Final Project Report: Movie Recommendation System

Topic

The main idea of this project is a movie recommendation system. It started when I accidentally found a movie that resembled my all-time favorite film, and I can spotted a lot many same characteristics. Hence, I think if there is a system to suggest movies based on movies as an input might be great. Furthermore, I know that streaming platforms such as Netflix also use algorithms to suggest the fittest show to the users, so I am interested in this topic.

Data Selection

The data I selected comes from Kaggle (https://www.kaggle.com/code/rounakbanik/movie-recommender-systems/input?select=movies_metadata.csv) .

There are 3 main tables that I used in this project which are movies_metadata.csv, and credits.csv which contain data about actors and directors and keywords.
Each table contains data around 45000 rows which contain some invalid data format

Data Cleansing

1. Some of movies_matadata table's ids have an invalid format (date format), so I spot them and drop them out
2. I merge those 3 tables together and then drop the duplicated ID, so the final number of table rows is 45432 rows.
3. I select only 14 features out of 27 features that would be impactful to clustering. The selected columns are 'budget', 'genres', 'id', 'original_language', 'overview', 'popularity', 'production_companies', 'revenue', 'spoken_languages', 'tagline', 'title', 'vote_average', 'cast', 'crew', 'keywords'.
4. I fill the NaN values with the proper format. '' for non-numeric data, mean value of the column for numeric data.

Data Preprocessing

1. The format of categorical data is in the format of dictionary -like, so I used ast library to extract the data and change it to be a list format.
2. The categorical data needed to be transform to be one-hot encoder format and some of the columns have to many labels, so I need to apply function to select only some most common one and leave others as 'others'.
3. For text data, I apply tf-idf method.
4. For numeric data, just make sure that it's in float format.
5. Concat these tables and don't forget to drop the original columns that used to creates one-hot encoder and tf-idf tables.

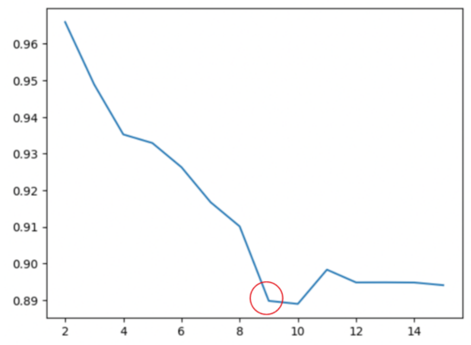
Training

I started with a cluster with several numbers of clusters. I used silhouette score for measurement and elbow method to find the most proper number of clusters.

The result shows that at n = 9, the score is slightly decreased, so this is the fittest number.

The score is 0.889

Elbow Method



I also apply DBSCAN clustering to compare the results. The score is 0.86 which is worse than KMeans method so I stay with a baseline solution.

The affine clustering also a choice to experiment but I haven't try it for now, so it will be considered in the future.

Result

For the result, the input is 3 movies and we expected to get output as 3 related movies in terms of genres, actors, directors, or film production company.

The system shows 3 movies but if the inputs didn’t have the same type of genre, the result tends to be nonsense.

To improve

1. We can apply leventine distance for the input process to avoid misspellings and mis format of the input movies.
2. For this version, it has limitation that the input movies must be in the database only, so we could make it more flexible.
3. We should try to apply other models.
4. We can improve the UI in the output page.