CS6120 PROJECT - MOVIE RECOMMENDATION

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

ABSTRACT

This recommender system is based on content. In this article, we use BoW to make an Item(movie) file and create a User file by producing a record randomly to do cosine similarity. In general, the recommender system will calculate the cosine similarity between the User file and Item(movie) file. But, to Movies file data, there are lots of movies that are in this data. Thus, we use K-means to do good classification. This way helps the system to find recommended movies quicker. Finally, each user can get 10 recommended movies. And the average of RMSE from 100 users is not over 0.14. RMSE is very low. In addition, the average R-square from 100 users is not over 0.10. R-square is very low.

1 Introduction

In recent years more and more people are choosing to watch movies on the Internet. The variety and number of movies are also increasing. People often don't know which one to watch instead of facing so many movies. If there is a good recommendation application for movies, these recommendations will help people to choose a movie of the type they want to watch. We will implement a movie recommendation tool by using unsupervised learning models to group movies, and using cosine similarity to select movies the users want to watch more effectively.

1.1 TMDB 5000 Movies dataset

The TMDB 5000 film dataset is a Kaggle dataset. There are two datasets that we use. The first dataset is $tmdb\_5000\_movies$ which contains information on 4083 movies. This information includes $budget$, $genre$, $keywords$, $overview$, $popularity$ and $votingaverages$. The second dataset is $tmdb\_5000\_credits$ which contains $movie\_id$, $title$, $cast$, and $crew$. We will take these two datasets of features of these two datasets to form 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

We first extracted the data from the TMDB 5000 Movies dataset and plotted a heat map for observational analysis. We then performed some pre-processing on the movie descriptions. Some appropriate data features were selected for labeling. Then we presented data on top genres, top actors, and top directors. A WordCloud plot of keywords was also created for better observation and analysis. We also Sort the movies by vote rating and popularity.

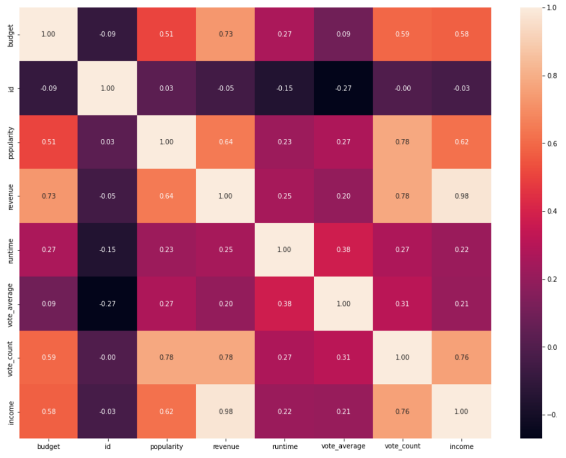


Figure 1: Heat Map

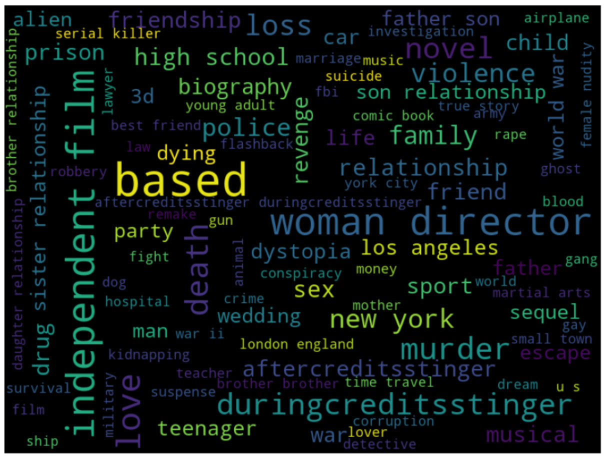


Figure 2: WordCloud

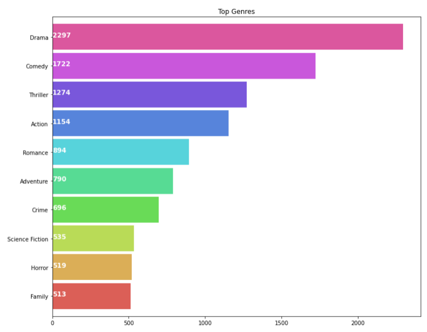


Figure 3: Top Genres

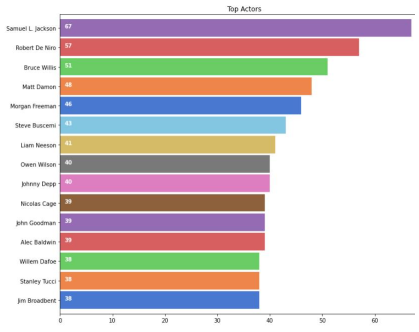


Figure 4: Top Actors

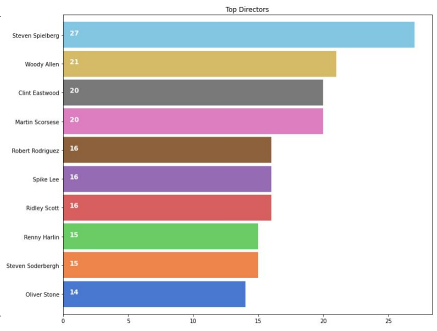
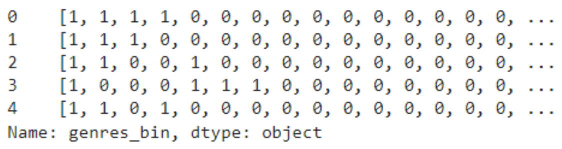


Figure 5: Top Directors

2.2 Data Clean and Vectorize the features

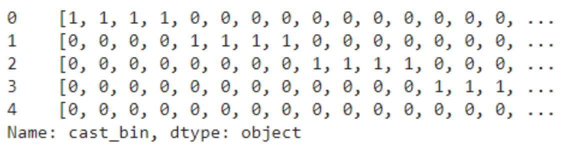
Then we cleaned the data and vectorized the chosen tokens from the data(genres, casts, directors, keywords, descriptions by using the BoW method.

