Movie Recommendation System based on Content Similarity

1 Summary

Our project is aiming at expanding user's movie archive and find he/she next potential favored movie. The methodology of our project is based on a similarity comparison between movie contents and attributes. In other words, users may get recommendations for movies based on a series of movie tags, such as movie name, reviews, plots, genres, directors, etc.

2 Database

The database is obtained by web scraping IMDb website. It contains 267,581 movies along with 12 attributes - movie name, genre, director, writer, star, keyword, reviews, release country, release date, rating value, and time in csv format.

2.1 Web Scraping

First, we extract all movie genres and their links from IMDb homepage. Inside the first page of every genre, we use recursion function to iterate over all pages and obtain every movie's link. Using movies' link, we go into the homepage of every movie and fetch available information. During web scraping process, we have encountered two main problems described as following and fixed them successfully.

2.1.1 Selenium

When fetching movie reviews, we couldn't get all reviews since there is no url change after clicking the load more button. Then we use selenium package to help us automatically click on load more button and stop clicking by either restricting the amount of reviews we want or until there is no more button to click.

2.1.2 Multiprocessing

Due to the huge amount of movie information we want to get, there is no way that we can finish the web scraping in a week. Eventually. we have come up with using multi processing to solve the issue and successfully build our dataset within 24 hours.

2.2 Cleaning

In the process of getting data from webpage, we use list to contain information of every attribute. The problem is that after writing and reading csv file, data type is transferred to string, which creates an obstacle to our future work. To solve this issue, we convert the dataframe to a nested dictionary and did several adjustments. By doing so, we can read all attributes as lists instead of string.

3 General Analysis

We have done lots of general analysis and visualization work as well. In general analysis part one, we are interested in movie performance of different genres. We found that rating value distribution of most genres are similar to the normal distribution, and among all genres, romance movies are more likely to obtain higher scores[1]. In general analysis part two, we plot pie charts to help users better visualize the proportion of release country under each movie genre[2]. To further provide users with information about different release countries, we also plot bar charts to show distributions of rating values by release countries for each genre[3].

4. Text Mining

4.1 Sentiment Analysis

One of the most important features to evaluate a movie is its reviews. In order to quantify movie reviews for our later analysis in machine learning, we construct sentiment analysis using Vader to assign a "compound" score based on heuristics (between -1 and +1) to reviews for each movie

4.2 Similarity

Since our recommendation system heavily relies on content similarity, we used the LSI model to detect similarity between movies. The corpus contains storyline, keywords, reviews, and genre of each movie. When a new document arrives, the function will return a list sorted by descending similarity associated with that document.

4.3 Simple analysis: Words Cloud

To help users better visualize a movie content before watching it, we combined movie's keywords, genre, reviews and storyline into one string, and generate a word cloud using words in that string. We also set a word length limit and a delete_word list to refine the word cloud. In the appendix, we have plotted word clouds of movie Frozen and Spider-Man.[4]

5. Machine Learning

There are more than 0.2 million movies in our dataset, but half of them don't have rating values. This is not only inconvenient for the audience to tell whether a movie is worth watching or not, but also difficult for us to recommend movies for them. Therefore, we plan to design several machine learning models to predict if a movie is worth watching or not.

5.1 Get preparation for feature value

5.1.1 Binary Attributes

- We set director, genres, keywords, release country, release date, stars, storyline, writers as binary independent variables. If they are null, we set the value as 0, otherwise, set as 1.
- We assign compound scores from sentiment analysis to reviews and normalize it.
- We set rating values which are larger than 7 as 1, otherwise as 0 to be our dependent variables.

However, there are still some problems with this method. There is no way to distinguish the real effect of certain attributes on movies since their values are either 0 or 1. Therefore, we came across a new idea trying to convert all binary independent variables to numerical variables.

5.1.2 Numerical Attributes

In this part, we design to improve our models by filtering features which are important for a movie's rating value and then assign them with numerical values. Taking director attribute as an example, we find all movies of the director and drop out movies without rating value, calculate the average rating and assign this average to the director as his/her numerical value. If all movies of

this director don't have rating values, we set 0 to this director. The same process applies to genres, release country, stars, and writers. Reviews and dependent variable are handled the same way as described in binary attributes.

We think this method is meaningful. Still taking the director attribute as an example, though the average rating value of the directors' other movies can't fully represent the director's level at the time he directed this movie, it can at least indicates his general level. To some extent, it has some reference value. Meanwhile, we never set one feature as the only reference to tell whether a movie is good or not. We usually combine multiple elements together. This will reduce the impact of this simple average value method.

