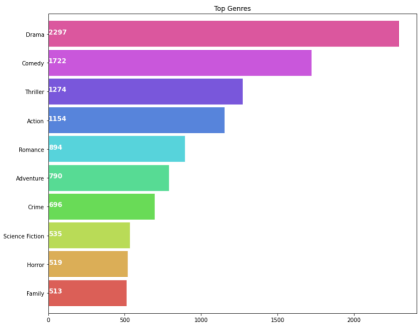

Figure 3: Top Genres

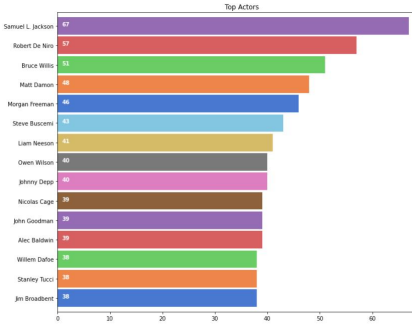

Figure 4: Top Actors

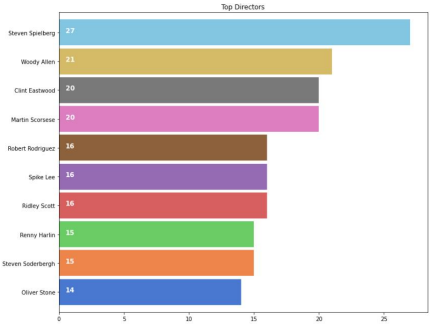

Figure 5: Top Directors

2.2 Data Clean and Vectorize the features

Then we cleaned the data and vectorized the chosen tokens from the data(genres, casts, directors, keywords, descriptions by using the BoW method.

The following are examples for BoW Vectors.

```
0    [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
1    [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
2    [1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
3    [1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, ...
4    [1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
Name: genres_bin, dtype: object
```

Figure 6: BoW Vector for Genres

```
0    [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
1    [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, ...
2    [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, ...
3    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, ...
4    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
Name: cast_bin, dtype: object
```

Figure 7: BoW Vector for Cast

2.3 Cluster Movies

To improve efficiency with large data sets, we are using the k-means clustering algorithm to group data based on BoW features. This will allow us to quickly and easily to do similarity calculation.

$$J(c,\mu) = \sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$$

Figure 8: K-mean

Our elbow was 5, which was the optimal parameter we found after testing. As shown in Figure 9, there is no point in adding more clusters.

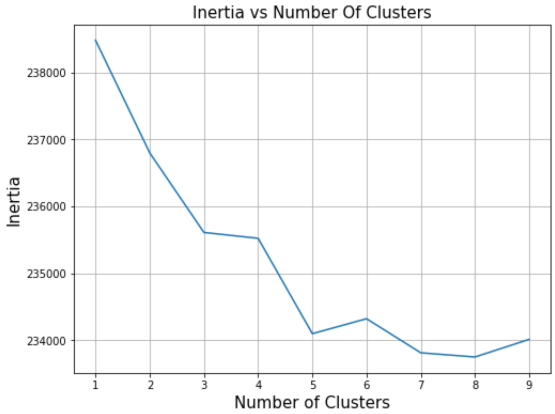

Figure 9: Inertia vs Number of Clusters

Here are the 5 clusters:

```
CLUSTER  1
Popular Movies: ['Whiplash', 'Fight Club', 'Fury', "One Flew Over the Cuc
koo's Nest", 'The Godfather: Part II', 'The Green Mile', 'Cinderella', "W
e're the Millers", 'The Twilight Saga: Breaking Dawn - Part 2', 'The Wolf
of Wall Street']

CLUSTER  2
Popular Movies: ['Mad Max: Fury Road', 'Dawn of the Planet of the Apes',
'The Hunger Games: Mockingjay - Part 1', 'Terminator Genisys', 'The Dark
Knight', 'Inception', 'Gone Girl', 'Rise of the Planet of the Apes', 'The
Maze Runner', 'Pulp Fiction']

CLUSTER  3
Popular Movies: ['Minions', 'Interstellar', 'Deadpool', 'Guardians of the
Galaxy', 'Jurassic World', 'Pirates of the Caribbean: The Curse of the Bl
ack Pearl', 'Big Hero 6', 'Captain America: Civil War', 'The Martian', 'B
atman v Superman: Dawn of Justice']

CLUSTER  4
Popular Movies: ['Frozen', 'Forrest Gump', 'Twilight', 'Bruce Almighty',
'The Twilight Saga: Eclipse', 'The Twilight Saga: New Moon', 'The Age of
Adaline', 'The Fault in Our Stars', 'Amélie', 'Sex Tape']

CLUSTER  5
Popular Movies: ['The Imitation Game', 'The Godfather', 'The Shawshank Re
demption', 'Inside Out', "Schindler's List", 'Titanic', 'Fifty Shades of
Grey', '12 Years a Slave', 'Blade Runner', 'Psycho']
```

Figure 10: Clusters

2.4 Generate Recommendations

Finally, We have done the following steps to implement movie recommendations:

1. Computing the cosine distance
   We use cosine similarity to measure the magnitude of the vector angle between two user vectors: the smaller the angle, the greater the cosine similarity, and the more similar the two users.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 11: Cosine Similarity formula

2. Generate recommendations and Predict score
   Choose the 10 films with the smallest cosine distance from all the films in the same cluster as recommendations.