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, 0, ...
1    [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, ...
2    [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, ...
3    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, ...
4    [0, 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

In this step, to big data, we use clustering algorithms (k-mean) to group data based on BoW features for us to find what we want more quickly and easily.

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

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.

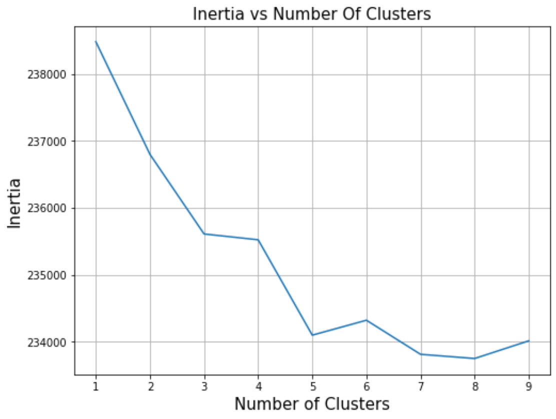


Figure 9: Inertia vs Number of Clusters

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:

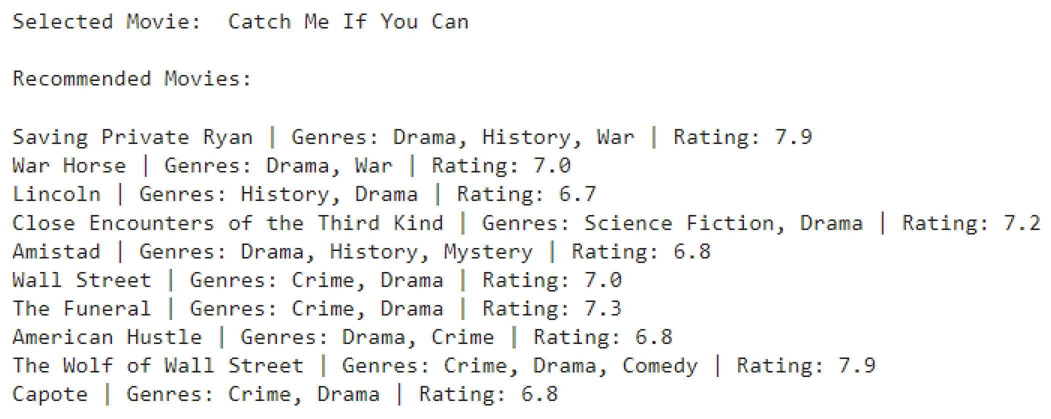
1. Computing the cosine distance
   We use cosine similarity to measure the magnitude of the vector angle between two user vectors: the smaller the angle, the greater the cosine similarity, and the more similar the two users.

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

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.

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

Figure 13: RMSE

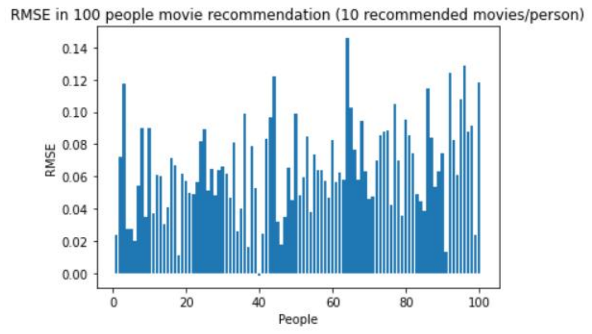


Figure 14: RMSE in each person
(differences in 10 recommended movies and 10 watched movies/person)

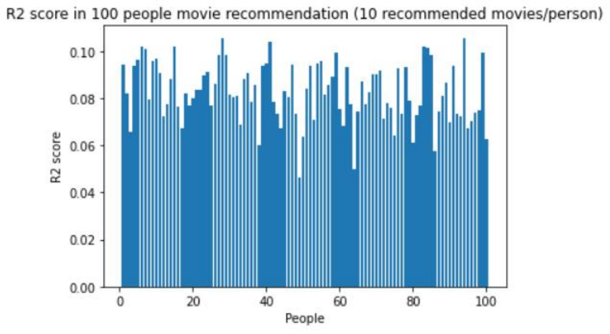


Figure 15: R2 score in each person
(differences in 10 recommended movies and 10 watched movies/person)

4 Results & Conclusion

In order to test the accuracy of the system, we takes 100 random users, who has seen 10 movies as base for recommendation, and predict 10 movies they may love, then compare feature vector of the recommendation to the base by rmse method.The figures from rmse section easily shows that rmse is below 0.15, and R2 Score is below 0.1. It indicates that our system provides decent recommendations for users, which is similar to their original preference, in an efficient way.

We did a good analysis of this dataset to select the important features and vectorize them by using bag of word. For large data, to reduce the computational complexity and computational time, we made a nice clustering with Kmeans. When making recommendations, we will first predict which cluster the referee belongs to, and use cosine as an indicator to recommend movies. Finally, our model achieves good results on RMSE and R-square.

5 Future Works

We can try to use other way (like TFIDF, PMI and Neural word embedding) to do vectorization and collaborative filtering in this project. The collaborative filtering will be based on the User file and Item file to fill out empty of these two files. I think it will make this model of this project better. In addition, there are some parts of the movies data we did not use in this project. It is possible to extract useful features from these movies data we did use them in this project. And we can using forward feature selection to select good features. It will help this model of this project to do a better classification.

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`