However, our model still has some limitations. In spite of assigning numerical values to those categorical attributes, there are still many missing values that we have assigned them to be 0. This is not very reasonable since missing values don't usually equal to a bad movie and in this way, we may misjudge the value of some movies. There is something we can do to improve our models from this aspect.

5.2 Logistic Regression Model

Before we start to put data into the model, we first calculate the correlation between different features and draw a cool-warm map[5]. The result shows that the relationship between each feature isn't very significant, indicating that our data is suitable for this model. The correlation between genre and other features is always NaN when genre is binary because every movie has genre and the binary value for each genre is always 1. Since we assume genre is very likely to have low correlation with other features, the data is still valid for this model. The result shows accuracy is 0.7331 for binary attributes and 0.7716 for numerical attributes.

5.3 Random Forest Model

Since we want to further improve the accuracy of our model, we try the random forest model. We set different values for each parameter: max_depth, min_samples_leaf, min_samples_split, N_estimator, and the result shows that 6, 8, 8, 50 is the best combination for binary attributes with accuracy 0.7336 and 15, 4, 4, 100 is the best combination for numerical attributes with accuracy 0.9226. From above results, we discover that accuracy using numerical independent variables is significantly higher than that using binary independent variables, which further confirmed our idea that taking average value indeed make sense. In addition, from the graph of feature importance images[6], we realized binary and numerical attributes have very similar importance order, which indicates the consistency between these two methods.

5.4 Neural Network Model

Finally, we try the neural network model. Similar to random forest model, we set different values for each parameter - solver, hidden_layer_sizes, max_iter, activation, learning rate. The optimal results are lbfgs, 80, 1000, relu, constant and lbfgs, 60, 1000, tanh, adaptive for binary and numerical attributes respectively with accuracy both exceed 0.9.

5.5 Comparison

After comparing the results from three models[7], we discover that random forest model and neural network model outperforms logistic regression model. Numerical attributes outperforms binary attributes. We originally expected neural network model is better than random forest model, but their accuracies are similar to each other. We think that's probably because the value of max_iter in neural network isn't large enough.

5.6 Forecast and Update Database

Finally, we use these three models to forecast the quality of movies without rating value. Movies with predicted rating value 1 is worth watching, 0 is not worth watching.

To apply this prediction into our recommendation system, we decided to use random forest model with numerical attributes. We set the rating value as 7.5 for movies with predicted binary rating value 1 and 4.5 for movies with predicted binary value 0. We update the database by adding numerical rating value.

6 Recommendation

6.1 Using movie attributes to get recommendation

Our system supports users to input arbitrary combined movie attributes and return top 10 rating movies. For example, when a user inputs a director name and a star name, our system will show movies with top ten rating value that the director directed and the actor participated in. At the same time, our system can plot graphs of rating value distribution of the director and actor respectively.

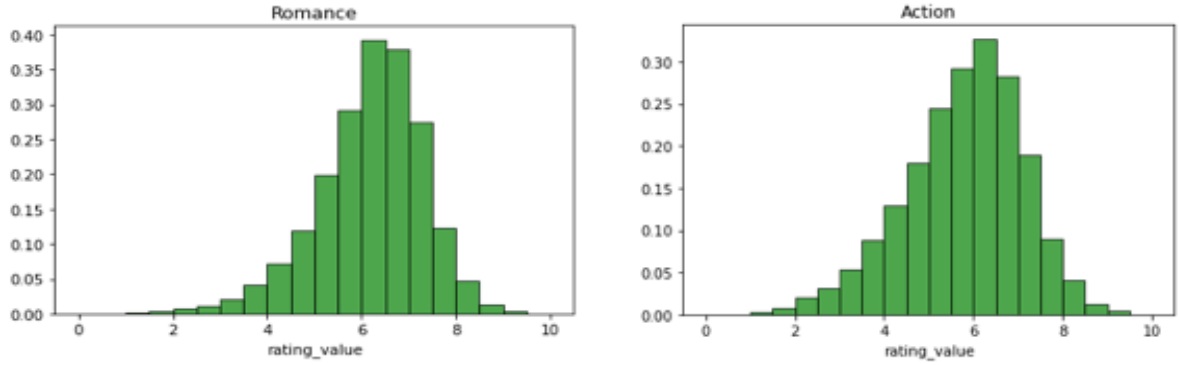
6.2 Using movie name to get recommendation