   And here is the result:

```
Selected Movie:  Catch Me If You Can

Recommended Movies:

Saving Private Ryan | Genres: Drama, History, War | Rating: 7.9
War Horse | Genres: Drama, War | Rating: 7.0
Lincoln | Genres: History, Drama | Rating: 6.7
Close Encounters of the Third Kind | Genres: Science Fiction, Drama | Rating: 7.2
Amistad | Genres: Drama, History, Mystery | Rating: 6.8
Wall Street | Genres: Crime, Drama | Rating: 7.0
The Funeral | Genres: Crime, Drama | Rating: 7.3
American Hustle | Genres: Drama, Crime | Rating: 6.8
The Wolf of Wall Street | Genres: Crime, Drama, Comedy | Rating: 7.9
Capote | Genres: Crime, Drama | Rating: 6.8
```

Figure 12: Movie Recommendation

3 Evaluation

From our recommendation system, we will utilize the Root Mean Square Error (RMSE) method to evaluate our model. We will calculate the average error between the actual 10 movies the person who watched before and the result of 10 recommended movies predicted in our model.

$$\text{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}}$$

Figure 13: RMSE

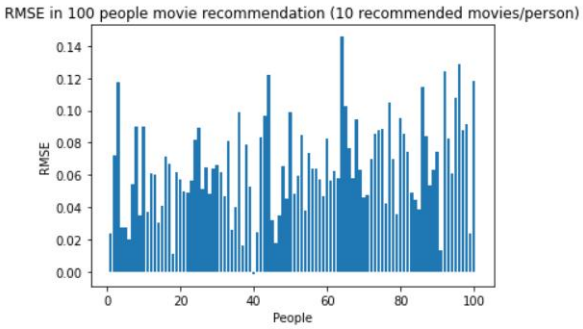

Figure 14: RMSE in each person (differences in 10 recommended movies and 10 watched movies/person)

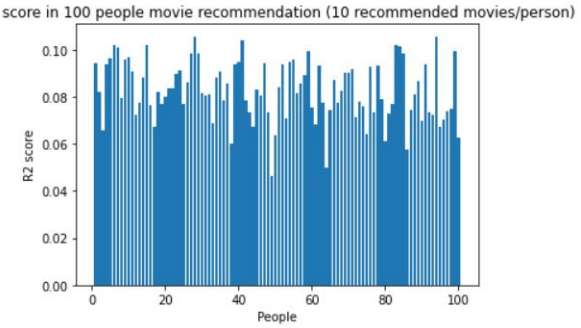

Figure 15: R2 score in each person (differences in 10 recommended movies and 10 watched movies/person)

4 Results

In order to test the accuracy of the system, we randomly selected 100 users to predict 10 films they might like, using the 10 films they had seen as the basis for their recommendations. Then compare the feature vector of the recommendation to the base by RMSE method. The figures from RMSE section easily show that RMSE is below 0.15, and the R2 Score is below 0.1. It indicates that our system provides decent recommendations for users, which are similar to their original preference, in an efficient way.

5 Conclusion

We conducted a thorough analysis of the dataset to identify important features and vectorize them using the bag-of-words model. To improve the efficiency of our recommendation system, we applied K-means clustering to the large dataset to reduce computational complexity and time. When making recommendations, we first predict which cluster the user belongs to and use cosine similarity as a measure to recommend movies. Our model achieved satisfactory results in terms of root mean squared error (RMSE) and R-squared.

6 Future Works

In order to further improve the effectiveness of our recommendation system, we could consider using alternative methods such as term frequency-inverse document frequency (TF-IDF), pointwise mutual information (PMI), and neural word embedding for vectorization and collaborative filtering. Collaborative filtering would be based on the user and item files to fill out any missing data in these files. Additionally, we could potentially extract useful features from the unused movie data and use forward feature selection to identify the most relevant features. These approaches could enhance the performance of our recommendation model.

References

[1] Jure Lekovec, Anand Rajaraman, Jeffrey D.Ullman, 2014. *Mining of Massive datasets.* `http://infolab.stanford.edu/~ullman/mmds/ch9.pdf`

[2] G Geetha et al 2018 J. Phys.: Conf. Ser. 1000 012101 *A Hybrid Approach using Collaborative filtering and Content based Filtering for Recommender System* `https://iopscience.iop.org/article/10.1088/1742-6596/1000/1/012101/pdf`

[3] Bagher Rahimpour Cami; Hamid Hassanpour; Hoda Mashayekhi *A content-based movie recommender system based on temporal user preferences* DOI: `10.1109/ICSPIS.2017.8311601`