Our system also support uses to get recommendation solely based on movie names. The system will fetch keywords, reviews, storyline, and genre of the movie from the database, compare it with the corpus we used in LSI model. It returns a movie list along with similarity index. We score movies between 0 - 1 with scoring rule - 85% * similarity index + 5% * rating value + 5%* (director + star + writer). All attributes are numerical attributes. Every Time users input a movie name, our system will output top 10 scored movies as our recommendation.
Here are some recommendation results we obtained:

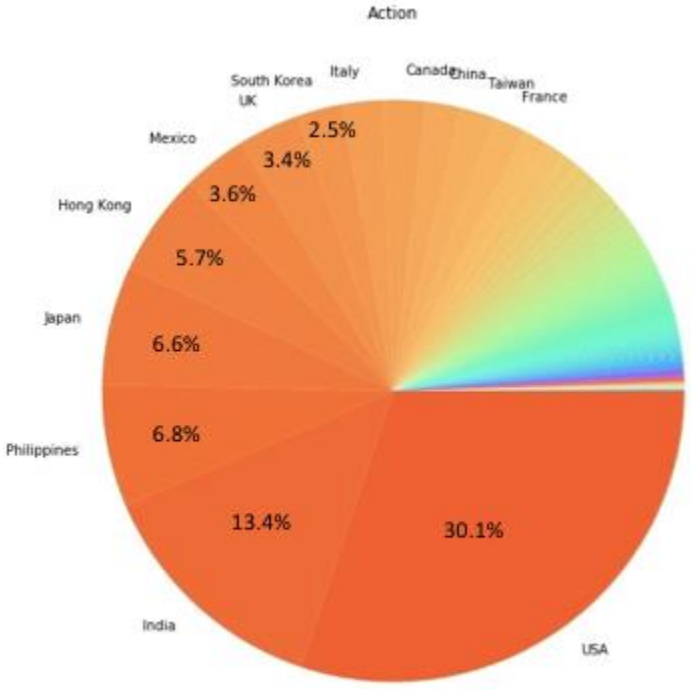
User input	Recommendation (order by score)
Frozen	Maleficent, Tangled, The Swan Princess, Moana, Enchanted, Beauty and the Beast, Aladdin, One Hundred and One Dalmatians
Spider-Man	Spider-Man 2, The Amazing Spider-Man, The Amazing Spider-Man 2, Spider-Man: Into the Spider-Verse, Spider-Man 3, X-Men, Hancock, The Avengers

Appendix

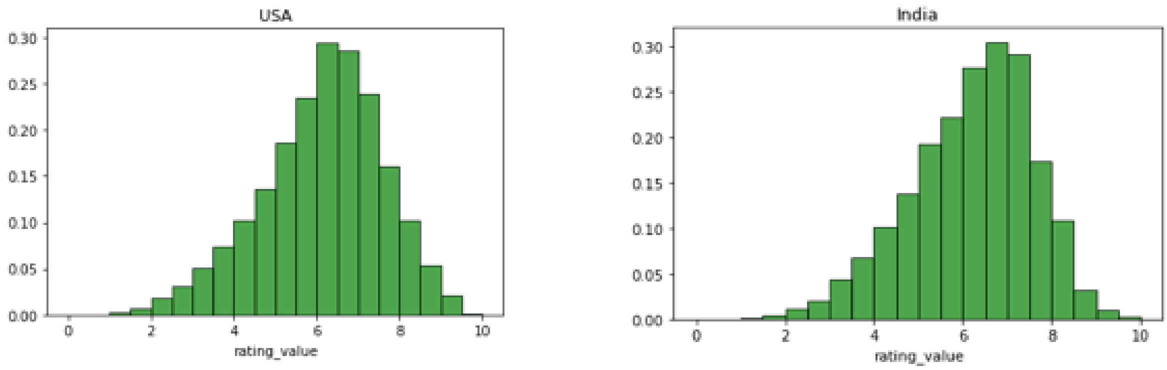
[1]


Romance
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
10
rating_value
Action
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
10
rating_value

[2]


Action
South Korea
Italy
Canada
China
Taiwan
France
UK
Mexico
2.5%
3.4%
3.6%
Hong Kong
5.7%
Japan
6.6%
6.8%
Philippines
13.4%
30.1%
India
USA

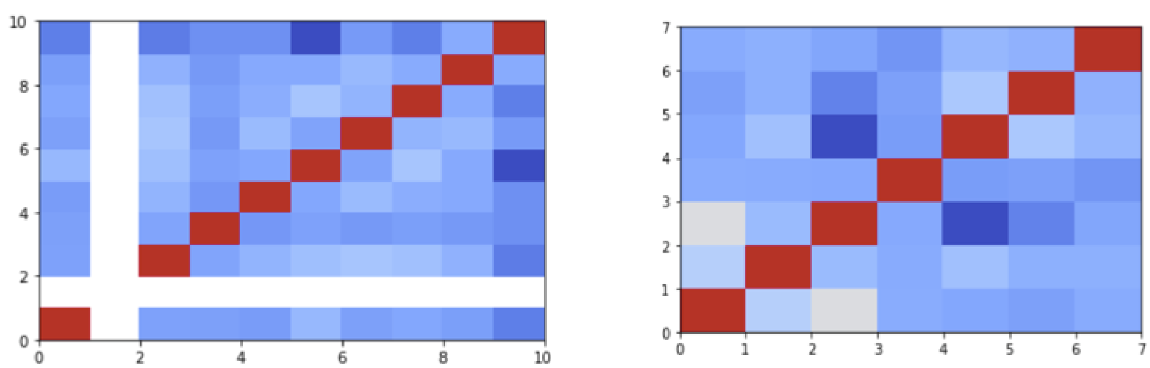
[3]


USA
India
rating_value
rating_value

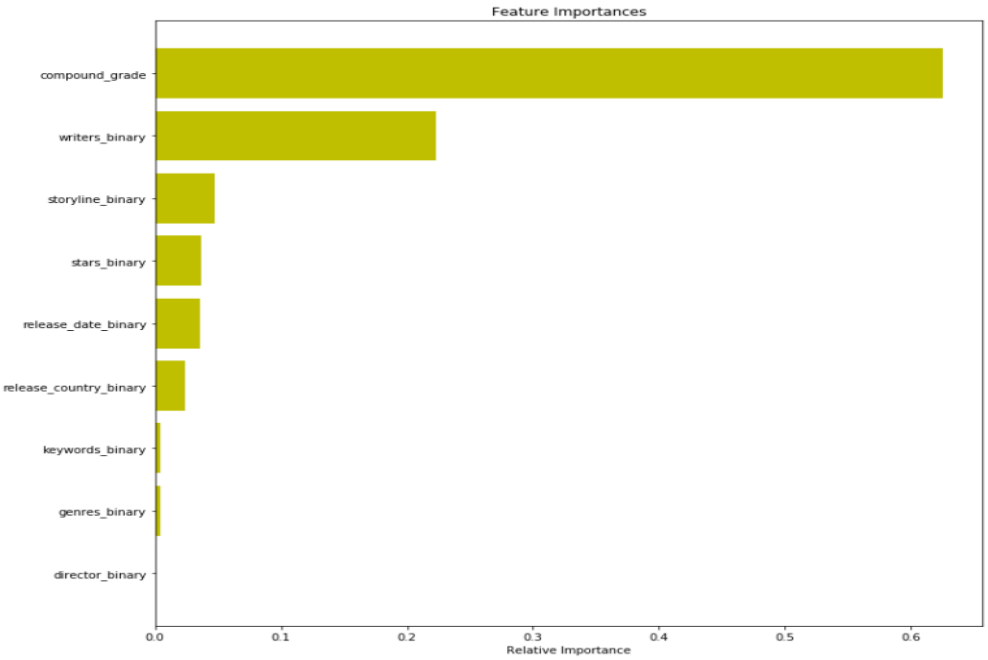
[4]

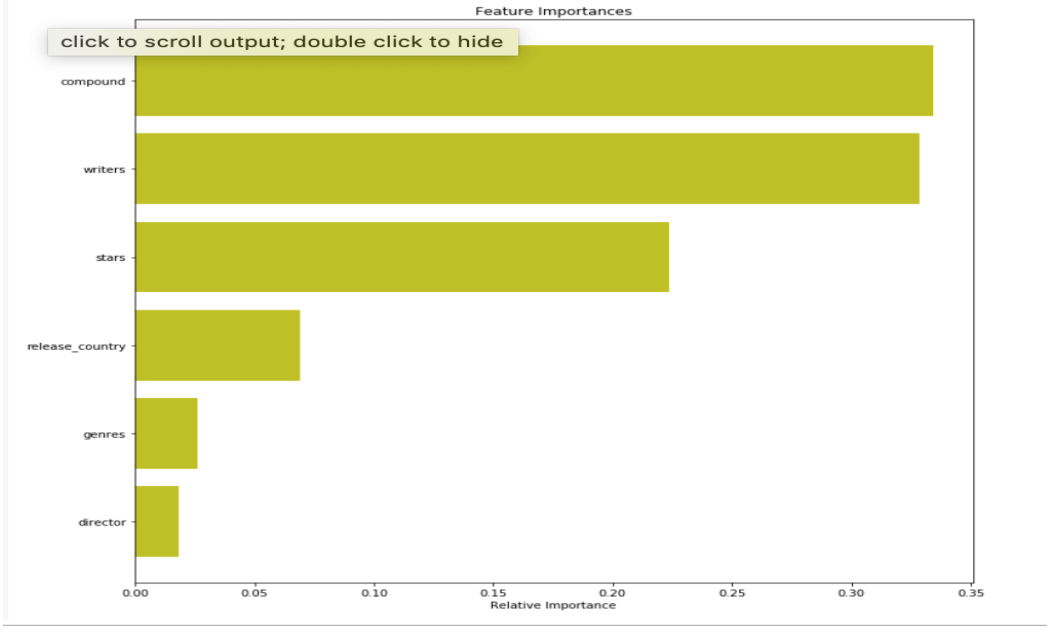

Elsa
Anna
Disney
character
story
love
Frozen
Spider Man
Peter Parker
superhero
Raimi
Green Goblin
comic book

[5]

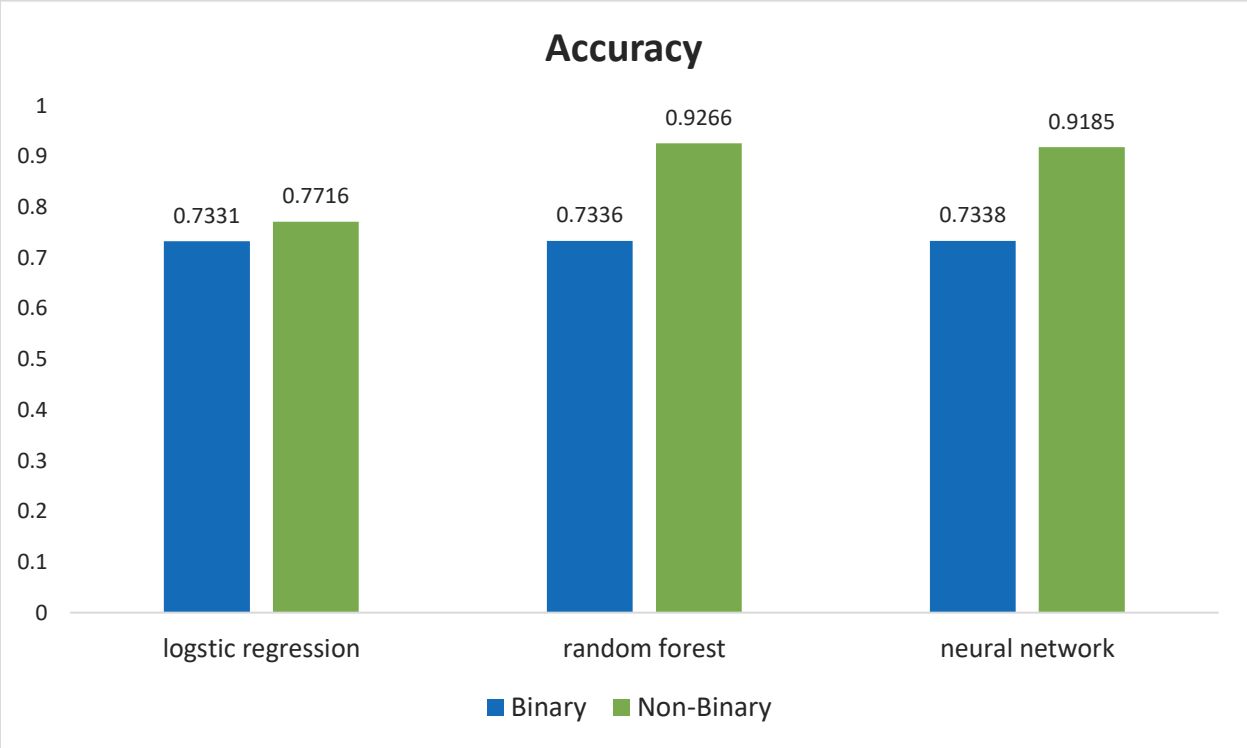


[6]


Feature Importances
compound_grade
writers_binary
storyline_binary
stars_binary
release_date_binary
release_country_binary
keywords_binary
genres_binary
director_binary
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Relative Importance


Feature Importances
click to scroll output; double click to hide
compound
writers
stars
release_country
genres
director
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Relative Importance

[7]


Accuracy
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.7331
0.7716
0.7336
0.9266
0.7338
0.9185
logstic regression
random forest
neural network
Binary
Non-Binary